Rapid Prototype of Earth Observing Digital Twin of the NESDIS Ground System

11.30.2023

Prepared by Science and Technology Corp (STC)
21 ENTERPRISE PARKWAY, SUITE 150. HAMPTON, VA 23666

Prepared for
NOAA, NESDIS
Office of Systems Architecture and Engineering (SAE),
Joint Venture Partnerships

Table of Contents

Study Authors

Principle Investigator

Rebekah Esmaili, Senior Research Scientist, Science and Technology Corp.

Report Co-Authors

Krish Narasimhan, Cloud Solutions Engineer, Science and Technology Corp.

Mark Schoeberl, Chief Scientist, Science and Technology Corp.

Yi Wang, Data Scientist, Science and Technology Corp.

David Daniel, Data Scientist, Aeroxis Enterprises

Tiffanie Choi, Software Engineer, Science and Technology Corp.

Special Thanks

(NOAA) Ramesh Rangachar, Beau Backus, Eric Maddy, Ryan Berkheimer, Kesha Hayes, Rob Redmon, Flavio Iturbide-Sanchez, Lynn Mayo

(NASA) Sid Boukabara, Jaqueline Le Moigne, Benjamin Smith

(AWS) Rayette Toles-Abdullah, John Dwyer, Danny Sheehan, Jason Amoah

Disclaimer

1 Introduction

NESDIS data holdings continue to grow beyond the economic, operational, and maintenance capacity of on-premises computing resources. Large data providers like NESDIS are increasingly turning to cloud computing resources, as demonstrated by the in-progress implementation of the NESDIS Common Cloud Framework (NCCF), which will migrate the operational ingest, archive, product generation, product distribution and access of their data products. While cloud computing solves problems of scaling data and access to an ever-increasing pool of users, it is also an opportunity to explore how new techniques can monitor and improve data flows to users within the next-generation NESDIS ground system. Digital Twins are platforms that can be used to experiment and develop new systems that are operationally efficient and more user-friendly.

A digital twin is a virtual representation of a physical system and is used throughout industry to monitor real-time processes, test experimental services, and support decision-making. This report details our development of a prototype digital twin to model core processes in the next-generation NESDIS ground system. We used our digital twin to test how innovations like machine learning (ML)-based anomaly detection and deep learning data fusion can provide better services to end users. Our prototype Observing Digital Twin (EO-DT) also explored tools for user engagement that can support NESDIS management decision-making on next-generation ground systems.

The purpose of building a prototype was to explore and demonstrate key features and recommend emerging technologies, estimate costs, and to study the value an EO-DT can provide to NESDIS. In **Section 1**, we provide an overview of digital twins, some anticipated benefits of our study goals, and the goals of the prototype EO-DT demonstrations that informed our recommendations. **Sections 2-4** will deep dive into the tools, techniques, and scientific methods we used to develop our prototype and how our system can extend into an operational one[1] (**Section 5.1**). We also document the state-of-the-science of other significant data providers' efforts (**Section 5.2**), such as NASA, to build Earth System Digital Twins (ESDT). We also document the associated costs of the prototype and a proposed operational system[1] (**Section 5.3**). A summary of our recommendations and lessons learned is found in **Section 6**.

1.1 How Could a Digital Twin Benefit NOAA?

As shown in **Figure 1.1**, measurements collected by Earth Observing satellites are processed through a complex pipeline consisting of downlinking, ingesting, archiving, processing, and finally disseminating the measurements and derived data products. We call this data pipeline the ground system. Some users, weather forecasters, and other disaster management professionals need these data in near real-time. The ground system's objective is to ensure data is delivered rapidly and towards that objective must consider bandwidth limitations, efficiently execute millions of lines of code on high-performance computing platforms to get the job done.

When the system is operating normally, the data is delivered to the end user on time. However, what if a satellite sensor malfunctions? Or if there is a bug in a piece of the code? What if we want to add a new satellite to the data stream? What if the production team

[1] There is no planning for activity beyond the demonstration projects. See Disclaimer on page 4.

wants to experiment with a new technology? It can be challenging to troubleshoot problems or make changes to the existing ground system while delivering data to users. However, a virtualized equivalent environment such as a digital twin can enable rapid identification of problems and the implementation of solutions without disturbing the operational ground system.

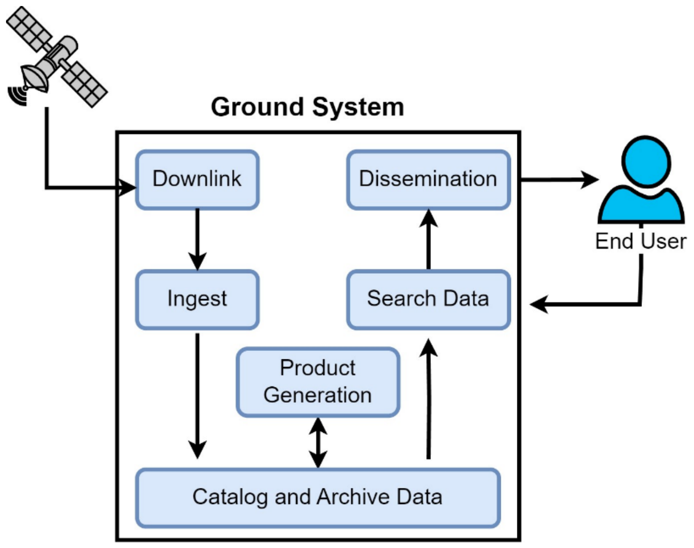


Figure 1.1 Simplified diagram of core processes in the NESDIS ground system. These are the core processes that need to be monitored and scaled to ensure timely data delivery to users.

1.1.1 What is a Digital Twin?

Many definitions of digital twins exist, but they share a common goal of representing a physical system virtually. A digital twin is a virtual representation of a system updated from real-time data and uses simulation and ML to help decision-making (IBM, n.d.). Historically, digital twins were employed in factory settings to mimic assemblies of products. However, digital twins can also replicate digital assets. This report defines a **digital twin as a virtual model of the physical ground system that is augmented with real-time data from satellite sensors, computing resources, and user interaction**.

Digital twins are beneficial because they (1) mirror existing processes to support decision-making and (2) enable research and design of systems and products without disrupting production. There is often confusion on the difference between digital twins and simulations. Both utilize models to replicate a system's various processes; however, a digital twin is a virtual environment, which makes it considerably richer for study. While a simulation typically studies one process, a digital twin can run multiple simulations to simultaneously study multiple processes and their mutual interactions. For example, a climate model may focus on Earth dynamics but not socio-economic impact. A digital twin can incorporate both variables to answer "what if" scenarios about the human impact of climate change.

1.1.2 What are the Benefits of Building a Digital Twin of the Ground System?

NESDIS's product portfolio consists of petabytes of satellite data and continues to increase daily. Managers of such large data providers often want to know how to optimize and scale their resources. However, it is not easy to assess the impact of implementing technological advancements on the operational ground system, which must be available 24/7 to protect life and property. An EO-DT is a digital twin framework designed to validate technological advancements (e.g., ML) before changing the operational ground system. For example, with

ML on the digital twin, it is possible to identify data demand patterns and dissemination bottlenecks, detect anomalies in satellite sensor data, and automate data quality control (Boukabara et al., 2019). In addition to optimizing the data flow, ML can facilitate data fusion for research and assimilation. ML thus has the potential to transform the current NESDIS data portfolio into a portfolio of analysis-ready data for highly efficient access by end users.

An EO-DT can also use real-time data generated by the ground system to monitor the dataflow for anomalies, efficiently allocate system resources, and combine datasets. Interest in satellite data is expected to grow more than all NOAA data assets, and an EO-DT can ensure that the infrastructure keeps up with future user needs. Furthermore, the EO-DT could support NOAA management when studying the impacts of employing ML and new satellite sensors in operations.

This report documents our efforts to build an EO-DT prototype to study the technologies needed, the associated costs, and the expected benefit to NOAA. Our prototype EO-DT has two roles, (1) to monitor and optimize the ground system delivery pipeline and (2) to serve as a development sandbox for enhancements without interrupting the existing system. Successful new technologies within the EO-DT can then be deployed into the ground system or integrated as an add-on using two-way communication channels.

While there are many benefits, deploying a fully capable digital twin could be costly and resource-intensive. Since this is a new area of research at NOAA, some of the questions we sought to answer through our efforts are:

- How much will it cost to develop a fully capable digital twin?
- What are the best commercial and open-source tools to use?
- Can ML improve the performance of some of the ground system components?
- What data formats are optimal within the digital twin?
- Can the digital twin improve data access to end users?
- Can the digital twin meet or exceed the latency requirements?
- How can the digital twin enable data fusion and ML?

1.1.3 Study Goals

For the EO-DT project, we built a simplified prototype that enables realistically estimate costs and a hands-on learning experience to identify strengths and weaknesses of new approaches and technologies. Our specific project goals were to:

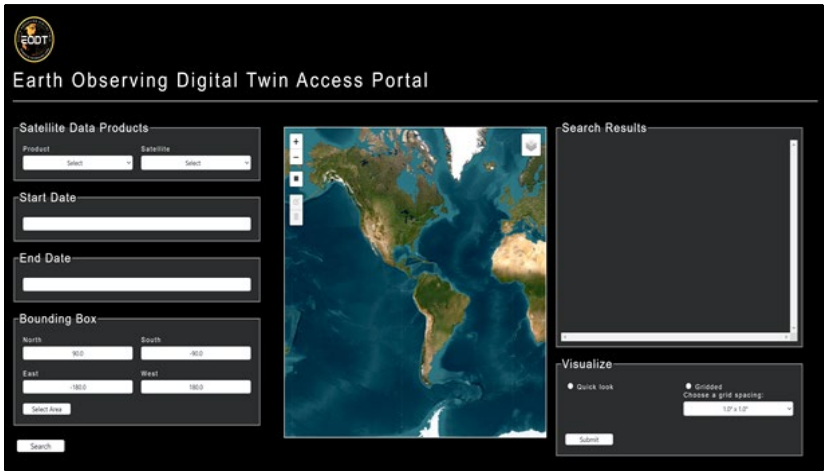
- Rapidly implement a prototype EO-DT of the current NESDIS ground system.
- Build and use functional components that enable a study of the challenges of developing a fully functional EO-DT and producing realistic cost estimates.
- Explore the strengths and limitations of using pre-built commercial systems to run the EO-DT. We will recommend which services can be commercially sourced and which should be custom-built.
- Employ ML to demonstrate use cases where the EO-DT can be used to model future advancements in the NESDIS ground system.

Our deliverables included (1) the technological assets used for the study, (2) bi-weekly progress meetings with the government, (3) kickoff, midterm, and final demonstration meetings, and (4) a final report with the results of the study.

1.1.4 Three Demonstrations to Inform Study

We developed three technology demonstrations to inform this study and to show how significant components could be built. Each demonstration touched upon one of our primary study goals.

1.1.4.1 Demo #1: Use real-time data to model current capabilities from satellite archives, computing resources, and user interaction.


Earth Observing Digital Twin Access Portal

Satellite Data Products
Product: Select
Satellite: Select

Start Date

End Date

Bounding Box
North: 90.0
South: -90.0
East: -180.0
West: 180.0

Search Results

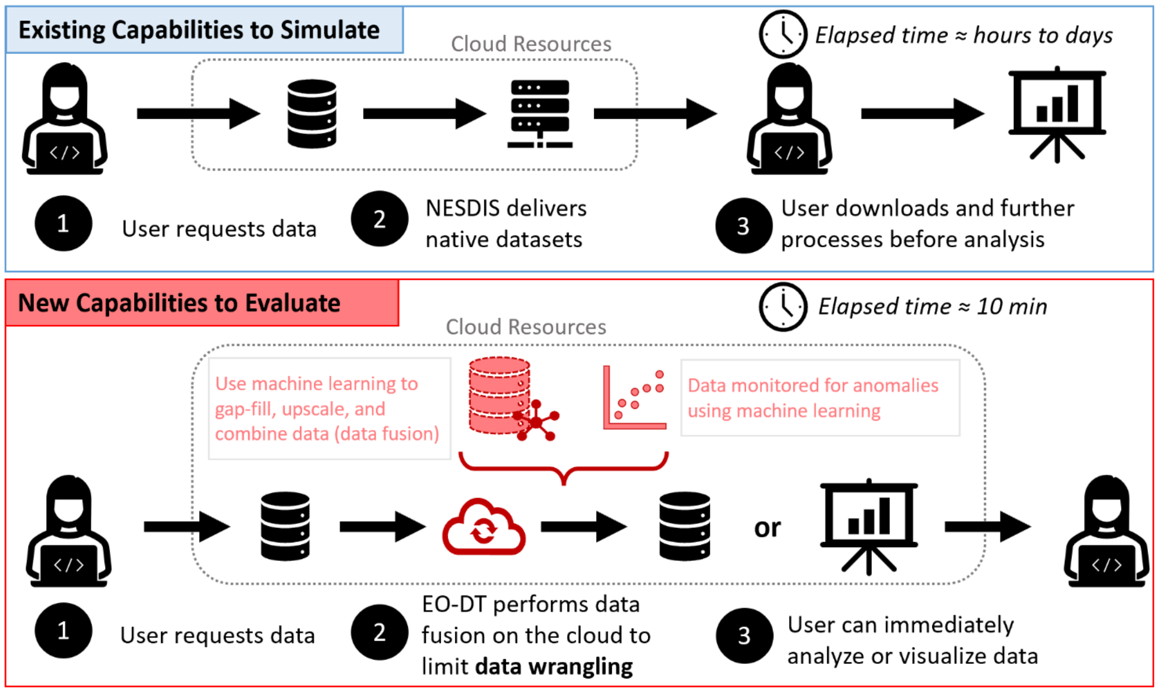
Visualize
Quick look
Gridded
Choose a grid spacing: 1.0° x 1.0°

Figure 1.2 Screen capture of the landing page. We incorporated a "keep it simple" design approach to not overwhelm users.

Key to the success of a ground system is its ability to process satellite data and deliver it rapidly and efficiently to the end-user in a manner that is most compatible with the user's workflow. We modeled key components of the existing NESDIS user data delivery system to accomplish this goal. Users can query data holdings on NESDIS's cloud services via a web portal. We built a user interface for searching and visualizing NESDIS data on the cloud using Amazon Web Services (AWS) (**Figure 1.2**). We incorporated a "Keep It Simple" philosophy on the user interface. While users may need complex features in a fully operational digital twin, we aimed to minimize clutter from the interface, a la the Google search interface. Google remains visually simple, a design element that continues to lead to its popularity. Advanced search features are possible in Google, but these are shown to users after the initial query.

Presently, users can query the NESDIS archive using the NOAA Comprehensive Large Array-data Stewardship System (CLASS) portal (**Figure 1.3**, top). The user provides a data product name, geolocation coordinates, and time. A link to an FTP download location is emailed to a user that could take even hours, depending on the volume of returned results.

Our user interface reconciles differences in the existing data delivery systems with those anticipated in the NESDIS Common Cloud Framework (NCCF). More recently, the NOAA Open Data Dissemination (NODD) has opened public data access to cloud-based archives for JPSS and GOES data products. This newer access point does not have an official query portal, so we developed a user-interface with the querying capabilities of CLASS along with a capability to evaluate new services to users, such as data fusion.



Existing Capabilities to Simulate
Cloud Resources
Elapsed time ≈ hours to days
1 User requests data
2 NESDIS delivers native datasets
3 User downloads and further processes before analysis
New Capabilities to Evaluate
Elapsed time ≈ 10 min
Cloud Resources
Use machine learning to gap-fill, upscale, and combine data (data fusion)
Data monitored for anomalies using machine learning
or
1 User requests data
2 EO-DT performs data fusion on the cloud to limit data wrangling
3 User can immediately analyze or visualize data

Figure 1.3 Existing user experience (top) allowed users to query and download data from the NESDIS archive. Users then spent significant time "wrangling" the data (combining, regridding, collocating) before they can analyze the results. In addition to simulate the existing capabilities, the EO-DT can evaluate new capabilities, such as exploring machine learning methods for data fusion, on demand cloud services, and novel anomaly detection schemes. These data are in an "analysis ready" format and the user can more quickly conduct research.

A capability demonstrated in our system, which represents a potential enhancement to the ground system, is that all user requests are processed on-demand (**Figure 1.3**, bottom). On-demand means that all backend cloud computing resources required to process the user request are launched only when the request arrives and are turned off after the necessary processing. In the past, data archive maintainers like NOAA often pre-process datasets to speed up data delivery. For example, the VIIRS Aerosol Optical Depth Level 2 data products are routinely gridded into a Level 3 data product. While such an approach may be appropriate when there is sufficient and continual demand for the products, it will be inefficient and incur needless costs for data with sporadic demands.

Furthermore, pre-processing forces users to work with already defined gridding and aggregation schemes, which may meet the needs of most, but not all, end users. Users spend significant time performing data wrangling - transforming and organizing raw data into a suitable format for further analysis. On-demand processing would reduce the time users spend data wrangling by allowing users to build custom aggregation schemes. By processing on-demand, we also can collect data on which datasets a user wants to aggregate and how they wish to aggregate it and perform a cost-benefit analysis on whether it is more economical to add such a product to operations or only compute it on demand. For example, if users repeatedly request certain aggregated data sets, the system can store those data sets for even faster retrieval.

A key feature of our approach is that we adopted a data-in-place philosophy to reduce cloud computing costs. We do not make any local copies of NODD data within our digital twin; instead, we monitor these resources from afar and ingest only the metadata of the datasets into our query database for cataloging and searching purposes. Minimally moving data around in, to, and from the cloud reduces costs and processing time.

1.1.4.2 Demo #2: Explore where new services and ML can enhance the user experience and limit data wrangling

In our EO-DT prototype, we demonstrated the use of ML for improved sensor anomaly detection. Currently, sensor data anomaly detection is based on quality flags. ML can classify images based on complex anomaly scenarios across an entire scene. The goal was not to remove the data from production but to better communicate to users and management if a dataset has significant degradation. Some examples are shown in **Figure 1.4**.

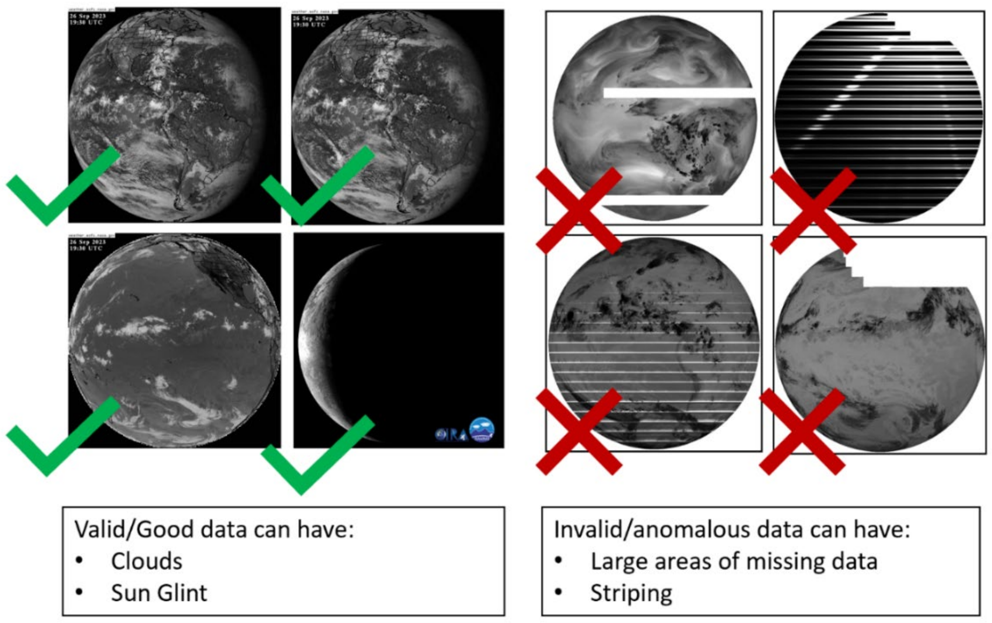

Valid/Good data can have:
• Clouds
• Sun Glint
Invalid/anomalous data can have:
• Large areas of missing data
• Striping

Figure 1.4 Examples of good quality GOES imagery (**left**) and imagery that has anomalies present (**right**). We demonstrated how we applied a convolutional neural networks (CNN) method to identify anomalies in imagery data from both GOES and VIIRS. Machine learning techniques can potentially identify complex anomaly patterns that may be missed by pixel-level quality flags alone.

In ML, this task is fundamentally a binary image classification problem. We used Residual Network (ResNet)-18, a convolutional neural network (CNN) that is 18 layers deep, trained on the ImageNet database. We trained ResNet-18 on 2-3 years of GOES-series ABI and JPSS-series VIIRS L1b data from all channels. Offline validation showed that our model was 99.2% accurate for ABI and 89% for VIIRS. We installed our most accurate GOES-16 model into operation and monitored every file for anomalies.

Figure 1.5 shows the processing workflow for our ResNet-18 model. At run-time, the model loads pre-trained model weights developed using our 2-3 years of ABI data. The model is then provided with a real-time, unseen GOES image, and then it provides a classification, valid (anomaly-free) or invalid (contains anomalies). VIIRS data was used only for research purposes. More details on this project are provided in **Section 3.1**. We also display these ML anomalies on a dashboard to support decision-making for NOAA management (See **Demo #3** and **Section 2.5** Data Analytics Dashboard for more information).

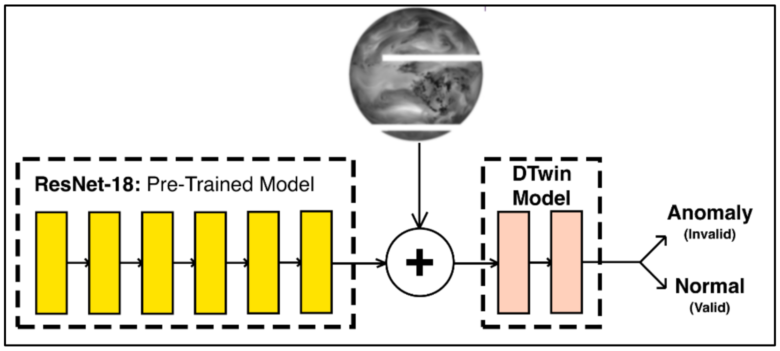
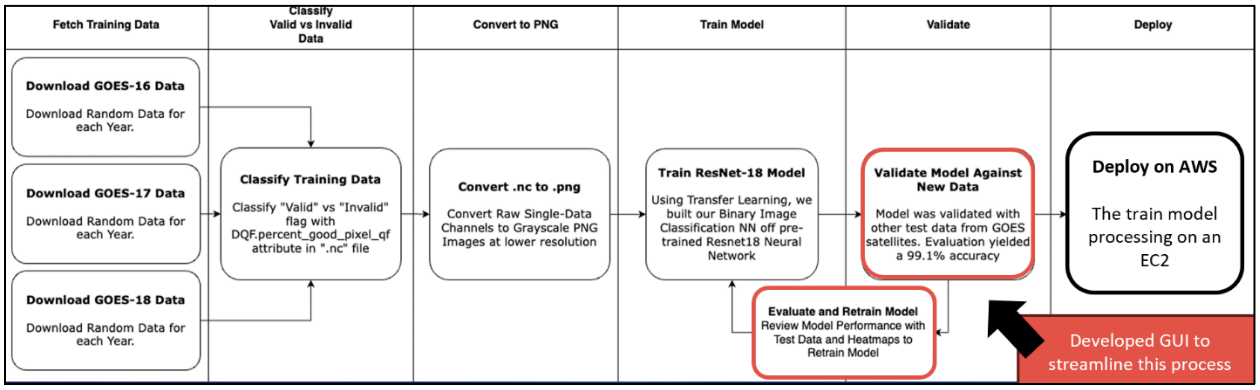


Figure 1.5 Simple overview of how ResNet-18 in operations. ResNet-18 loads an existing pre-trained model and is provided with a real time images of ABI L1b imagery data (all channels). The ResNet-18 model then classifies the image as "valid" or "invalid" depending on if the scene is anomaly free or contains anomalies, respectively.

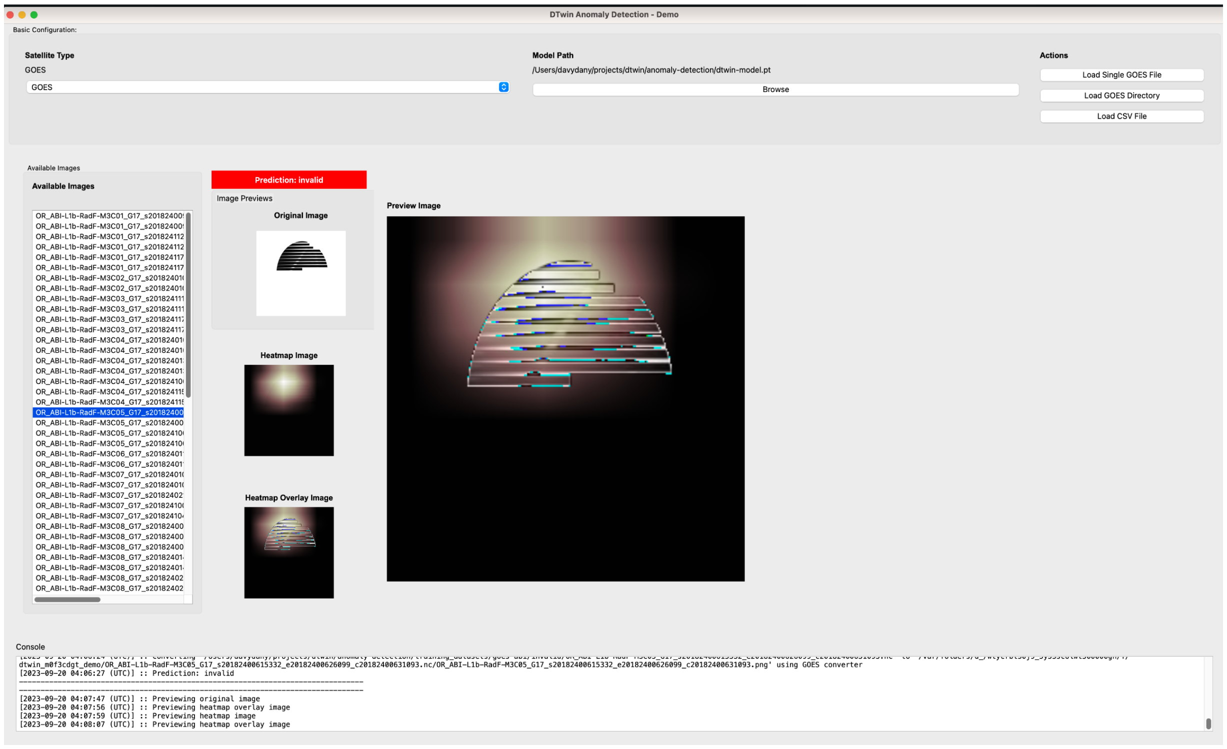
We aimed to develop a proof-of-concept (*Lavin* and Renard, 2020) anomaly detection scheme and document the computing requirements and workflow unique to ML methods. Our complete development workflow is shown in **Figure 1.6**. A total of 4,550 randomly selected images from 2-3 years of GOES ABI data were downloaded from the NODD. One challenge of conducting ML research is the lack of pre-labeled training data. For that, we classified identifies images like those in **Figure 1.4** as "valid" or "invalid" using the quality flags inside the file. The data needed to be reformatted from netCDF, the native file format of the ABI, to PNG format, which processes more quickly in the ML model. The images were converted to greyscale, downscaled from 5000x5000 pixels to 240x240 pixels, and randomized flipping across the horizontal and vertical axis to improve the accuracy of feature identification.



Fetch Training Data
Classify Valid vs Invalid Data
Convert to PNG
Train Model
Validate
Deploy
Download GOES-16 Data
Download Random Data for each Year.
Download GOES-17 Data
Download Random Data for each Year.
Download GOES-18 Data
Download Random Data for each Year.
Classify Training Data
Classify "Valid" vs "Invalid" flag with DQF.percent_good_pixel_qf attribute in ".nc" file
Convert .nc to .png
Convert Raw Single-Data Channels to Grayscale PNG Images at lower resolution
Train ResNet-18 Model
Using Transfer Learning, we built our Binary Image Classification NN off pre-trained Resnet18 Neural Network
Validate Model Against New Data
Model was validated with other test data from GOES satellites. Evaluation yielded a 99.1% accuracy
Evaluate and Retrain Model
Review Model Performance with Test Data and Heatmaps to Retrain Model
Deploy on AWS
The train model processing on an EC2
Developed GUI to streamline this process

Figure 1.6 The development workflow for the machine learning anomaly detection method tested in our demonstration. Training required labelling thousands of images and performing image transformations to improve the training model quality. Because machine learning models need to be easily updated to accommodate future changes to the data, our workflow heavily used automation and we developed a GUI tool to facilitate validation, which requires the data scientist to examine many images. The best model was then deployed onto AWS and operationally checks all new GOES ABI L1b data for anomalies.

During the training and validation phases, thousands of images were generated and required human inspection. Furthermore, it can be challenging to identify why a model like ResNet-18 chooses a particular image classification. To streamline the testing and validation process, we developed a Python-based GUI to improve the data inspection and to incorporate explainable ML to improve understanding (**Figure 1.7**). Like other CNNs, ResNet-18 cannot explain "why" it classified an image as valid/invalid. Heatmaps provide "visual explanations" for decisions from CNN model families. Heatmaps determine the model attention mechanisms for each convolutional layer (e.g., ResNet-18 has 18 layers) using Gradient-weighted Class Activation Mapping (Grad-CAM). Attention allows a model to focus on different parts of the input data, assigning varying degrees of importance or relevance to different elements. Our GUI displays where the ResNet-18 model focused attention on the GOES imagery using the final layer, the last layer focused before the ResNet-18 model assigned its classification. Given that numerous models were trained and tested, the GUI can easily swap with other models to compare and select the best model for anomaly detection.



Basic Configuration:

Satellite Type: GOES

Model Path: /Users/davydany/projects/dtwin/anomaly-detection/dtwin-model.pt

Available Images

```
OR_ABI-L1b-RadF-M3C01_G17_s20182400
OR_ABI-L1b-RadF-M3C01_G17_s20182400
OR_ABI-L1b-RadF-M3C01_G17_s201824112
OR_ABI-L1b-RadF-M3C01_G17_s201824112
OR_ABI-L1b-RadF-M3C01_G17_s201824117
OR_ABI-L1b-RadF-M3C01_G17_s201824117
OR_ABI-L1b-RadF-M3C02_G17_s20182401
OR_ABI-L1b-RadF-M3C02_G17_s20182401
OR_ABI-L1b-RadF-M3C03_G17_s201824111
OR_ABI-L1b-RadF-M3C03_G17_s201824111
OR_ABI-L1b-RadF-M3C03_G17_s20182411
OR_ABI-L1b-RadF-M3C03_G17_s20182411
OR_ABI-L1b-RadF-M3C04_G17_s20182401
OR_ABI-L1b-RadF-M3C04_G17_s20182401
OR_ABI-L1b-RadF-M3C04_G17_s20182401
OR_ABI-L1b-RadF-M3C04_G17_s20182401
OR_ABI-L1b-RadF-M3C04_G17_s20182410
OR_ABI-L1b-RadF-M3C04_G17_s20182411
OR_ABI-L1b-RadF-M3C04_G17_s20182411
OR_ABI-L1b-RadF-M3C05_G17_s20182400
OR_ABI-L1b-RadF-M3C05_G17_s20182400
OR_ABI-L1b-RadF-M3C05_G17_s20182410
OR_ABI-L1b-RadF-M3C05_G17_s20182410
OR_ABI-L1b-RadF-M3C06_G17_s20182401
OR_ABI-L1b-RadF-M3C06_G17_s20182401
OR_ABI-L1b-RadF-M3C07_G17_s20182401
OR_ABI-L1b-RadF-M3C07_G17_s20182401
OR_ABI-L1b-RadF-M3C07_G17_s20182402
OR_ABI-L1b-RadF-M3C07_G17_s20182410
OR_ABI-L1b-RadF-M3C07_G17_s20182410
OR_ABI-L1b-RadF-M3C08_G17_s20182400
OR_ABI-L1b-RadF-M3C08_G17_s20182400
OR_ABI-L1b-RadF-M3C08_G17_s20182400
OR_ABI-L1b-RadF-M3C08_G17_s20182401
OR_ABI-L1b-RadF-M3C08_G17_s20182401
OR_ABI-L1b-RadF-M3C08_G17_s20182402
OR_ABI-L1b-RadF-M3C08_G17_s20182402
```

Console

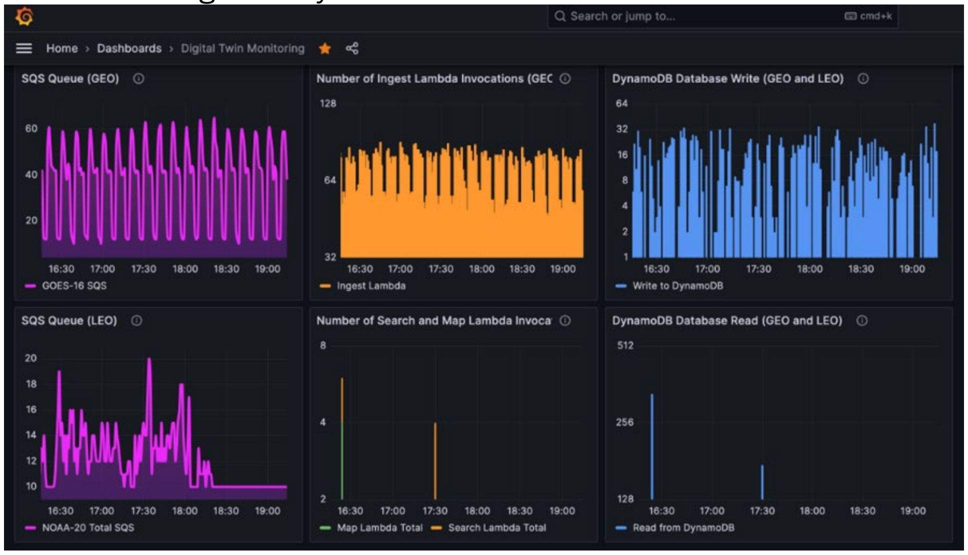
```
[illegible]
dtwin_m0f3cdgt_demo/OR_ABI-L1b-RadF-M3C05_G17_s20182400615332_e20182400626099_c20182400631093.nc/OR_ABI-L1b-RadF-M3C05_G17_s20182400615332_e20182400626099_c20182400631093.png' using GOES converter
[2023-09-20 04:06:27 (UTC)] :: Prediction: invalid
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
[2023-09-20 04:07:47 (UTC)] :: Previewing original image
[2023-09-20 04:07:56 (UTC)] :: Previewing heatmap overlay image
[2023-09-20 04:07:59 (UTC)] :: Previewing heatmap image
[2023-09-20 04:08:07 (UTC)] :: Previewing heatmap overlay image
```

Figure 1.7 Demo #2: ML-based data anomaly detection models

By developing our proof-of-concept anomaly detection system, we determined there are unique considerations for ML models that can be tested and optimized in the EO-DT. For example, large samples of labeled training datasets can improve anomaly detection accuracy. The work on our project is a promising start and, in the future, could lead to multi-label classifications to rapidly prescreen data sets. We have also determined that NetCDF4 and other NOAA archive formats are not well-suited for ML research, but data transformations to ML-friendly formats (like PNG) could be built into a digital twin.

1.1.4.3 Demo #3: Support decision-making related to optimizing NESDIS data processing

Digital twins can monitor processes in real time and generate a lot of valuable metrics. Converting this data into information is critical to answering what-if scenarios. An analytics dashboard can support decision-making by providing a one-stop location for real-time system monitoring for NOAA management. Examples of analytics include system performance metrics (e.g., accuracy and reliability) and the timeliness of different data fusion techniques. These metrics help monitor the data quality performance of the system regarding completing user requests and performing cost assessments for current and future additions to the ground system.



Search or jump to...
Home › Dashboards › Digital Twin Monitoring
SQS Queue (GEO)
Number of Ingest Lambda Invocations (GEO
DynamoDB Database Write (GEO and LEO)
GOES-16 SQS
Ingest Lambda
Write to DynamoDB
SQS Queue (LEO)
Number of Search and Map Lambda Invoca
DynamoDB Database Read (GEO and LEO)
NOAA-20 Total SQS
Map Lambda Total
Search Lambda Total
Read from DynamoDB

Figure 1.8 Demo #3: Real-time analytics dashboard

We built a dashboard (**Figure 1.8**) using Grafana for our demonstration. Grafana is an open-source analytical and visualization tool that consists of multiple individual panels arranged in a customizable grid. Grafana could easily make a time series of system resource usage and data processing requirements, such as data product latency and the timeliness of the system's response to user requests. Grafana can display statistics on user interaction, such as which NESDIS products were accessed, when, and over what region, alongside data fusion requests were made.

Grafana interfaces seamlessly with AWS CloudWatch, a service that monitors AWS processes. On-demand, modular cloud resources are easier to monitor than complex on-premises systems. Our dashboard supports custom scripting monitoring tasks, such as data processing time and user queries. In our prototype demonstration, we used Grafana Cloud, where Grafana hosts the portal. While this service was adequate for our demonstration, we recommend using Grafana hosted on AWS in an operational EO-DT[2].

1.2 Digital Twin Framework

1.2.1 Concept of Operations

Our three demonstrations show an interlinked system of processes. This section emphasizes the primary processes we simulated and describes how they interact. The NESDIS ground system's existing and planned "real" processes include the downlink, ingest, product generation, catalog and archive, search, and dissemination. We have also simulated new services within the digital twin, which we employ to investigate how an EO-DT can serve as an experimental platform for improving the ground system, especially those enhancements that leverage ML. Some example enhancements include ML-based anomaly monitoring (**Section 3.1**) and data fusion operations (**Section 3.2**). We chose these improvements because we prioritize the user experience at the core of our approach. It is important to note that the objective of our demonstration was not to achieve flawless replication of the ground

[2] There is no planning for activity beyond the demonstration projects. See Disclaimer on page 4.

system but, instead, to gather insights for our final report and study recommendations and to create proofs-of-concept for new services.

Figure 1.9 shows the concept of operations of our EO-DT, both in terms of how we model existing capabilities (blue) and what new services we explored (red). The left-hand side shows the key ground system capabilities (the "real" system) and matches that in **Figure 1.1**. The right-hand side shows how these processes and new services flow in our digital twin. Starting from ❶ in the top right, data populates the NODD cloud archives, publicly available datasets on commercial platforms such as AWS. We utilize the NODD because (i) the data is already in the cloud and thus provides easy and secure connections to our cloud-based EO-DT and (ii) it is more representative of the future NESDIS archive that will be available after the NCCF has been fully implemented. When a new file is available on the NODD, the EO-DT is notified and retrieves essential metadata about the file ❷, such as the product name, satellite source, start and end times, computes geolocation, and stores all these metadata in the enhanced digital twin catalog database (**Sections 2.3.1–2.3.2**). The satellite data remains on the NODD and is not duplicated within the digital twin. In ❸, each new GOES-18 L1b full disc file (all channels) is monitored for anomalies using a ML approach that we describe in detail in **Section 3.1**. While we only monitored GOES-18 files operationally, our methodology was tested with VIIRS L1b datasets with promising results. As described, anomaly detection is an example of a new service that has the potential to classify complex anomaly patterns to better communicate data quality to both end users and NOAA management.

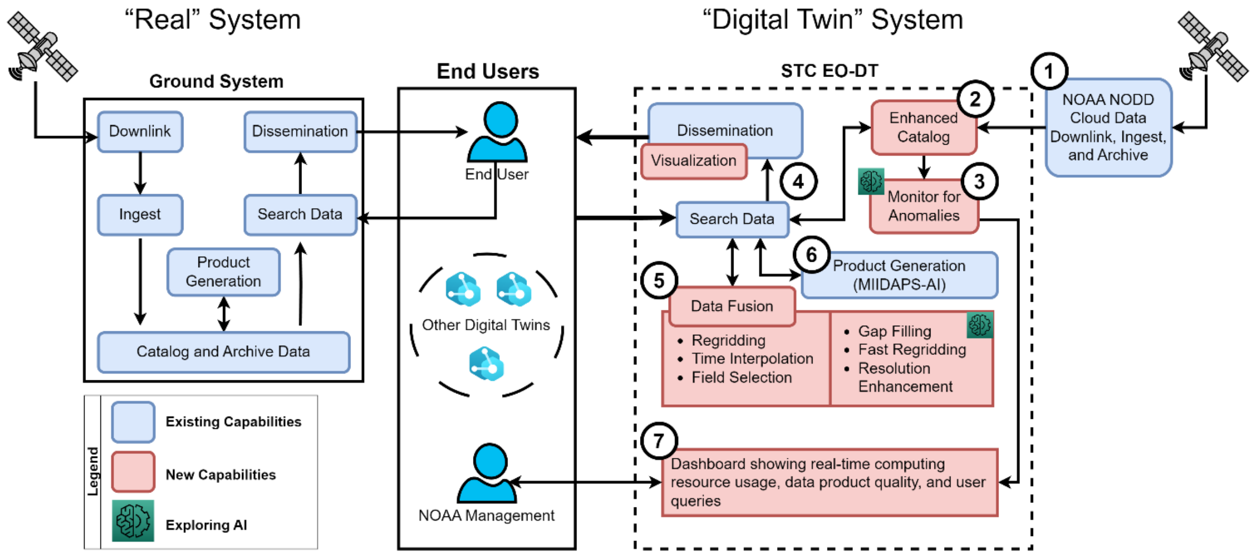


Figure 1.9 The concept of operations for the STC EO-DT prototype. The STC EO-DT models key ground system processes and provides a sandbox to explore new features, such as ML-based anomaly monitoring, to explore user experience what if scenarios. Improving the user experience was central to our approach.

In CLASS, the current operational user interface for data searches allows users to search and retrieve (order) data based on the product name, date, time, and location from the NCEI archive. After completing the search, CLASS provides a link to end users to visualize their ordered data. However, CLASS currently lacks the capability for direct data visualization.

In the digital twin, we developed a cloud-based portal that serves as a front-end to order NODD data ❹. The portal also integrates a planimetric (plan) view capability to display the ordered datasets, offering users a rasterized visual on an open-source global map (**Section 2.3.4**).

Working with NOAA project management, we identified data fusion as a critical service to test in the digital twin. Within our EO-DT prototype, we developed a data fusion scheme ❺ to perform common transformations to align dissimilar data. Data from the same NOAA satellite sensor can have varying resolutions, gridding, and temporal sampling. We developed cloud-friendly code that can be deployed to perform these transformations rapidly as shown in our demonstration (**Section 2.4**).

We developed a proof-of-concept for two ML-based data fusion enhancements (**Section 3.2**) that can augment our "classical" regridding approach. The two enhancements address the significant challenges of combining satellite datasets with other products, models, and in-situ observations, such as (i) the presence of a significant number of missing values in satellite scenes due to atmospheric clouds and (ii) too coarse of a spatial resolution of satellite data products to be fully exploited for their target application. Atmospheric clouds can obscure the Earth's surface from satellite visible, infrared, and some microwave sensors, thereby affecting their derived products. To overcome this, we explored gap filling clouds using Long Short-Term Memory (LSTM), a recurrent neural network (RNN) architecture used in Deep Learning. We also explored using Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN), a model that can be trained to leverage LEO and GEO observations to produce a high-resolution image when only low-resolution data is available. Both ML techniques showed promise in overcoming these challenges.

Installing new algorithms into the ground system is a critical component of the NCCF. In our prototype, we demonstrated how an ML-based algorithm could be containerized and installed ❻. To do this, we installed a version of the multi-instrument inversion and data assimilation preprocessing system, artificial intelligence version (MIIDAPS-AI) algorithm. MIIDAPS-AI estimates vertical profiles of temperature and moisture, surface temperature, surface emissivity, and cloud parameters from multiple instruments (Maddy and Boukabara, 2021). MIIDAPS-AI is not presently an operational NESDIS product, but given that there are many ML-based retrievals in development and the future, it is an important example to study. The NCCF must be able to accommodate the different development and operational needs, such as improved automation for deploying training models that are routinely updated.

Each of the steps described above produces a wealth of actionable real-time data. For example, user request processing times, products of interest, and computing resource usage. We developed a visualization system for all these analytics ❼ using a web-based dashboard. This feature distinguishes a digital twin from a traditional simulation: we simulate many elements of the whole system and capture data in real-time to support decision-making.

All the above enhancements were made to demonstrate a user-centered design that focuses on the simplicity of the user interface to get data of interest as quickly as possible. While the existing ground system provides invaluable services to the end user, our prototype explored how new services can be added. Furthermore, we explored ways to improve support and benefit for the NOAA management user responsible for the ever-increasing number of satellites, data products, and data volume. The on-demand resources and 'data in place' approach can reduce operational cloud costs substantially without compromising performance scalability to meet increasing user demands.

Finally, we are looking to a future that consists of a federation of digital twins. Through correspondence with other digital twin efforts at NASA and Destination Earth (DestinE), the community has a vision of an interconnected set of modular digital twins. Much work and discussion are centered around interoperability, or how to work together, share data, and exchange actionable information seamlessly. We describe our interactions with the community in **Section 5.2** and our recommendations for harmonizing efforts.

1.2.2 Ground System Components Modeled in the Digital Twin

1.2.2.1 Downlink and Ingest

In the ground system, data are staged upon receipt to ground stations in Svalbard, Norway McMurdo Antarctica, and Suitland, MD. Our EO-DT modeled the behavior and properties of the existing downlink system using CrIS and ATMS Level 1 assets by monitoring the satellite sensor performance, such as coverage, quality assurance, and latency, using a Grafana dashboard (**Figure 1.10**). We tracked these statistics to monitor for bottlenecks and anomalies in real-time in the dashboard. This one-stop location makes it easier for a resource manager to respond to problems. For example, **Figure 1.10 a** and **b** show the mean and time series of latency of CrIS and ATMS products generated on the NODD on 09/27/2023 18-21 UTC. Access to the NOAA operational system requires strict security, so we used the NODD, which is public, as a proxy for internal NOAA operations. NESDIS EO-DT project management requested that the EO-DT have a two-week data window to simulate the real-time data flow to monitor coverage, latency, and data quality flags for all products. We exceeded this requirement using our data-in-place approach (See **Section 1.1.4**). Because we were only storing metadata, the database requires little storage (< 1GB), so we could retain a complete data record from June 2023 through November 2023, when the project was completed.

As shown in **Figure 1.9 ❷**, one enhancement we explored is using ML techniques to detect anomalies in imagery data from the GOES-16/-17/-18 ABI and NOAA-20 VIIRS L1b products. **Figure 1.11a** shows the Grafana state timeline of GOES-18 Channel 16 from 09/27/2023 18-21 UTC. State timelines are helpful to visualize whether the data are classified as valid (anomaly-free) or invalid (contains anomalies) using our ResNet-18 model (**Section 1.1.4.2** and **Section 3.1**). As described in **Section 3.1**, our system classifies each new GOES-18 image from the NODD, and the dashboard displays the classification. At 20:10 UTC, an anomaly was detected by the model. Upon inspecting the imagery (**Figure 1.11b**), a small part of the south pole region was missing from the full disk image (circled) at 20:10 UTC while the other images were fully filled in. While quality flags are invaluable to determining whether data meets requirements, ML techniques can determine problematic data patterns. In our demonstration, we used a binary classification where valid or invalid were the two states. By building on this work, it is possible to train a multi-label classification model to not only find invalid images but also to predict what kind of anomaly patterns are within the image.

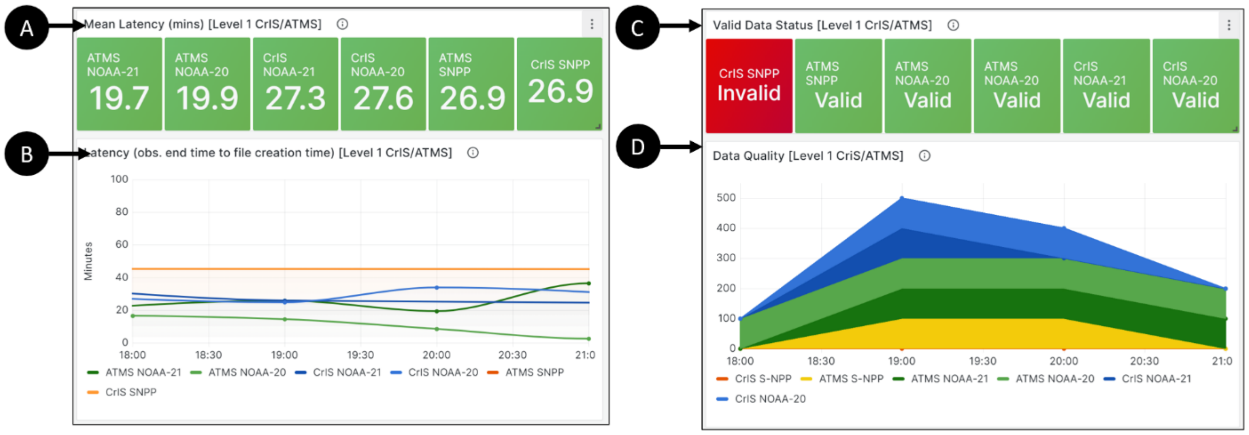


Figure 1.10 The Grafana dashboard showing the real-time latency and quality information for Suomi NPP, NOAA-20/-21 CrIS and ATMS on the NODD at 09/27/2023 18-21 UTC. (a) The mean hourly latency in minutes for the full record (b) the mean hourly latency time series. Latency is calculated from the end time and the file creation time to approximate the data processing time. (c) The data quality, which displays valid when >80% of quality flags are "good" and invalid otherwise. CrIS SNPP shows invalid because of ongoing sensor malfunctions. (d) The "good" quality flag percent time series in percent. The values for each product range from 0 to 100% and are stacked so all variables are visible on the graph.

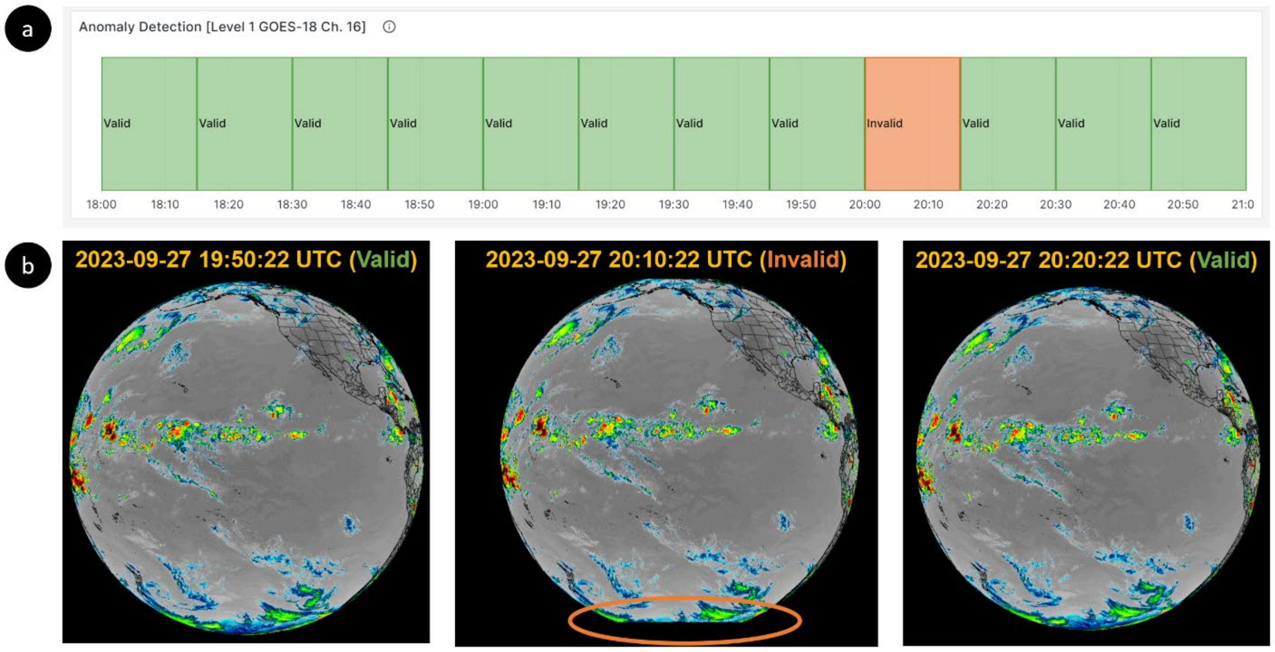

a
Anomaly Detection [Level 1 GOES-18 Ch. 16]
Valid
Invalid
b
2023-09-27 19:50:22 UTC (Valid)
2023-09-27 20:10:22 UTC (Invalid)
2023-09-27 20:20:22 UTC (Valid)

Figure 1.11 (a) Grafana state timeline display showing the quality status of GOES-18 data using machine learning image classification. At 20:10 UTC an anomaly was detected, and a visual inspection shows there was a small data outage in Antarctica (circles). Other images were complete. The example shown is for GOES-18 ABI channel 16 at 09/27/2023 18-21 UTC. Imagery Source: CIRA SLIDER.

1.2.2.2 Processing

In our EO-DT, processing encompasses Level 1 and Level 2 product generation and related services. We worked with the NESDIS EO-DT project management team to run a version of MIIDAPS-AI (Maddy and Boukabara, 2021) to simulate how our EO-DT could run and monitor retrieval algorithms. MIIDAPS-AI uses ATMS and GFS as inputs to predict profiles of temperature, relative humidity, total precipitable water, and cloud liquid water. **Figure 1.12** shows a single granule processed by MIIDAPS-AI, which has a total processing time of 246s, of which the ML model prediction only took 3 seconds to run on our system (**Section 5.3.1** describes the compute environments used).

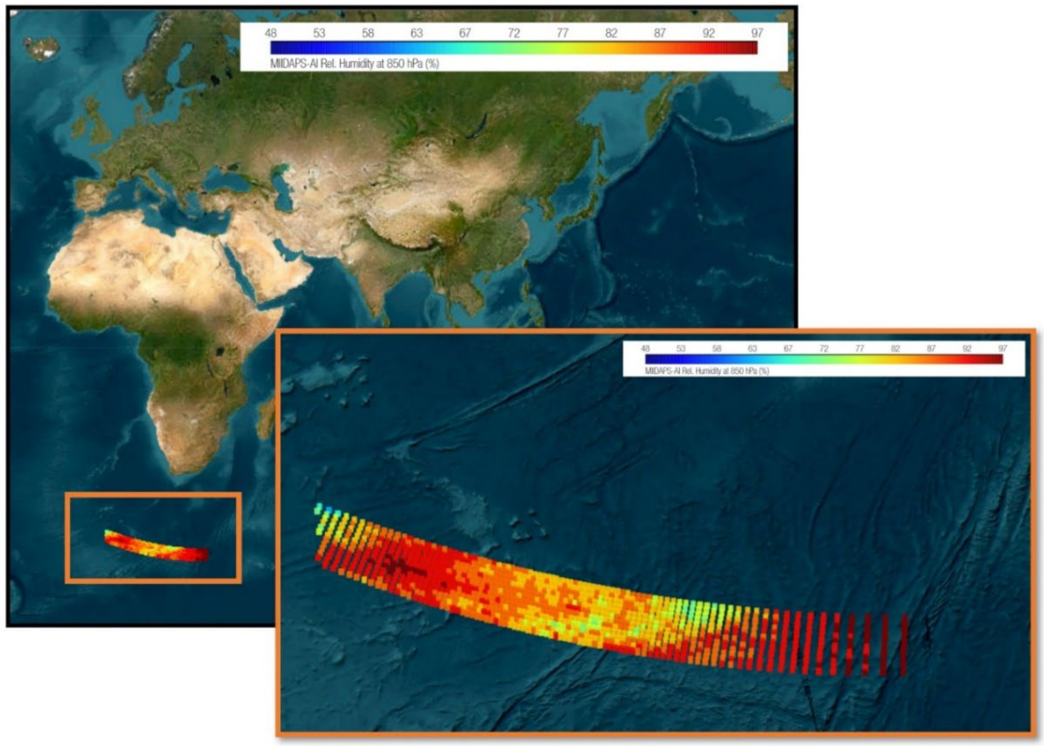

48
53
58
63
67
72
77
82
87
92
97
MIIDAPS-AI Rel. Humidity at 850 hPa (%)
48
53
58
63
67
72
77
82
87
92
97
MIIDAPS-AI Rel. Humidity at 850 hPa (%)

Figure 1.12 Relative Humidity at 850 hPA from MIIDAPS-AI on Sept 7, 2023 00:00 UTC. The ATMS granule was processed on-demand on the EO-DT.

An important enhancement in our EO-DT was data fusion using classical (**Section 2.4**) and ML (**Section 3.2**) combination techniques. Data fusion encompasses several transformations, such as spatio-temporal gridding, gap filling, quality control, and trend analysis. In our demonstration, we employed a classical regridding scheme to validate our cloud architecture and tested ML methods offline. In the future, more complex fusion schemes can be employed using the same system and on-demand file format conversions. Our data fusion had a processing time requirement, so that it was within 10 minutes of a global day. Typical regridding processing times for each granule were typically around 30-60 seconds using a single core but can be scaled using multithreading and container orchestration to ensure the time requirements are met for a complete global day. We installed several Python scripts to monitor the data fusion processing time and displayed them on the dashboard (**Figure 1.13**). Some essential statistics are CPU usage, memory usage (MB), and processing time (minutes). By monitoring the data fusion performance, system administrators can decide how to allocate cloud resources best to achieve user requirements and accurately estimate their cost.

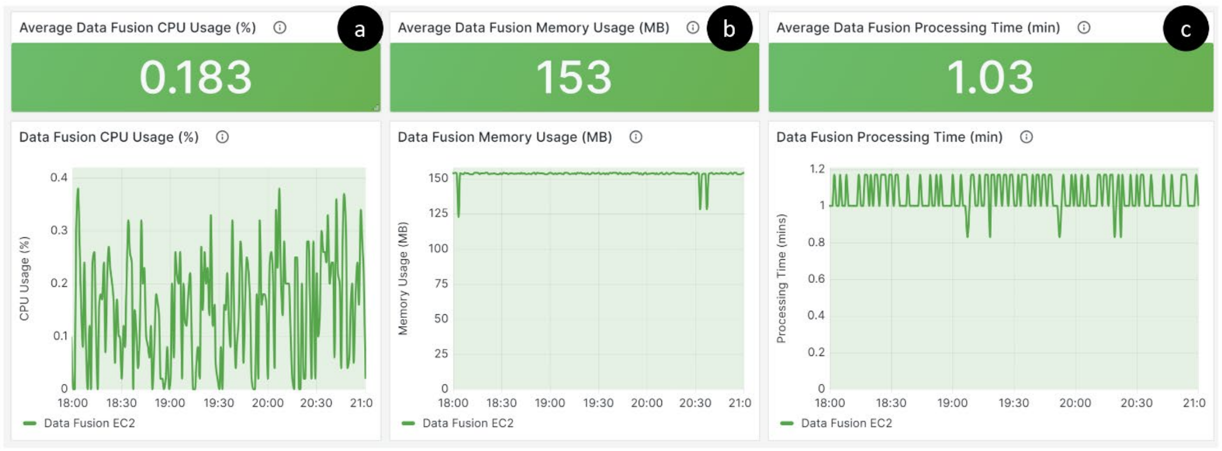


Figure 1.13 The dashboard output of cloud computing resource usage to perform data fusion on 09/27/2023 18-21 UTC. (a) shows the mean CPU percentage for each user request, (b) shows the total memory usage (MB), and (c) shows the processing time (mins). Monitoring resources in real-time allows management to perform a cost-benefit analysis of using on-demand services or pre-processing the data.

1.2.2.3 Catalog and Archive

After data products are generated, they are cataloged and archived on NOAA data archives, such as those maintained by NCEI. Nearly all operational Level 1 and Level 2 products are archived on CLASS. Some Level 3+ datasets are archived on CLASS but generally must have strong stakeholder justification. Users who access the data via CLASS can search it because data are cataloged in the archive based on their metadata. Metadata parameters enable fast searching, so the system does not have to open each file and examine its contents. This is especially important in determining the location of a granule.

In our EO-DT, we developed a catalog system that did not require opening individual files, thus eliminating readers all together. For NESDIS LEO datasets, file names commonly contain information such as the satellite, sensor, product, and start and end times. However, associating location information often requires the system to open and find the geolocation data elements, which adds cost and complexity to maintain because each data product's content structures are unique and need a custom reader to parse. Instead of determining geolocation from within the file, we used the two-line element (TLE) and the file start and end times to determine the approximate location. Instead of using latitude and longitude coordinates, we used a geohash to identify the location, which is faster to query in a database (**Section 2.3.4**).

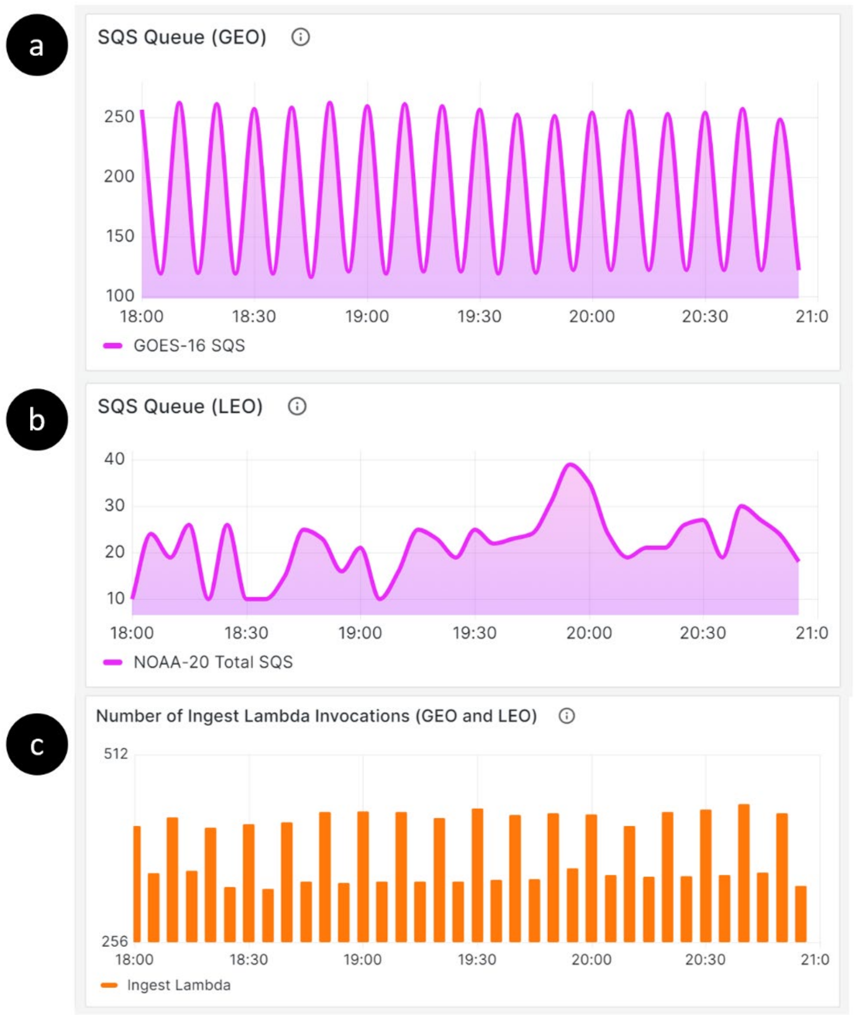
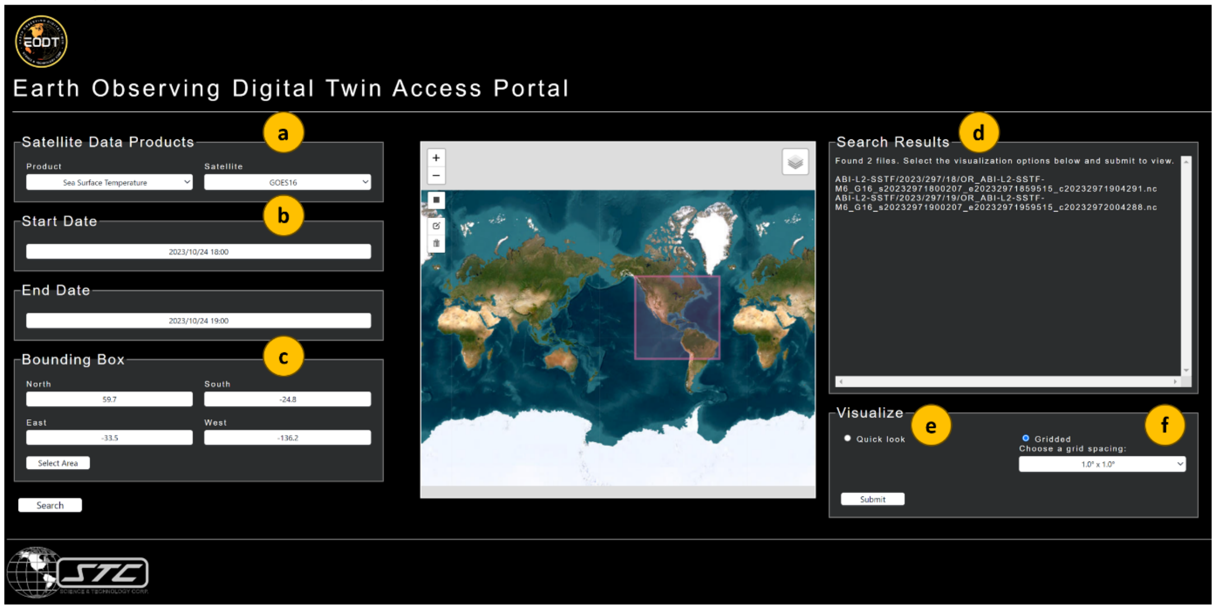


Figure 1.14 The dashboard output of the catalog and archive on 09/27/2023 18-21 UTC. The top two panels are a timeseries of number of new (a) GOES-derived dataset and (b) LEO datasets that appear on the NODD. (c) shows a time series of the new data being cataloged and a count of on-demand computing resources that are invoked.

While not part of the demonstration, we studied some emerging data formats and compared them with netCDF4 which is one of the most prevalent formats at NOAA (**Section 4.2**). Self-describing data formats like netCDF4 offer excellent compression and are considered a standard in the Earth Sciences. While excellent for archiving, netCDF4 files are not cloud-optimized, unlike Zarr, an emerging format that is cloud-optimized. Cloud-optimized formats enable users to access parts of the data file without downloading the full file. Other popular cloud-optimized formats include Avro, Parquet, and ORC, but those are more suitable for time or other point-based data and less suitable for geospatial data. While netCD4 is not optimized for the cloud, tools are emerging that will allow users to load only parts of the netCDF file with a slight performance penalty. Since converting the entire archive to a new format can be prohibitive, we recommend integrating processing tools with the current netCDF4-based archive.

1.2.2.4 Search and Dissemination

Presently, users query NESDIS data by visiting www.class.noaa.gov. Users can search by time and geographic location for data on a geospatial map. A list of available files is returned, and the user places their order. Once the data are processed, which can take anywhere from a few minutes to days, an FTP link is sent to the end user for them to download. In the EO-DT, we developed a web portal (**Figure 1.15**) that allows users to query select data products on GOES and JPSS NODD archives, which is available as drop down menu items in (**a**). Like CLASS, users can subset their data by (**b**) time and (**c**) location and the interface returns a list of matching products (**d**). Since the data are already staged on the cloud, users are not delayed downloading the files. We added two additional features not available on CLASS: the ability to visualize the data in its native grid as a 'quick look' or (**e**) to regrid the data to a regular grid (**f**).



Earth Observing Digital Twin Access Portal

Satellite Data Products
Product: Sea Surface Temperature
Satellite: GOES16

Start Date: 2023/10/24 18:00

End Date: 2023/10/24 19:00

Bounding Box
North: 59.7
South: -24.8
East: -33.5
West: -136.2

Search Results

```
Found 2 files. Select the visualization options below and submit to view.

ABI-L2-SSTF/2023/297/18/OR_ABI-L2-SSTF-
M6_G16_s20232971800207_e20232971859515_c20232971904291.nc
ABI-L2-SSTF/2023/297/19/OR_ABI-L2-SSTF-
M6_G16_s20232971900207_e20232971959515_c20232972004288.nc
```

Visualize
Quick look [✓ Gridded]
Choose a grid spacing: 1.0° x 1.0°

Figure 1.15 Web portal that has the user interface to search the NODD. Users can query the archive by (a) data product and satellite, (b) date and time, (c) geolocation bounding box. After clicking "search" in the lower left, the portal returns a list of filenames (d) that match the query. The user can then visualize the data in its (e) native grid using "quick look" or (f) regrid the data to a regular grid of their choice.

After the user hits submit, a new tab opens in their browser, and the data are overlaid on a map. **Figure 1.16** compares SST in the quick look and the regridded (1 km) view. For the prototype, the quick look displays only the first file in the list, but it could be improved to allow the user to choose a file of interest. The regridded data combines both files, and a future feature could allow users to select their desired spatiotemporal averaging. The data are irregularly spaced in the quick view shown in **Figure 1.16** because they occur on the ABI fixed grid. The regridded visualization places everything on a common, evenly spaced 1.0˚x1.0˚ grid.

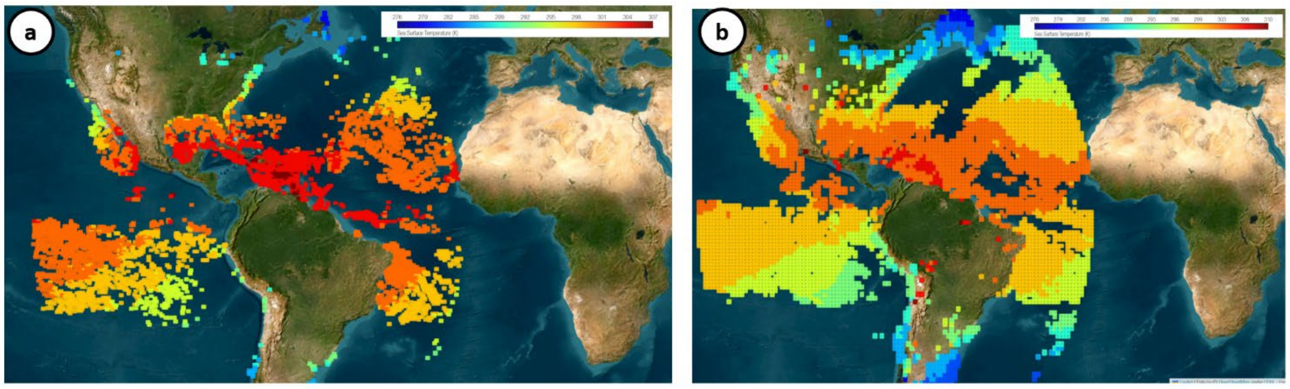


Figure 1.16 A comparison of a visualization user interface of the GOES-16 Sea Surface Temperature product of a single granule in (a) single-file quick look and (b) regridded two-hour average on Oct 24, 2023 18-19 UTC. Inland pixels are due to lakes or large rivers.

Like the other ground system components, the dashboard monitors real-time user interaction with the portal. Recall that user requests are all performed on-demand, so **Figure 1.17a** shows the number of on-demand invocations of the system when the user searched the catalog (orange) and requested a visualization (green). Each time the user clicks the search button, the catalog count will increase, and each time the user selects submit, the map count will increase. In addition to computing resources, we can see what products interest the users and where. **b** shows a heatmap of regions that users searched, where the redder the region, the more interest. **c** shows which data products were searched and how many times. Finally, **d** shows the regridding resolution that was requested.

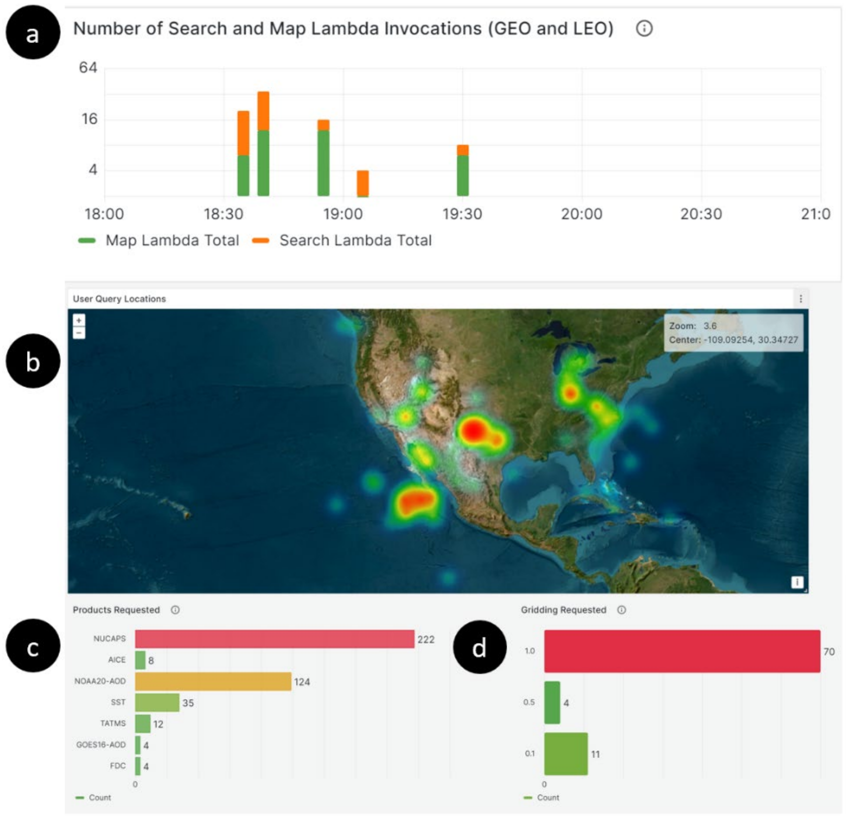


Figure 1.17 The dashboard showing analytics of how users interacted with the user interface. (a) shows the number of on-demand invocations of the system when the user searched the catalog (green) and requested a visualization (orange). (b) shows a heatmap of regions that users searched (c) shows the most popular data products and (d) shows what regridding scheme was requested,

1.3 Summary

In this section, we provided a high-level overview of the EO-DT and how we designed our system to achieve our goals, which were to (i) use real-time data to model current capabilities from satellite archives, computing resources, and user interaction, (ii) explore where new services and ML can enhance the user experience and limit data wrangling; and (iii) support decision making related to optimizing NESDIS data processing. We performed a live demonstration for NESDIS and stakeholders, and this report provides specific project details, technologies used, and lessons learned through the process. The remainder of the report contains a detailed description of our work so that others can use our approach and benefit from the lessons we learned about building a prototype digital twin.

1.4 References

Boukabara, S.-A., Krasnopolsky, V., Stewart, J. Q., Maddy, E. S., Shahroudi, N., & Hoffman, R. N. (2019). Leveraging Modern Artificial Intelligence for Remote Sensing and NWP: Benefits and Challenges. Bulletin of the American Meteorological Society, 100(12), ES473–ES491. https://doi.org/10.1175/BAMS-D-18-0324.1

IBM. (n.d.). What is a digital twin?" Retrieved September 27, 2023, from https://www.ibm.com/topics/what-is-a-digital-twin

Lavin, A., & Renard, G. (2020, December 16). Technology Readiness Levels for AI & ML. arXiv. Retrieved from http://arxiv.org/abs/2006.12497

Maddy, E. S., & Boukabara, S. A. (2021). MIIDAPS-AI: An Explainable Machine-Learning Algorithm for Infrared and Microwave Remote Sensing and Data Assimilation Preprocessing - Application to LEO and GEO Sensors. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 8566–8576. https://doi.org/10.1109/JSTARS.2021.3104389

2 Concept of Operations and Computing Infrastructure

2.1 Digital Twin Architecture

Speed, safety, and scalability contribute to the rise in popularity of cloud technologies (Arora, 2019). Our EO-DT architecture supports a user interface that simulates existing capabilities on NOAA CLASS and explores new features. The user interface was designed to support a limited number of simultaneous users but can leverage the underlying cloud platform for a fully operational EO-DT[3]. Our multi-tier (front, middle, and server-side) architecture leverages AWS serverless resources where possible and is designed to be flexible, maintainable, and scalable.

We evaluated a variety of AWS services during our exploration and determined the following set to be ideally suited for implementing the EO-DT. EC2, or Elastic Compute Cloud, allows users to run virtual servers in the cloud, offering scalable computing capacity. Lambda is a serverless function that runs code that responds to events and automatically scales based on demand. DynamoDB is a managed NoSQL database service that provides fast and predictable performance with seamless scalability. S3, or Simple Storage Service, provides object storage through a web interface and is designed for online backup and archiving of data and applications. API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. CloudFront is a content delivery network (CDN) that distributes content globally with low latency and high transfer speeds. Simple Notification Service (SNS) distributes notifications to subscribers using a publish-subscribe model. Simple Queue Service (SQS) provides a message queue service to store SNS messages awaiting processing by other resources. Lastly, CloudWatch collects and tracks metrics, set alarms, check logs, and explore changes to cloud resources.

Figure 2.1 shows a diagram of the cloud architecture of our prototype EO-DT. We used CloudFront for content delivery and cashing, EC2 instances for computing, S3 for storage, DynamoDB as a database, Lambda for serverless functions, API gateway for networking, and SNS/SQS for messaging and notification. We also used a managed Grafana Cloud dashboard to view system-generated metrics. Below, we describe the computing processes and the data flows.

Two core processes are persistent. First, a process that include cataloging new data from the NOAA NODD into our DynamoDB to create a searchable database of available data for end users. In this process, new data triggers a Lambda function that creates and writes metadata to the DynamoDB. Second, a process that monitors new Level 1 ABI data for anomalies as the NODD receives new data. These processes are monitored using CloudWatch, and the anomaly detection output is written to an S3 for display in the Grafana dashboard.

[3] There is no planning for activity beyond the demonstration projects. See Disclaimer on page 4.

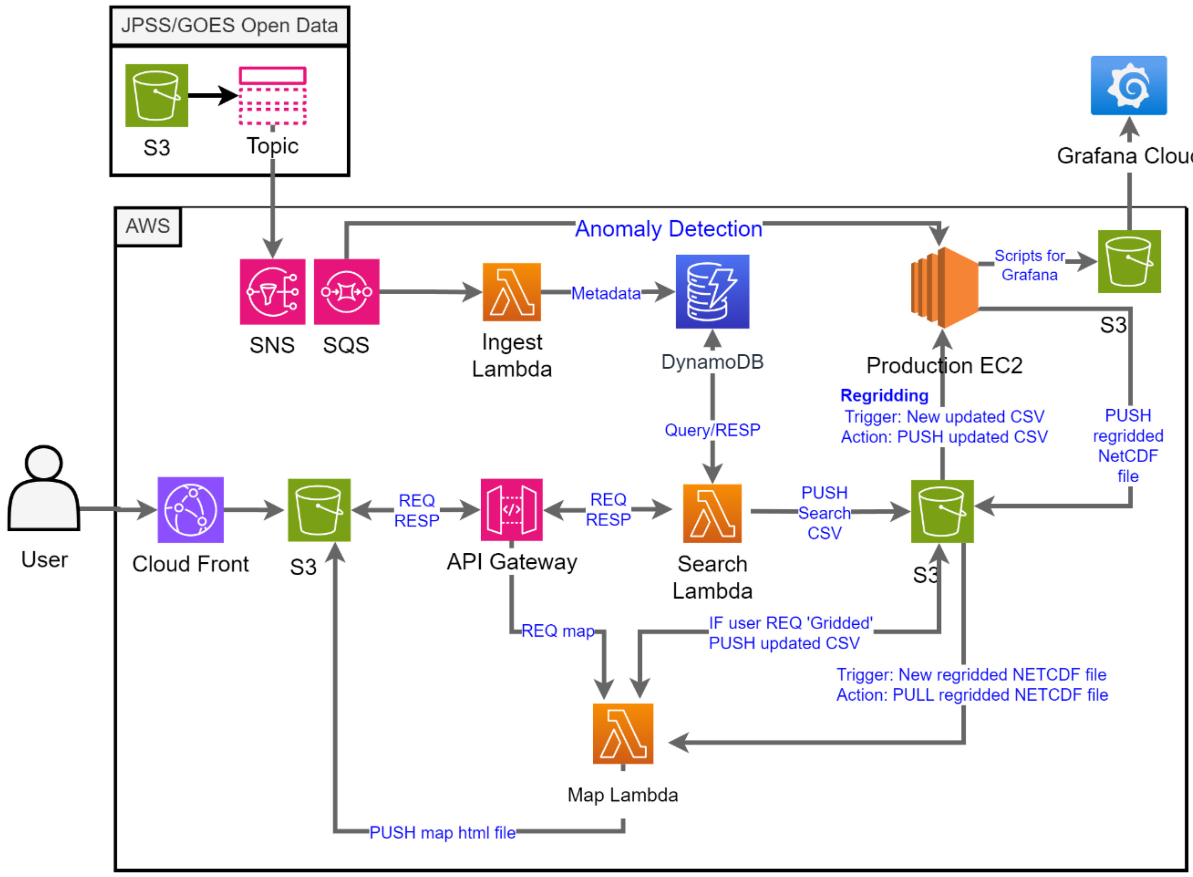


Figure 2.1 Cloud architecture of our prototype EO-DT. There are three main data flows: (1) the metadata from the NODD (JPSS/GOES Open Data) that is stored in a database (DynamoDB); (2) the user request data, which passes into the API gateway, triggering a database search and which returns matching results. The user then can request a visualization type and (3) the resulting output file, which the user can visualize as a map. If the user selects 'data fusion,' the data are processed on an EC2 before being saved to an S3 bucket; if the user selects quick look, the data is imported directly from the NODD. A lambda imports the data and creates an HTML map, which is delivered as a new tab in the user browser using CloudFront and public S3.

In parallel, the system has several on-demand processes: user data queries, data fusion, and visualization. The user can access the user interface via a static website delivered by CloudFront and a public S3 bucket. Their search request is delivered using the API gateway, which triggers a Lambda to search the DynamoDB and return the files that match the user's search parameters. Users can select "quick look" or "gridded" as their visualization option. Performing a "quick look" visualizes a file in its native, irregular grid. Their request is processed through the API gateway, which triggers another Lambda that opens the user's requested file, extracts a data variable, and saves it to a map in a vectorized format. If the user selects "gridded," the request is again processed through the API gateway, which is now routed through an EC2 to perform a grid conversion from the data's irregular grid to a regular grid of the user's choosing. The regridded file is saved to an S3, triggering the map Lambda, which extracts a data variable and saves it to a map in a vectorized format.

The system has three core data flows: the catalog data, the user request, and the output file. Metadata from the NODD is parsed by Lambda and stored in the DynamoDB to create the catalog data, which the user can search. The user request data consists of data search parameters and a map visualization option. Their search request passes into the API gateway, triggers Lambda to search the database, and returns the matching files. The user performs a second request to select a visualization type. Their second request determines how the data will flow through the system. If the user selects 'data fusion,' the data are processed on an EC2 before being saved to an S3 bucket. If the user selects 'quick look,' the data is imported directly from the NODD. A Lambda retrieves the data and creates an HTML map, delivered in a new tab in the users' browser using CloudFront and public S3.

There are also two smaller data flows, the first being the time series of the anomaly detection system and the second is the system analytics data. Both data flows are routed into Grafana. The anomaly detection system creates a JSON file with a time series of every GOES-18 L1b channel and whether the image contains an anomaly ('invalid') or not ('valid). A second data flow involves system analytics, which consists of several smaller data streams. We utilize CloudWatch to fully monitor the health of the cloud resources and access logs to check for errors. We also developed several custom Python scripts to monitor user request statistics. Grafana is a one-stop interface for NESDIS management users to access data quality, system performance, and user engagement with the EO-DT.

Where possible, we leveraged cloud scalability in our design. Managed services, like Lambda and DynamoDB, automatically scale and can handle thousands of simultaneous calls. S3 storage is essentially unlimited and expands as the data holdings grow. The EC2 production environment is not scalable in our prototype design. However, with containerization and clustering, the resources could auto-scale to accommodate more simultaneous users.

During the project, we also evaluated several services that we did not ultimately decide to deploy in the EO-DT. For instance, we explored using TwinMaker, Amazon's new digital twin service that provides a framework to integrate data streams from IoT sensors. An appealing characteristic of TwinMaker was that it fully managed the messaging and data flow within the system, and we could expand to accommodate new sensors and datasets and remove components if features were retired. However, upon testing, TwinMaker did not easily ingest NESDIS data sources, which included satellite sensor data, retrieval products, dataset production data, and user inputs. Instead, we found that a combination of SNS/SQS and Lambda functions carried out many of the same functions as TwinMaker and were able to leverage Python packages that can read geospatial data formats.

We also considered using containers and resources such as AWS Fargate, a serverless compute engine for containers that work with Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Fargate would have allowed us to deploy containerized applications without managing the EC2 instances. We ultimately decided using EC2 with Python virtual environments was sufficient for the demonstration. However, if one is built, we recommend Fargate or another container orchestration approach for an operational EO-DT.

Overall, flexibility, maintainability, and scalability are the key strengths of our architecture. Our system was tested when the JPSS part of the NODD underwent reorganization, and our data paths no longer pointed to the files needed to populate the catalog - as seen in reduced data flow into our catalog database. Fortunately, our SNS filters are easy to update. The system remained online through this episode, albeit with a short data outage for some products. Some future improvements of the EO-DT would include incorporating containerization, adding security measures, and load balancing so the system can handle more simultaneous users.

2.2 Data Products

For our study, we used twelve data products from NESDIS' portfolio (**Table 2.1**) representing the five earth domains: ocean, atmosphere, land, cryosphere, and space weather. These products have all undergone maturity review, are deemed mission critical, and have an active user base. These datasets are derived from a blend of geostationary and polar orbiting sensors and span processing levels one to three. These datasets are also helpful for monitoring long-term trends and represent diverse spatial resolutions and latencies. All these datasets are also available on AWS S3 buckets as part of the Amazon Sustainability Data Initiative and the NOAA Open Data Dissemination Program, which the prototype system can fetch. There are other data pathways, such as using NOAA CLASS subscriptions. We chose to use the Cloud because it represents the future state of NOAA data holdings as part of the NESDIS Common Cloud Framework (NCCF).

Table 2.1 Datasets used in the EO-DT prototype.

Product Name	Platform	Level	Resolution	Latency	Earth System
VIIRS SDR	NOAA-20	1	15 km	90 min	All
ATMS TDR	NOAA-20	1	15 km	90 min	All
CrIS SDR	NOAA-20	1	15 km	90 min	All
ABI	GOES-16	1	0.5-2 km	5 min	All
Sea Surface Temperature	GOES-16	2	2 km	1 hour	Ocean
Sea Ice Concentration and Temperature	NOAA-20	2	15 km	90 min	Cryosphere
Active Fire	GOES-16	2	2 km	90 min	Land
Aerosol Optical Depth (AOD)	NOAA-20	2	2 km	103 min	Atmosphere
NUCAPS temperature profiles	NOAA-20	2	50 km	90 min	Atmosphere
SUVI L1b Data Products	GOES-16	1	2.5 arcsec	5 min	Space Weather
GFS	-	-	27 km	6 hours	Atmosphere

We incorporated subsets of the above data in different parts of our prototype EO-DT. We chose to display Sea Surface Temperature, Sea Ice Concentration and Temperature, Active Fire, Aerosol Optical Depth (AOD), and NUCAPS in the user interface. We used Level 1 imagery, such as VIIRS and ABI, in our anomaly detection system. The ATMS TDR and GFS were ingested in the MIIDAPS-AI algorithm. Even if not explicitly used in a demonstration, all the above data were included in our catalog.

Working with different data is challenging because they each require unique readers. The DEEVA team at STAR collaborated with us and provided several data readers. Their code was beneficial for reading ATMS and CrIS, which are challenging due to separate data and geolocation files. We used Python packages such as xarray to simplify opening, reading, and working with Level 2+ data. The GOES products are operationally produced on a fixed grid, which must be converted to latitude and longitude coordinates before regridding and mapping. We internally generated several geolocation files that we reused when working with the GOES-16 data so that we did not repeatedly have to perform the conversion, thus saving computing time.

2.3 Serverless Resources

2.3.1 Cataloging

Data users need to be able to query data holdings quickly. The GOES and JPSS S3 buckets are organized by product time but lack native search capabilities. This limitation implies that users create their own tools to search the archive. To address this, we created a metadata-based catalog that allows users to query the cloud archive.

One widely recognized method of cataloging is the Spatio-temporal Asset Catalog (STAC). STAC is a standardized model designed primarily for geospatially focused catalogs and is compatible with databases like MongoDB, and DynamoDB, as they all support JSON formatted catalog entries.

We took a pure DynamoDB approach in our prototype for several reasons. First, DynamoDB is a managed service and easy to deploy. The setup process did not require expertise in database languages like SQL. Additionally, we did not need complex querying in our demonstration. Thus, DynamoDB's straight forward and efficient native querying capabilities met our needs. DynamoDB has not been as extensively examined for remote sensing and geo data applications, and we wanted to experiment with this new approach. If needed, DynamoDB could be replaced with a relational database in an operational EO-DT (if built). While not strictly a STAC catalog, the DynamoDB entries are structured similarly to STAC. For example, users can efficiently query and retrieve geospatial data based on specific parameters such as time, geolocation, product, asset location, and other relevant metadata.

In DynamoDB, each row represents a single file on the NODD. **Table 2.2** shows some example entries in our DynamoDB. For simplicity, we show only five fields: `Obkey`, `satellite`, `geo5`, `product`, and `starttime`. The DynamoDB is indexed based on a partition key and a sort key. The DynamoDB needs to have a unique partition and sort keys for the search to work effectively. However, most of our fields are not unique. The satellite field is not unique because many products are generated from sensors on NOAA-20 and GOES-16. The same limitations are true for the products field. Start times may be shared for multiple products from the same satellite. We will discuss geohash (`geo5`) in **Section 2.3.4**, but like the latitude and longitude coordinates it represents, geohash is not unique. The only unique field is `obkey`, the address on the S3 bucket for the product file.

Table 2.2 Example entries from our DynamoDB. The asterisk (*) indicates the partition key.

obkey*	satellite	geo5	product	starttime
NOAA20/SOUNDINGS/NOAA20_NUCAPS-EDR/2023/07/24/NUCAPS-EDR_v3r0_j01_s202307240130559_e202307240131257_c202307240338040.nc	NOAA20	z8zep	NUCAPS	2023-07-24T01:30:00
NOAA20/SOUNDINGS/NOAA20_NUCAPS-EDR/2023/08/31/NUCAPS-EDR_v3r0_j01_s202308311848239_e202308311848537_c202308312000210.nc	NOAA20	qnyh6	NUCAPS	2023-08-31T18:48:00
NOAA20/VIIRS/NOAA20_VIIRS_Aerosol_Optical_Depth_EDR/2023/09/19/JRR-	NOAA20	7s0ns	AOD	2023-09-19T15:09:00

AOD_v3r2_j01_s202309191509277_e202309191510523_c202309191601326.nc				

The fastest searches will query based on the partition and sort keys; querying the other fields is significantly slower. We needed users to be able to search the table based on multiple parameters: time, location, and product. A solution is to create Global Secondary Indexes (GSI), which copies the main table with different partition combinations and sort keys (**Figure 2.2**; AWS. (n.d.)). GSI makes a copy of the main table, once for each different sort key. In our example table above, we would have one main table (`obkey` x `product`) and three GSI tables, `obkey` x `satellite`, `obkey` x `geo5`, and `obkey` x `starttime`. The results are then combined using an inner join on the `obkey`, so only the data with the same `obkey` in all four tables is returned.

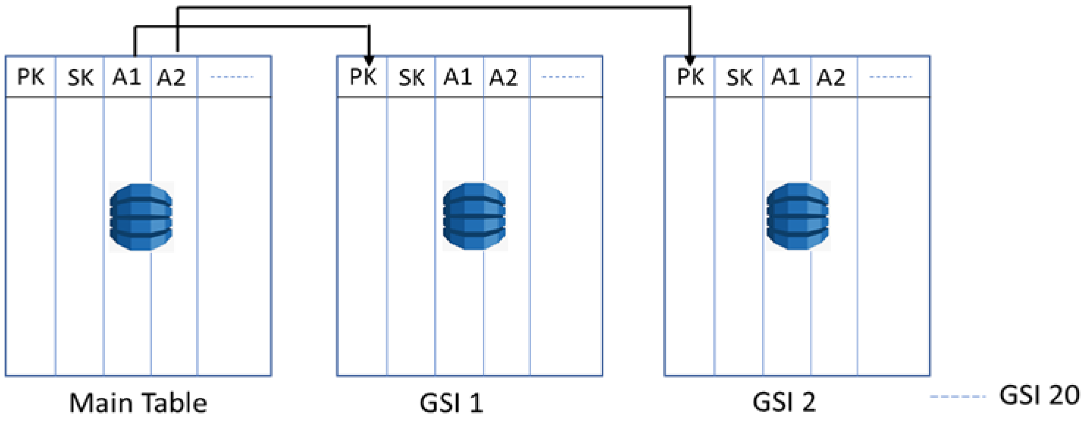


Figure 2.2 Illustration of GSI technique. The GSI duplicates the main table with different partition key and sort key combinations so that the system can perform multifaceted queries.

There are some drawbacks to this approach. Copying the table means your costs will increase for each GSI. Secondly, since each table is searched, the number of tables read capacity units also increases because you are now reading from multiple tables instead of one.

We met with the AWS NOAA account team and validated our approach. Toward the end of the project, we showed our approach to a DynamoDB subject matter expert who proposed an alternative, faster, and more cost-effective approach using Z-Order Indexing (Slayton, 2017). This approach would require us to create a new column with a unique value by combining multiple columns' values and setting that field as the sort key (**Table 2.3**). Then, we can use a non-unique field as the partition key.

Table 2.3 Example entries from our DynamoDB using. Here, we created a new sort key, starttime_geo5_product, by combining the values from starttime_geo5_product to ensure the values are unique. The asterisk (*) indicates the partition key.

starttime_geo5_product	satellite	geo5	product*	starttime
2023-07-24T01:30:00_ z8zep_NUCAPS	NOAA20	z8zep	NUCAPS	2023-07-24T01:30:00
2023-08-31T18:48:00_qnyh6_NUCAPS	NOAA20	qnyh6	NUCAPS	2023-08-31T18:48:00
2023-09-19T15:09:00_ 7s0ns _AOD	NOAA20	7s0ns	AOD	2023-09-19T15:09:00

To illustrate, let us suppose a user makes the following query:

Search for NUCAPS, from 2023-07-24 00:00:00 UTC to 2023-07-25 00:00:00 UTC, for a bounding box between 40.0-50.0˚N and 167.0-168.0˚E.

The DynamoDB would then divide the table using the partition and sort keys :

```
Index: timestamp_lat_long

Key condition: Product = “NUCAPS” AND timestamp_lat_long BETWEEN “2023-07-
24T00” AND “2023-07-25T00”
```

The pseudocode above will return a much smaller subset of values in the table. Then, we can further subset the results using the following filters (shown in pseudocode):

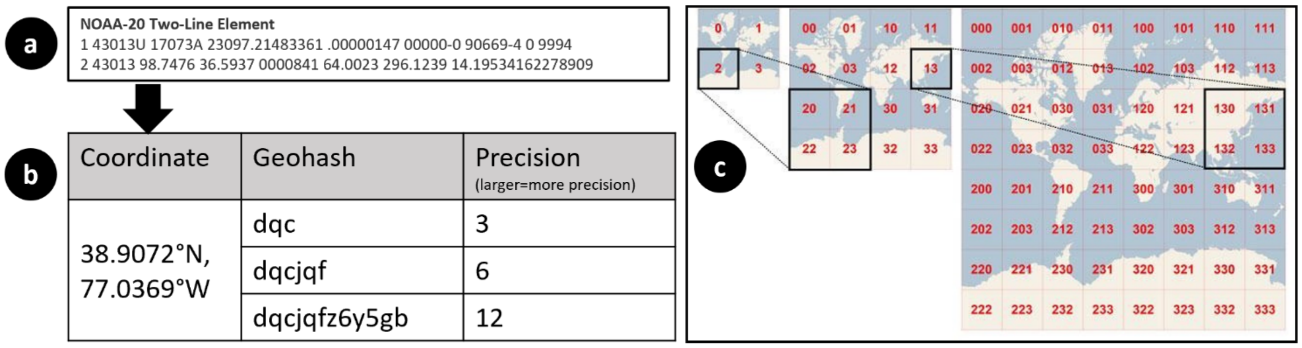
```
latitude BETWEEN 40.0 AND 50.0
AND longitude BETWEEN 167.0 AND 168.0
AND timestamp BETWEEN 2023-07-24T00 AND 2023-07-25T00
```

While we did not implement this approach, the fact that it uses only one table and fewer read capacity units indicates it will provide cost savings over the GSI approach.

2.3.2 Geolocation

As described in **Section 1.2.2.3**, we designed a system that estimates the data location based on the filename. We leveraged a satellite two-line element (TLE) along with the start and end times in the file to approximate a data file's position (**Figure 2.3a**). This approach is agnostic to the data product because it only requires knowledge of the satellite name and the observation capture start and end times. There is no longer a dependency on data-specific readers like HDF libraries or specialized file readers, making the system more flexible for accommodating new data. A drawback is that the file position is not exact but is the nadir point of the center of the granule. To address this, we designed the search to return more results than needed, which can be further filtered in later processing steps once the file has been opened.

While we store the file’s latitude and longitude values in the DynamoDB, we also convert the coordinates to a geohash to improve the search speed (**Figure 2.3b**). Geohash is a spatial data encoding system representing a rectangular geographical area as a concise string of letters and digits (AWS, 2020). Geohash facilitates storage and speeds up the query. Increasing the length of the geohash string can improve the precision of the area it represents. What makes geohashes particularly useful is their hierarchical nature: similar geospatial areas have geohash strings that share common prefixes (**Figure 2.3c**). For example, the geohash "dqc" might represent a large area near Washington D.C. By extending it to "dqcjqf," the area becomes more precise, pinpointing a specific neighborhood or even a street within that city. This hierarchical structure enables efficient spatial queries and proximity searches, as areas near each other often share parts of their geohash strings.



NOAA-20 Two-Line Element
1 43013U 17073A 23097.21483361 .00000147 00000-0 90669-4 0 9994
2 43013 98.7476 36.5937 0000841 64.0023 296.1239 14.19534162278909

Coordinate	Geohash	Precision (larger=more precision)
38.9072°N, 77.0369°W	dqc	3
	dqcjqf	6
	dqcjqfz6y5gb	12

Figure 2.3 The geolocation for new files is cataloged (a) using the satellite two-line element (TLE) to determine the latitude and longitude coordinates based on the file start and end times. The coordinates are then converted to (b) a geohash, which is a concise string representation of the location. (c) shows how the decimal precision of the latitude and longitude coordinates are expressed by longer geohashes.

Some popular catalog options like STAC use very precise polygons to describe the exact granules, which can lead to more accurate search results. These two approaches are not mutually exclusive. One can design a system that opens a file, reads the granule information, and stores a polygon in the DynamoDB to follow the STAC specification. The same DynamoDB can use the TLE/geohash approach to quickly subset the data. Then, the system can use a Python script in Lambda to perform a slower but more precise process, like polygon intersection, on a much smaller subset of the data catalog.

2.3.3 Ingest Lambda

Lambda is helpful because it can process data immediately after it is added to S3 (also called an event-driven trigger). In our EO-DT, as soon as a satellite data file is uploaded, the Lambda function parses it, extracts necessary metadata, and stores it in DynamoDB without any manual intervention or the need to run a constantly active server. We call this process the Ingest Lambda to distinguish it from other Lambda functions.

We wrote our ingest script using Python because it is natively supported in Lambda. We used the AWS Serverless Application Model (SAM) to develop and deploy the code. SAM is an open-source tool that streamlines building and deploying serverless applications on AWS, including Lambda functions. Utilizing SAM for Lambda deployment means you can define your function, its event sources, and any necessary AWS resources (like DynamoDB tables) in a single SAM template. When deploying with SAM command line interface (CLI), it packages your code, uploads the package to Amazon S3, and uses CloudFormation to deploy your serverless application. The SAM CLI also offers local testing capabilities, which are used to test our Ingest code in a Docker environment that simulates the AWS runtime.

The main steps in our ingest code are as follows: (1) parse the filename for a product, satellite, start time, end time, and creation time; (2) determine the TLE from the satellite and start time and end time, (3) convert to geohash, (4) add the result to the DynamoDB. In its entirety, the process takes 10s per invocation. Some minor challenges in the setup include inconsistency in filename conventions for different satellite systems. For example, GOES data uses the day of the year to specify the date, while JPSS uses a year-month-day format. Level 1 JPSS data separates the day from the time differently than for JPSS data products. As a result, we had to write multiple filename parsers.

An important consideration is that a satellite's TLE changes over time to reflect satellite orbit changes due to drag. So, the TLE table must be updated once a day to reflect these changes. We learned this lesson quite painfully when our initial testing did not return results in the specified bounding box when we tested it in March.

Lambda deployment involves zipping the entire contents of a working directory and uploading, so we included a static copy of the TLE in the package. The query results were thousands of miles away from our bounding box by September because we were using an outdated TLE. Because we did not want Lambda to download the current TLE file every time it was invoked (which may lead to our IP being blocked by the host website), we wrote a script to download the TLE once a day to our local S3 and Lambda, then imports it each time it is invoked.

2.3.4 Search and Map Lambdas

AWS API Gateways connect a front-end portal with server-side operations, such as our user interface. API Gateway creates RESTful APIs that trigger AWS Lambda functions, enabling client applications to interact with your backend without the servers' direct involvement. We chained API gateway with a Lambda function to allow users to search our DynamoDB. We call this process the search Lambda.

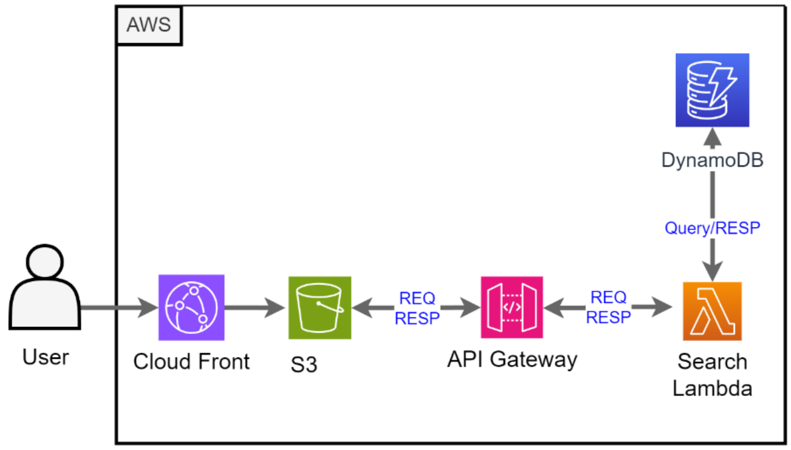


Figure 2.4 The architecture pattern for an API gateway to lambda to DynamoDB.

Search Lambda is triggered by a user request to the API gateway (e.g., when the user clicks the search button in **Figure 1.15**). **Figure 2.4** shows a simplified diagram of this part of the architecture. The user portal is represented by CloudFront and S3 icons. The user's request (REQ) will contain their desired query parameters, such as the product name, dates, and bounding box. The API gateway sends a request to search Lambda, which queries the DynamoDB. The search Lambda handler function ensures the query is valid, performs the GSI search on the DynamoDB, finds the intersection of the results, and returns the matching files to the API gateway (RESP). The files are displayed on the top right-hand side of the website. While not shown in the figure for simplicity, the search results are saved as a CSV file on an S3 bucket. At this stage, the user is given a random search ID number to track their requests through the system.

The third and final lambda generates two maps for the user (map Lambda). This Lambda is invoked through the API gateway when a user requests a map. The user has two options: display the data as a quick look (**Figure 2.5a**) or regrid their results and visualize them (**Figure 2.5b**). Examples of each are shown in **Figure 1.16**.

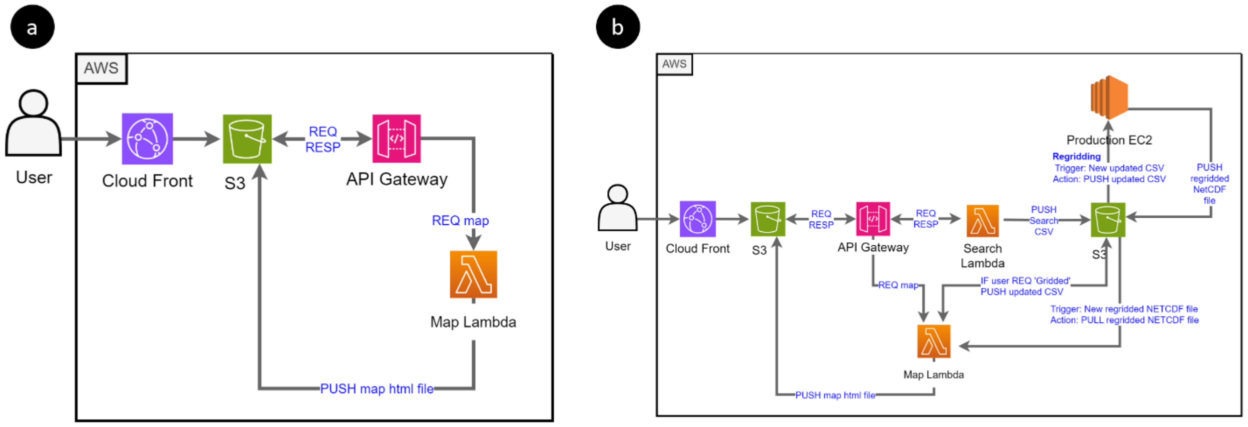


Figure 2.5 Shows two patterns, one for (a) the native grid “quick look” map display and another for (b) a regridded map which requires additional processing on an EC2.

The quick look architecture is relatively simple. The API gateway sends the user request for a “quick look” along with the search ID number, thereby invoking the map Lambda. The Map Lambda opens the corresponding search ID CSV file stored on an S3 bucket and reads the first filename. Our approach was simplistic in the demo. Note that in an operational EO-DT, we envision the user will be able to select which files they wish to display. It is here that we open the files themselves for the first time. Each product requires a unique reader to parse the file contents. We only extracted one variable for simplicity, even though numerous fields are in the files. In the future, we imagine the end user could use the file contents as a search parameter. Because the quick look is intended to display data rapidly, we downsampled the data at a ratio of 10:1 for some of the larger files, such as those from the full disk ABI. We selected a Python package called Folium to generate a vectorized data map. Folium helps quickly create an HTML file with the data converted to a vectorized format and overlays it on a Leaflet map. Leaflet (https://leafletjs.com) is a popular, lightweight, open-source JavaScript library for interactive maps. The leaflet map is uploaded onto the web portal S3 bucket and automatically loaded as a new tab in the users’ web browser.

Users can also regrid their data to a regular grid. The user has three options for the demonstration: 1.0˚x1.0˚, 0.5˚x0.5˚, and 0.1˚x0.1˚ latitude and longitude. In the future, we envision users entering custom values or using more complex grids. If “gridded” is selected, the same map Lambda is triggered, but it does not make a map yet. Instead, it opens the CSV file, appends the users' grid choice, and copies the file to an S3 directory monitored by our data fusion script for activity. We will describe our data fusion approach in the next section, **Section 2.4**. Early in the project, we considered using Lambda to perform data fusion. However, Lambda has strict size limitations for the entire package (1GB) and strict processing time limits (<15 minutes to run). The data fusion processing time may exceed the time limit for large requests. Instead, we installed the code on an EC2. For a fully operational EO-DT, we recommend creating a docker image and deploying using a service like Amazon Elastic Container Service (ECS) to fully scalable the resources. Once the file is regridded, it is saved to another S3 directory linked to the map Lambda. If a new file is present, the map Lambda triggers. Lambda then reads the file and displays the regridded data on the leaflet map, which is then opened as a new tab on the user’s browser.

We learned several lessons using our approach. The fine print around the AWS resources must be factored in early in the design because they impact the architecture significantly. For example, there is a 6MB size limit on the Lambda response, so large results need to be uploaded as a file and not delivered to other resources as a message. Similarly, there are size and timeout restrictions for using Lambda. For simple processes, Lambda is both efficient and cost-effective. However, more complex code or code with large dependencies must be deployed using a container or a persistent service like EC2. We recommend carefully defining your requirements upfront to fully consider the trade-off between on-demand and persistent resources.

2.4 Classical Methods for Data Fusion

In earth science, spatial regridding is the "process of interpolating from one grid to another" (NCAR, 2014). More broadly, the concept of combining multiple datasets is referred to as data fusion, which can be defined as "a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates." (White, 1991; Steinberg et al., 1999).

We envision that our data fusion system will enable the user to specify the required datasets, and then EO-DT will find the data and combine them on a common space/time grid for analysis and display. If there is sufficient end user justification, NESDIS generates analysis-ready data via a Level 3 product, where data are typically merged from multiple satellites and stored on a common, uniform grid. Level 3 data is essentially “pre-processed” for the end user to save them time. While cloud storage is relatively inexpensive, for less popular datasets, on-demand processing charges may be cheaper than maintaining a full data record. The resulting custom datasets may serve a greater variety of end users, whereas a static Level 3 product may only serve a few power users.

This section describes our on-demand algorithm for spatial mismatch and gap filling using classical techniques in the prototype EO-DT. We use the term classic to indicate data fusion techniques that do not involve ML. We also explored using ML for gap filling and other challenges, such as leveraging data redundancy from multiple satellite instruments, in **Section 3.2**.

2.4.1 Spatial Regridding

One of the primary challenges of combining satellite datasets is addressing the mismatch of observations in time and space. We show some examples of the different grids and resolution of NESDIS in **Table 2.4**. Even if from the same sensor, datasets can have varying resolutions and grids. While humans may be able to visually combine mismatched observations, software, models, and analysis tools require the datasets to be on the same grid. Regridding to a common grid makes the data more analysis-friendly, but at the expense of information loss or distortion. Fine features in the datasets are smoothed over or lost entirely if the projected grid spacing is too large. The data can appear noisy if the projected grid is too small. Regridding can be inherently slow, especially when searching over many pixels for nearest neighbors. Selecting an appropriate solution is complicated because satellite data footprints often vary depending on global position and viewing angle, leading to irregular, non-uniform-spaced horizontal grids (Shea, 2014). GOES data is a regular grid in satellite viewing angle space (called the fixed grid; NOAA, 2019) but becomes irregular when projected onto latitude-longitude coordinates. Irregularly gridded satellite data is often combined with model data on regular, uniform grids. For example, satellite observations are often combined or assimilated onto the GFS model's regularly spaced grid. There are many other possible grids in the Earth sciences (e.g., Equal area, curvilinear, hexagonal, polar, and meshes), but most spatial data is either in an irregular or regular latitude-longitude grid.

Table 2.4 shows the differences in resolution across NESDIS earth observations data. Level 1 and Level 2 products have different resolutions, making it difficult for users to combine.

Product Name	Platform	Resolution (km)	**Grid Type**
NUCAPS	NOAA-20	50	Irregular swath
Aerosol Optical Depth	NOAA-20	.75	Irregular swath
Aerosol Optical Depth	GOES-16	2	Regular viewing angle
Sea Surface Temperature	GOES-16	2	Regular viewing angle
Ice Concentration and Extent	GOES-16	2	Regular viewing angle
Active Fire	GOES-16	2	Regular viewing angle
ATMS TDR	NOAA-20	15	Irregular swath
GFS	-	10	Regular grid

When converting a regular grid to a different regular grid, you are taking pixels with coarser or finer spacing than the target and interpolating them onto the target grid. Since regular grids have consistent spacing, the relationships between grid points are predictable. The ratio between source and target grids is often an integer or simple fraction. For instance, regridding from a 2 km grid to a 1 km grid involves a direct 2:1 relationship, making interpolation straightforward. Python has several fast routines that perform regular regridding, such as the `ndimage` and `interpolate.griddata` functions in SciPy (https://scipy.org).

Converting data from an irregular grid to a regular grid is challenging and requires complex interpolation. Some interpolation methods include inverse distance weighting, kriging, or natural neighbor techniques, which are more computationally intensive than nearest neighbors and linear interpolations. For each grid cell in the regular grid, the algorithm must search for the nearest data points in the irregular dataset. When irregular data is scattered sparsely across the globe, there are a lot of different approaches. For example, the data can be binned to a coarse grid and then interpolated to the target grid. The interpolation method can be bilinear, spline, or nearest neighbor using piece-wise interpolation. Alternatively, the input data can be distance-weighted to the target grid points. A drawback is that this type of interpolation can be computationally intensive and slow.

If the irregular data is dense, data binning routines are appropriate and relatively fast. Data is averaged in each regular grid data bin. Some data binning routines will allow for weights, weighing the data in a bin based on how close it is to the grid point. Our methodology in the EO-DT assumes the measurements are dense, although, in practice, this is incorrect because there are missing values due to clouds, orbit gaps, sun glint, and other retrieval errors. We will address this in **Section 2.4.2**.

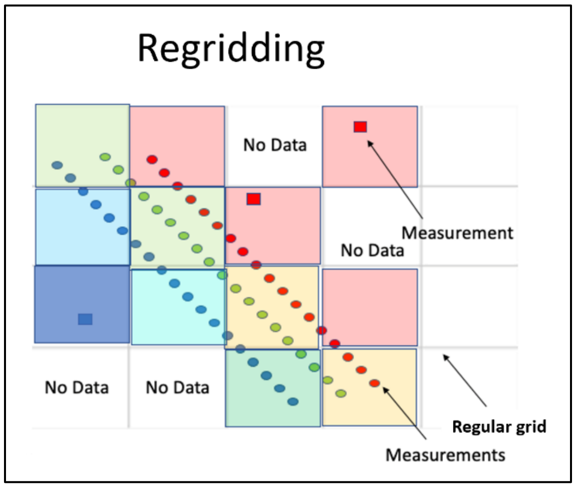


Figure 2.6 Shows how average data regridding works with the small dots representing the satellite measurements, small squares in situ measurements. The larger squares are the values mapped to the regular grid.

Some of our goals were to write an algorithm that was (1) dataset and grid agnostic and (2) met a processing time requirement of < 10 minutes to run a global day. The latter is highly dependent on the instance type running the data, so while we optimized the code, this was not a particularly restrictive environment.

Our regridding algorithm gathers all the data within each grid box centered on the grid point (**Figure 2.6**) and assigns the average value to the grid point. This average data regridding works best if many data points are in the fixed grid boxes. Statistics can be generated on the data inside the bin, which is useful in evaluating the representativeness of the gridded data. The data in the box can also be weighted so that some measurements contribute more to the average - this can be important if one data set adds a significantly more amount of data to the bin than another (e.g., aircraft versus satellite) or if one set of data is considered more accurate than another.

Our pseudocode for the primary subroutine that grids the data is:

```
1. We initialize our array with a fill value (-999.0)
      $newdata[i,j] = fill\_val$
      for all $i$ in $[0, len(x_{out}) - 1]$ and all $j$ in $[0, len(y_{out}) - 1]$.
2. For $j$ in $[0, len(y_{out}) - 1]$:
      $if\ (y_{out}[j] - dy \leq y_{in} < y_{out}[j] + dy)$ and $(\ data\ \neq fill\_val\ )$.
         for $i$ in $([0, len(x_{out}) - 1])$:
            $if\ x_{out}[i] - dx \leq x_{in} < x_{out}[i] + dx$
               $\mu = \frac{\sum_k data[k]}{cnt}$ where the sum is over valid $(\ k\ )$
               $\mathrm{newdata[i,j]} = \mu$
3. Return newdata.
```

Where x_{in} and y_{in} are the input data coordinates in the native/original grid, $data$ is the array containing the original data values at those coordinates, x_{out} and y_{out} are the coordinates of the regular new grid, $fill\_val$ is the placeholder value to indicate absent or invalid data, dx and dy are the half the distance between consecutive grid points in the x and y dimensions of the regular output grid, respectively, and $newdata$ is the returned array containing the regridded data.

Like all other components in the EO-DT, we wrote the regridding code in Python. We chose to write the regridding subroutine in Cython, a programming language designed to combine Python's ease and readability with C's performance. The Cython syntax is like Python and has built-in support for NumPy, but Cython functions must be compiled before running and integrating into pure Python scripts. The Cython-enhanced code ran significantly faster, reducing computing time from 6 minutes to < 1 minute per granule compared to a pure Python approach.

2.4.2 Gap Filling

Gap filling is the process of filling in missing or corrupted data points within a dataset. Satellite measurements produce data gaps for various reasons, such as instrumental malfunctions, data transmission errors, or environmental factors like cloud cover obstructing the sensor's surface view. These gaps can hinder data analysis, as continuous data coverage is often essential for understanding environmental trends and patterns. Our goal is to create a complete and continuous dataset that maintains the integrity and accuracy of the original observations. Proper gap filling is crucial to ensure that the resultant dataset is representative and does not introduce artificial biases or errors into subsequent analyses.

We developed a use case to evaluate two classical gap filling methodologies feasible in the digital twin. We chose to study wildfire smoke plumes using AOD from ABI and VIIRS for several reasons. First, wildfire smoke plumes are of interest to multiple agencies, including the EPA, USGS, and NASA, and are particularly important given the record fire season in Canada in 2023. Additionally, wildfire smoke presented an opportunity to merge satellite and model data. While our results center on this use case, the gap filling techniques we developed can be extended and adapted to other data products. We did not install a gap filling method in our prototype EO-DT. If installed, our gap filling methodology could be run within the regridding code, and there would be no changes to the cloud architecture in **Figure 2.1**.

AOD is particularly prone to missing values for several reasons. Aerosols reflect sunlight and AOD is measured in the daytime as an enhancement in the visible and NIR solar reflectivity. Clouds obscure the surface and prevent AOD measurements. In addition, sun glint within the field of view will appear as an enhancement in reflectivity and is indistinguishable from AOD or cloud enhancements. Unlike clouds, the position of the glint is known, and GOES L2 AOD product is masked over the glint region. Because of clouds and glint, a large portion of the viewing region is obscured at any moment. However, AOD measurements from geostationary instruments are made every 15 minutes, and clouds and the glint change locations.

We tested two methodologies, backfilling and inpainting, to handle missing pixels, using temporal and spatial data for interpolation. **Figure 2.7** shows an illustration of backfilling. The current image (T=0) has missing pixels. The algorithm searches backward in time to the previous timestep (T=-1), finds a valid pixel, and propagates it forward to fill the missing pixel. A pixel is missing in T=0 and T=-1 but is available in T=-2 and propagated forward. The combined data are shown on the right in the backfilled field.

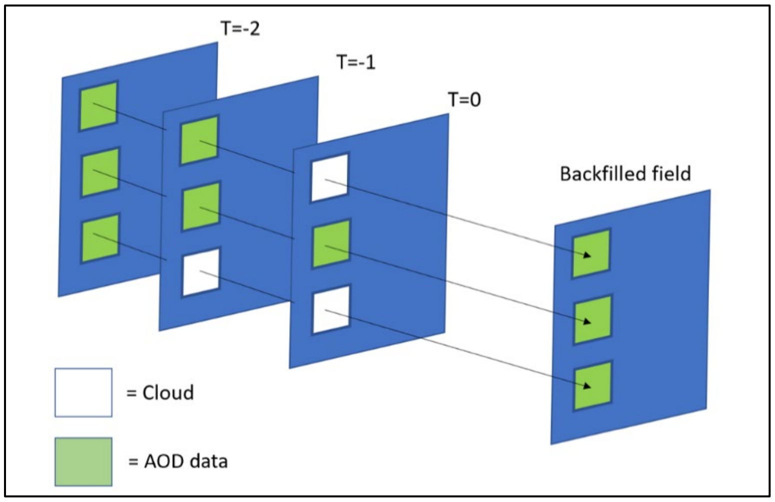


Figure 2.7 Backfilling technique for AOD. Data is brought forward in time to fill in regions obscured by glint and clouds.

In contrast, inpainting fills missing data from adjacent data or other data sets not included in the original set (e.g., model predictions). For example, a single missing pixel or small cluster of pixels could be filled in using neighboring pixels. An advantage of inpainting is that it does not require a long time series of data but it does require accurate quality flags; otherwise, neighboring pixels may be filled with inaccurate data.

Figure 2.8 shows the results from the two gap filling approaches algorithm. Figure 2.8a shows the original scene, and Figure 2.8b shows the backfilled scene with most clouds removed. Backfilling is quite effective at removing clouds because of cloud motions. If we assume that AOD is slowly varying over a few hours (Figure 2.8c), backfilling provides a reasonable approach to generating more complete data fields for an EO-DT user. The inpainting applied (Figure 2.8d) was less successful. The reason is that the GOES AOD algorithm often mistakes cloud edges for high AOD regions. Inpainting then pushes these high AOD regions into the gaps, as seen in the lower left.

When using regridding methods, several caveats must be kept in mind. First, regridding and gap filling create “new” observations derived through interpolation, extrapolation, and averaging techniques. Hence, it is important to understand the data's variability when employing these methods. Second, numerous datasets are accompanied by quality flags that are pivotal in assembling fused data. The challenge is that these quality flags are defined differently across multiple datasets without any standard protocol. For instance, the quality flags for SST are distinct from those for AOD. Finally, cloud masks help filter bad pixels, but not all products explicitly have them. Surface observations are especially vulnerable to cloud mask flags. Sometimes, these cloud masks are integrated into the quality flags, while they exist as separate entities in other datasets. Any data treatment system must be well-informed about the data screening methodologies utilized during the production of Level 2 data. Data fusion is not simply a spatial process but needs to include temporal variations, especially for rapidly varying geophysical variables.

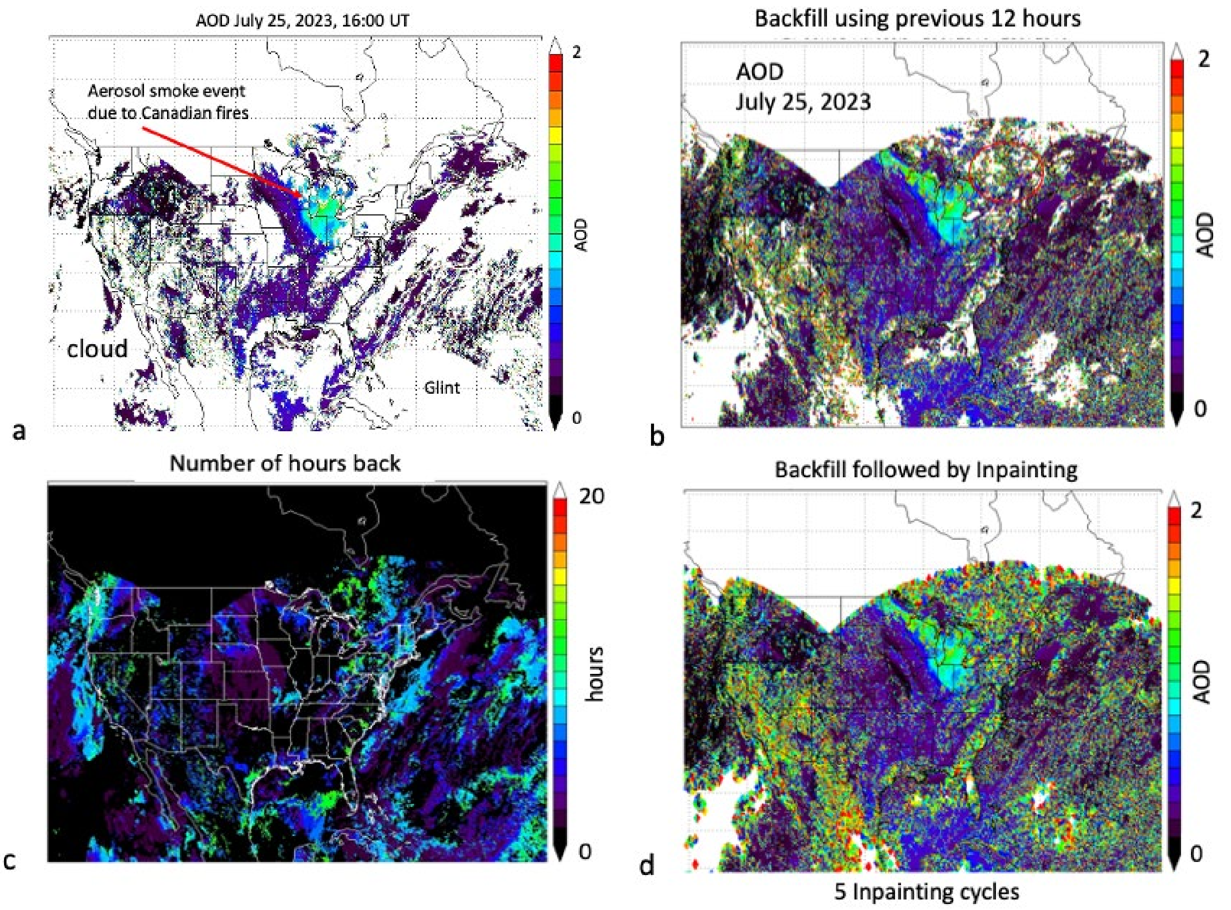

AOD July 25, 2023, 16:00 UT
Aerosol smoke event due to Canadian fires
cloud
Glint
AOD
2
0
a
Backfill using previous 12 hours
AOD
July 25, 2023
AOD
2
0
b
Number of hours back
hours
20
0
c
Backfill followed by Inpainting
AOD
2
0
d
5 Inpainting cycles

Figure 2.8 Illustration of backfilling using combined GOES-16 and NOAA-18 AOD. (a) Shows an example image at 16:00 UT with clouds and glint contaminating most of the field. Using 12-hour backfilling (part b) most of the cloud fields are filled in with AOD data. (c) Shows how far back in time (hours) the algorithm had to go to fill in the cloud field – most of the data are replaced with information only a few hours old. (d) Shows the addition of inpainting.

2.4.3 Summary

We developed a fast, simple regridding method that is agnostic of the data source and can be applied to all NESDIS datasets without re-training. We successfully tested and installed this code into our EO-DT prototype, and it successfully transformed satellite datasets from an irregular grid to a regular grid at a user's requested spacing. The code processed a data granule on one CPU core within 30 seconds to 3 minutes for a 50 km and 750 m VIIRS granule, respectively. Processing speeds can be improved by utilizing multithreading and containerization.

While ML-based regridding schemes are popular in the Earth Sciences due to their ability to learn complex patterns and adapt to varied datasets, classical regridding methods like the one we discussed here offer several advantages. First, classical regridding methods, such as bilinear or nearest neighbor interpolation, are deterministic and have a precise mathematical formulation. They produce consistent results and are generally more accessible for users to understand and interpret. While slower, many classical regridding techniques are less computationally intensive than their ML counterparts, as some ML models require GPU-equipped instances for training.

Our analysis identified several bottlenecks that made data processing and usage difficult. A primary challenge arose from the unpacking and utilization of typical NetCDF4 files, which contain numerous variables helpful to an algorithm developer but may not apply to the average user. Unfortunately, the user must download the entire file to access only a small part of its contents. For instance, we only needed to access four variables (`AOD550`, `QCAll`, `Latitude`, and `Longitude`) out of the 21 geospatial variables in the VIIRS AOD. We recommend adopting a cloud-optimized file system to expedite the processes of opening, downloading, and further processing the files. (**Section 4.2**). While our cataloging system (**Section 2.3.1**) offers a solution for geolocating LEO data products, integrating orbit region and time into the file name could facilitate regional sub-selection for average users.

As notes earlier, the structure and meaning of quality flags vary significantly across NESIDS datasets. To use these flags efficiently, we constructed a table defining the semantics of each quality flag. Harmonizing the meaning and structure of the quality flags in a digital twin is important to broaden the use of NESDIS' high-quality data products.

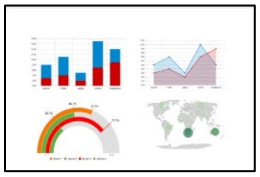
2.5 Data Analytics Dashboards

A digital twin can quickly produce a lot of information, thus requiring tools to organize the various data, metrics, and visualization concisely. We wanted to provide a one-stop location for managers to examine all the data that the EO-DT produces. Data analytics dashboards have become an increasingly popular way to organize complex data to produce actionable management insights (Pauwels, 2009; Microsoft (n.d.)). Static graphics and reports were traditionally the primary way of communicating information to management. While useful, the increasingly data-driven nature of society means stakeholders need information at smaller timescales – by the week, day, minute, and, at times, even on second timescales. These new timeliness requirements mean communicating analytics must be performed using real-time software tools. Furthermore, decision-makers may want to interact with the data, which is impossible using a static medium. Thus, there has been a proliferation of "at a glance" tools everywhere, from our personal banking accounts to technology giants like Google, Amazon, and Microsoft.

In digital twins, dashboards can show the various components of the system in real-time. In a manufacturing environment, a digital twin dashboard can display all levels in a factory, from the shop floor to management (Lin and Low, 2021). A dashboard can quickly identify errors or bottlenecks in the system and enable a fast fix to address the problem. The main challenges include collecting and combining data sources and developing user-friendly visualizations.

In creating a dashboard, we could have developed one entirely from the ground up, but we found that many open-source platforms provide a clean interface and seamlessly integrate with cloud computing resources. The costs are generally free for under five users but can accommodate more users for under a few hundred dollars a month. We evaluated several open-source services, which are shown in **Table 2.5**. Our selection criteria included (1) how well the platform interfaced with our EO-DT system and data streams, (2) if they produced a variety of customizable visualizations, including from geospatial data, and (3) if the operating costs were within our budget. Since our project focused on open-source solutions, we did not evaluate commercial platforms, like Tableau or Microsoft PowerBI.

Table 2.5 List of dashboard services evaluated and compared for the EO-DT.

Name/Preview	Website	Strengths	Drawbacks
Grafana	https://grafana.com/	- Many templates and plugin for GIS - Well documented online - Integrates with AWS services - Highly customizable	- Setup and learning are challenging
Dashbuilder 	https://www.dashbuilder.org/	- Does not require coding expertise - Examples included ML-model monitoring	- No fully managed version - Not easily integrated with other tools - Not easily customizable - Limited GIS support
Freeboard	http://freeboard.io/	- Simple to setup and learn - Easily add data sources - Easily shareable	- Fewer data visualization options - Requires JavaScript knowledge - Limited online community

Most dashboard platforms provide online interactive samples so potential customers can see if the dashboard meets their needs. While ease of use is important for a digital twin, we especially wanted to select a system that could handle many different visualizations. Dashbuilder and Freeboard had many types of bar and line plots, but Grafana has a more extensive number of plots. We did not need the dashboard to display raw satellite imagery, but we wanted it to display maps showing satellite positions and where users queried data products, which requires a map. While none of the dashboards had more than two visualizations, Grafana's maps were more customizable. For the demonstration, we also wanted to quickly deploy the system to focus on the types of analytics that would be useful for an EO-DT, so a low-cost, fully managed system was useful for us.

Of the three platforms, we ultimately selected Grafana because of its extensive online documentation integrations with AWS CloudWatch and S3. **Figure 2.9** shows the simplified architecture for our data to flow into Grafana. Data from CloudWatch flows directly into Grafana using an access key for security. From there, Grafana can access the hundreds of possible metrics on AWS resources. We also wanted to monitor how quickly user requests were processed, so we wrote Python scripts that tracked the CPU, memory, and run times of specific code. In the future, this process can be containerized and directly monitored in CloudWatch. We also wanted to observe data flows on the NODD, so we wrote Python scripts to capture the latency and quality flag statistics. Additionally, we monitored our anomaly detection subsystem's runtime and anomaly classification. These data flows are summarized in **Table 2.6**.

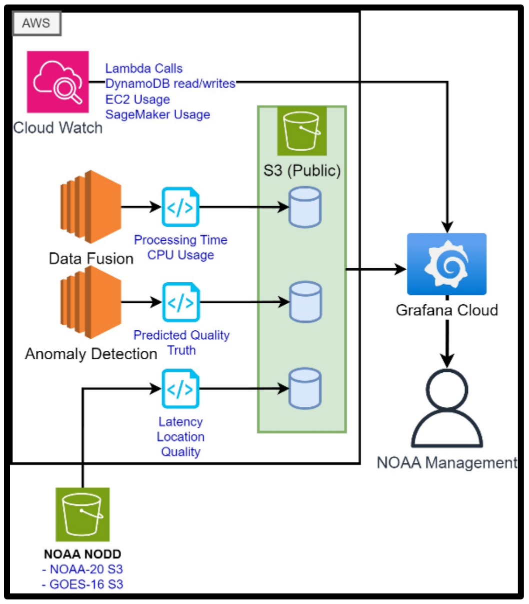


Figure 2.9 Data flow to the Grafana dashboard, which is fully managed via Grafana Cloud. Grafana displays analytics from multiple sources in a single location. CloudWatch tracks AWS analytics and Grafana can natively access the data in real-time. We wrote several Python scripts to monitor specific process in our EO-DT, such as the processing time for data fusion and whether data contains anomalies. We also monitored the NODD to check the latency and quality of Level 1 datasets.

There were some challenges to using Grafana. Generally, most dashboard services are designed for monitoring computer systems since software developers are the most likely to contribute to the code base. Grafana supports non-traditional visualizations, such as video and GIS, but these displays are not as mature as those designed for time series data. However, we felt that an off-the-shelf tool like Grafana was appropriate for the demonstration and could be used in a fully functional EO-DT (if built). We recommend that if NOAA builds an EO-DT, the development team gets a clear set of visualization requirements from the dashboard's various users and allows the system users to interact with the visualization examples online. If sufficiently complex, such as requiring very advanced GIS visualization, the dashboard will likely need to be built from scratch.

Table 2.6 The component being monitored in Grafana, along with the data source and the specific metrics that tracked in real-time.

EO-DT Component	Data Source	Metrics Tracked
Catalog, archive, and processing	CloudWatch	- Number of SNS/SQS - Lambda Invocation Sound - DynamoDB read/write - CPU/GPU usage
Search and dissemination	EC2 and Python Scripts	- Data products queries - Region users searched - Data fusion latency
Downlink and ingest	NOAA NODD and Python Scripts	- Product latency - Data quality - Anomaly detection

While we chose to use the fully managed Grafana Cloud, Grafana is open-source dashboards can be self-hosted for free. An advantage of Grafana Cloud is that the developer only must focus on setting up the dashboard and metrics and less on the infrastructure to support it. We discuss the differences in cost in **Section 5.3.2**.

2.6 References

Arora, K., Farr, E., Gilbert, J., & Zonooz, P. (2019). Architecting Cloud Native Applications: Design high-performing and cost-effective applications for the cloud. Packt Publishing.

AWS. (2020, June 22). Implementing geohashing at scale in serverless web applications | AWS Compute Blog. Retrieved October 29, 2023, from https://aws.amazon.com/blogs/compute/implementing-geohashing-at-scale-in-serverless-web-applications/

AWS. (n.d.). Using Global Secondary Indexes in DynamoDB - Amazon DynamoDB. Retrieved October 28, 2023, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html

NOAA. (2019). GOES-R SERIES PRODUCT DEFINITION AND USERS' GUIDE (No. 416- R- PUG- L2 Plus- 0349 Vol 5) (p. 726). Retrieved from https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf

Lin, W. D., & Low, M. Y. H. (2021). Design and Development of a Digital Twin Dashboards System Under Cyber-physical Digital Twin Environment. In 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) (pp. 1716–1720). https://doi.org/10.1109/IEEM50564.2021.9672870

Microsoft (n.d.). What Is a Data Dashboard? Retrieved October 27, 2023, from https://powerbi.microsoft.com/en-us/data-dashboards/

Shea, D. (2014, January 13). The Climate Data Guide: Regridding Overview. Retrieved October 27, 2023, from https://climatedataguide.ucar.edu/climate-tools/regridding-overview

Pauwels, K., Ambler, T., Clark, B. H., LaPointe, P., Reibstein, D., Skiera, B., et al. (2009). Dashboards as a Service: Why, What, How, and What Research Is Needed? Journal of Service Research, 12(2), 175–189. https://doi.org/10.1177/1094670509344213

Slayton, Z. (2017, May 17). Z-Order Indexing for Multifaceted Queries in Amazon DynamoDB: Part 1 | AWS Database Blog. Retrieved October 28, 2023, from https://aws.amazon.com/blogs/database/z-order-indexing-for-multifaceted-queries-in-amazon-dynamodb-part-1/

Steinberg, A., Bowman, C., & White, F. (1999). Revisions to the JDL Data Fusion Model. ERIM International, Inc. Retrieved from https://apps.dtic.mil/sti/tr/pdf/ADA391479.pdf

White, F. (1991). Data Fusion Lexicon (No. OMB No. 0704-0188). Retrieved from https://apps.dtic.mil/sti/pdfs/ADA529661.pdf

3 Enhancements using Machine Learning

3.1 Anomaly Detection in Satellite Datasets using Convolutional Neural Networks

NOAA has developed an extensive data product portfolio to monitor and forecast the environment. For the data to be useful, end users must be assured of the “trustworthiness” of the data, a challenge that NESDIS has met using strict statistical requirements. Data quality assurance is commonly provided to end users globally, using pixel-by-pixel quality assurance flags in NESDIS products (NOAA, 2019). Requirements can range from global comparisons with models and ground truth to strict error and standard deviation values, which can vary by Earth surface type, degree of cloud contamination, and solar zenith angle, to name a few. For example, the AOD product for GOES-16 has respective precision and accuracy of 0.25 and 0.05 for 0.1<AOD<0.8 over land globally when compared with AERONET, a network of in situ measurements (Laszlo and Liu, 2022). Distilling complex information like data quality into concise, digestible, and actionable information is challenging in big data analytics.

Moreover, it is also essential for the end user to understand data quality on a pixel-by-pixel and scene-by-scene basis. NESDIS communicates pixel-level quality using data quality flags, where each pixel is labeled as no retrieval, low, medium, or high quality. Reasons for the classification can vary from product to product, but for AOD, some factors include estimated cloud contamination, coastlines, bright surfaces, and high solar zenith angle. Communicating the uncertainty of datasets is essential for end users and decision-makers to arrive at correct and actionable conclusions based on the information they present.

Identifying scene-by-scene data quality is more challenging but possible using deep learning and real-time analytics within an EO-DT. Scene-based quality control requires understanding baseline patterns and robust detection of deviating from the norm. We define dataset anomalies as unexpected and widespread data quality degradation. While algorithm and sensor level strengths and weaknesses are well known, classifying regions of anomalous data (and labelling by anomaly type) has not been thoroughly explored. ML-based solutions are important because simple counts of scene quality flags alone cannot identify if anomalies are caused by sensor degradation or data outages due to, for example, solar flares. Thus, utilizing deep learning has the potential to identify regions of poor or noisy performance and provide correction (Gibert et al., 2018).

In this section, we develop and demonstrate an anomaly detection system prototype within a the EO-DT to classify imagery with real-time anomalies. **Figure 3.1** shows examples of anomalous data consisting of stripes or chunks of missing values. As the GOES and VIIRS satellites produce satellite imagery, the Digital Twin Anomaly Detector (DTAD) subsystem analyzes Level 1 data. It determines if the imagery is valid using a pre-trained ML model instead of relying on quality flags. By relying on a pre-trained model, the digital twin can adapt to new anomalies as they arise.

On a broader scope, we envision an anomaly detection scheme to process data in real-time and apply labeling and possibly data correction where possible. In our prototype, we seek to (1) identify which deep learning techniques and frameworks can meet NOAA requirements, (2) measure the skill of the approach to progressively more complicated results, and (3) design the baseline cloud architecture needed to implement such a system for cost and resource needs.

The DTAD subsystem is configured to perform near real-time classification of anomalies, which serves as a gatekeeping mechanism for other downstream systems within the DT service architecture. This prevents the dissemination of images containing anomalies to end-users.

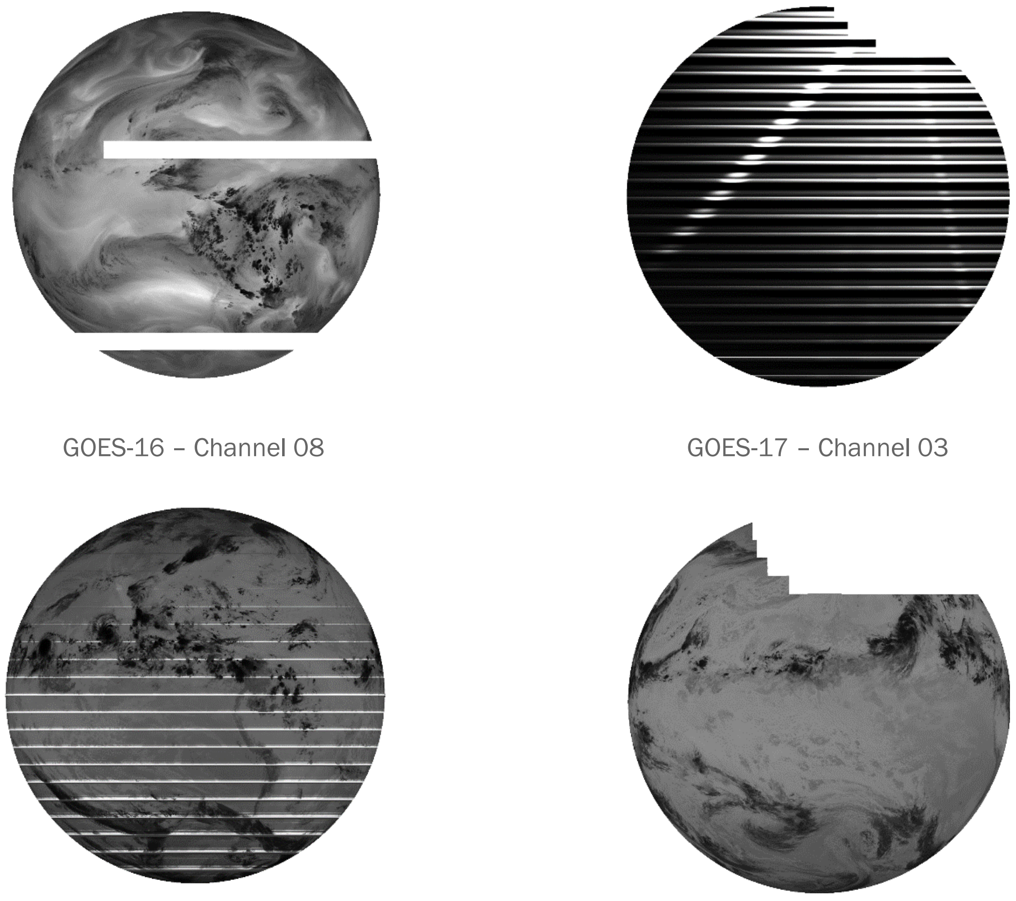


Figure 3.1 Examples of anomalies from GOES-R series L1b data showing missing data with various patterns.

3.1.1 Methods and Datasets

The DTAD constitutes a comprehensive toolkit developed due to the labor-intensive task of curating and classifying training data to create a highly accurate ML model. Within the DTAD framework, a set of utilities were devices to aid in assembling a dataset that consists of anomalous and non-anomalous satellite imagery. This dataset was used to train a new ML model using the PyTorch framework. In addition to the data preparation utilities, DTAD incorporates tools for model validation against new datasets to ensure performance continues to meet anticipated standards. Finally, the subsystem is engineered to perform near real-time analysis of data from the GOES and VIIRS satellites and offer timely classification of newly acquired imagery based on the presence or absence of anomalies.

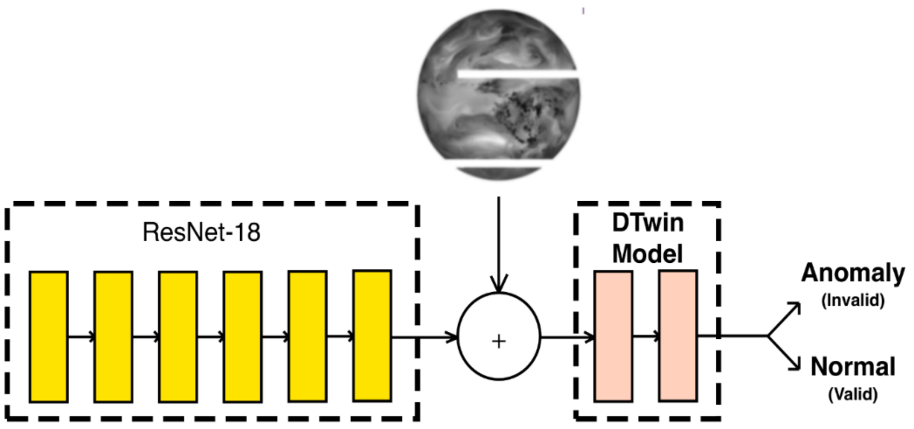


Figure 3.2 Overview of the Binary Image Classification training paradigm.

The DTAD is a binary image classification system (**Figure 3.2**) trained using Convolutional Neural Networks. As a binary image classification system, the DTAD can answer the question of “Does this image contain anomalies?” with a “Yes” or “No” answer. Throughout the project, the DTAD was trained to be a multi-label classification system to try to answer the question “What type of anomaly does this image have?” with multiple anomaly times, including “blanks” (when the entire scene was black) “horizontal-stripe noise,” and “missing” (when the entire scene was white) image.

3.1.1.1 Training for GOES ABI Data

The DTAD subsystem initially underwent training utilizing data from the GOES series. All 16 channels from GOES-16, GOES-17, and GOES-18 satellites were selectively downloaded from their corresponding S3 storage locations (**Figure 3.3**). The data was then split into "valid" (indicative of an anomaly-free state) and "invalid" (denoting the presence of anomalies) categories, as determined by the evaluation of the image's quality flags. Images with quality flags attaining a value of 0.989 or higher were categorized as "valid", while those registering a value below 0.989 were designated as "invalid.” We estimated an optimal threshold of 0.989 through iterative experimentation.

Considering we needed thousands of images for optimal classification, the DTAD subsystem was augmented with a suite of utilities. These tools streamlined the download of GOES ABI imagery from AWS S3 storage and enabled subsequent classification based on a given classification criterion, such as 0.989 or 0.999. Moreover, an auxiliary utility was developed to enhance the efficiency of the iterative threshold determination process for the quality flag. This tool allows the operator to input a novel quality flag value, facilitating the immediate reclassification of the image sets. Concurrently, the utility segregates the images, directing them to their respective "valid" or "invalid" directories.

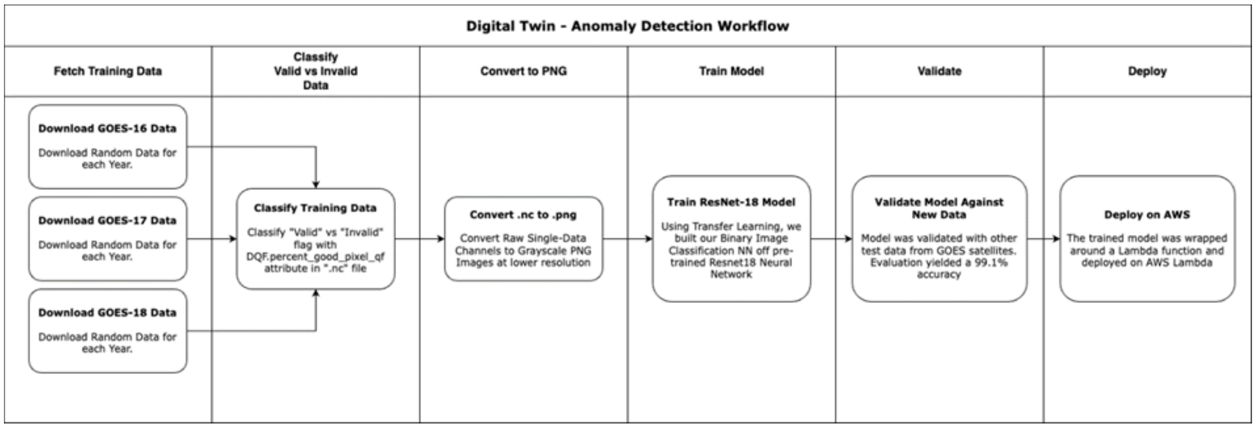

Digital Twin - Anomaly Detection Workflow
Fetch Training Data
Classify Valid vs Invalid Data
Convert to PNG
Train Model
Validate
Deploy
Download GOES-16 Data
Download Random Data for each Year.
Download GOES-17 Data
Download Random Data for each Year.
Download GOES-18 Data
Download Random Data for each Year.
Classify Training Data
Classify "Valid" vs "Invalid" flag with DQF.percent_good_pixel_qf attribute in ".nc" file
Convert .nc to .png
Convert Raw Single-Data Channels to Grayscale PNG Images at lower resolution
Train ResNet-18 Model
Using Transfer Learning, we built our Binary Image Classification NN off pre-trained Resnet18 Neural Network
Validate Model Against New Data
Model was validated with other test data from GOES satellites. Evaluation yielded a 99.1% accuracy
Deploy on AWS
The trained model was wrapped around a Lambda function and deployed on AWS Lambda

Figure 3.3 Overview of the Anomaly Detection Workflow for GOES data.

Images procured from the GOES satellites were contained within large HDF files. For enhanced manageability and training feasibility, we downsampled these HDF files into more compact PNG files in the early stages of the project. To optimize and ensure better model performance, every image was integrated into the training framework, undergoing randomized transformations. We used this methodology to prevent incorporation of external biases in the training dataset. Specific transformations included converting images to grayscale, arbitrary flipping along horizontal and vertical axes, and resizing to 250x250 pixels. The data were flipped to prevent overfitting based on location or semi-persistent features.

Before commencing the training process, the entire dataset undergoes a randomization procedure, partitioning into an 80% segment for training and a 20% fraction reserved for testing. Partitioning the data helps guarantee that the data utilized for weight verification during backpropagation in training is distinct yet remains representative of the overarching training dataset.

Upon ascertaining the optimal quality flag threshold and standardizing the requisite image transformations, the next logical progression entailed initiating the model's training phase. The desired accuracy for the model's performance was to predict images 75% of the time or more correctly.

Early in the project, we had to decide which ML Python package to use for training. TensorFlow and PyTorch are both free, easily accessible, open-source software libraries. TensorFlow is older and thus has more online documentation and community support. PyTorch is a popular tool because of its user-friendly interface and flexible design. Despite PyTorch's comparatively smaller ecosystem and restricted multi-GPU support, it was designated as the chosen framework for DTAD. This decision was due to several factors: PyTorch's seamless integration with Python and its more intuitive API, the provision of dynamic computational graphs that render it particularly conducive for intuitive processing and experimental endeavors, its robust foothold within academic circles, and its native support for the Open Neural Network (ONNX) format.

After segregating the images into their respective directories, labeled as "valid" or "invalid," we initiated model training. We initially chose to use a convolutional neural network (CNN) with a custom-trained model. However, the model's accuracy did not achieve the desired benchmarks. The most proficient model trained under this paradigm attained an accuracy of 75% in the binary classification of satellite images. Given how large some of the anomalies were, the accuracy was lower than expected.

We used transfer learning to enhance the accuracy of the model. Instead of building a CNN model from scratch, we leveraged a pre-existing model and adapted it to our labeled data. ResNet-18 is a model trained on over a million images from the ImageNet database, categorizing images into 1,000 distinct object classifications, including items like keyboards, mice, and pencils (**Figure 3.4**). The Resnet CNN was selected for transfer learning due to its ability to address the vanishing gradient problem. Vanishing gradient occurs during backpropagation when the calculated gradients from the output layer progressively wane in magnitude as they traverse back toward the input layer. Vanishing gradients lead to suboptimal learning in the initial layers, resulting in longer training durations.

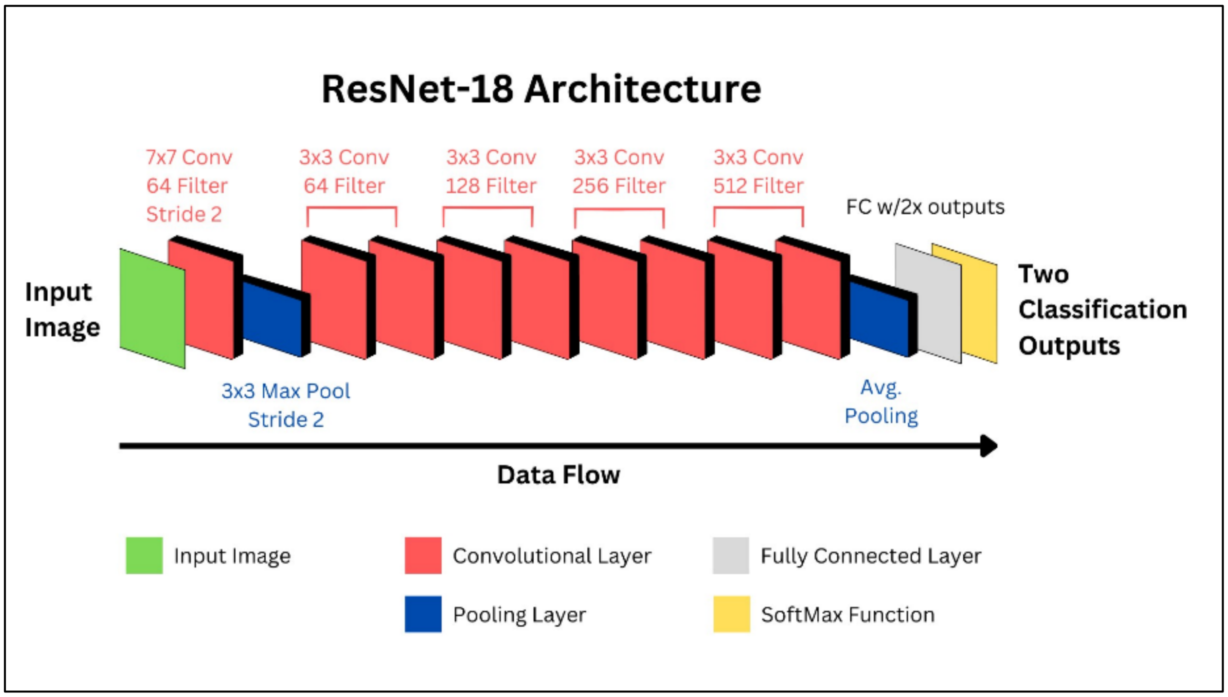


Figure 3.4 Overview of the ResNet 18 model. Adapted from https://arxiv.org/abs/1902.08897

We selected Resnet CNN because it incorporated a distinct neural network component called a residual block. This block is designed specifically to counteract the vanishing gradient issue. Within the residual block is a skip connection which circumvents one or more layers. This architectural design facilitates more streamlined backpropagation, enhances learning, and accelerates training speed.

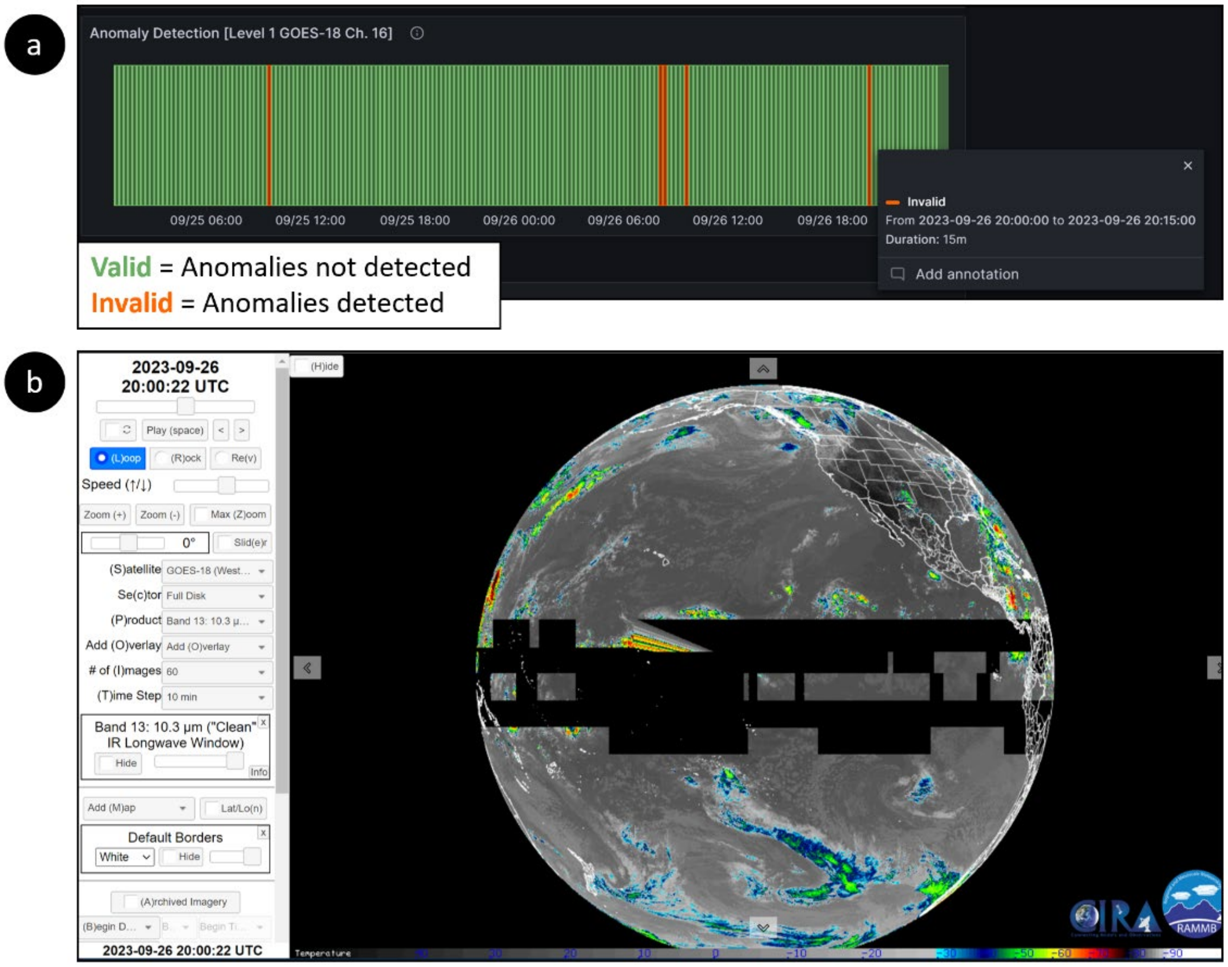
The pre-trained ResNet 18 model was trained on GOES-16, GOES-17, and GOES-18 satellites for anomaly classification with an accuracy of 99.2%, much higher than that of the custom CNN model (**Table 3.1**).

Table 3.1 Confusion matrix showing the ResNet-18's performance classifying anomalies found in GOES ABI L1b imagery between 04/01/23 - 04/05/2023.

		Actual Classification	
		Anomaly (Invalid)	**Normal** (Valid)
ML Classification	**Anomaly** (Invalid)	94 True Positive	14 False Positive
	Normal (Valid)	19 False Negative	3,836 True Negative
Accuracy: 99.2%			

We initially envisioned installing the DTAD subsystem in a Lambda function. However, the training model alone was 43 MB, and PyTorch can exceed 100 MB, exceeding the maximum Lambda deployment size. Owing to this limitation, the DTAD subsystem was instead deployed on an EC2 instance using a cronjob set to execute every 15 minutes. In case of an error, the DTAD subsystem records the last checked timestamp, ensuring that any new files added since its previous run are processed. Consequently, in scenarios where the system downtime exceeds 15 minutes, it can process all files from the point of the last operation.

Upon deployment and when tested against real-world data, the model's accuracy experienced a slight drop, descending from 98.9% to 96.5%. The lower is expected because there may be anomalies present in unseen data that do not match that in the training dataset. After installation, we found numerous examples of the DTAD performing well. **Figure 3.5a** shows a screenshot of the DTAD installed in the EO-DT dashboard. The dashboard displays a time series of the GOES-18 ABI Channel 16 state, where green means the data are valid and orange means invalid. At 2023-09-26 20:00 UTC, an anomaly was detected. A subsequent check on the CIRA SLIDER web portal shows missing data (**Figure 3.5b**). The detection scheme exceeded our requirements despite the slight reduction in detection capability when processing unseen data.



Invalid
From 2023-09-26 20:00:00 to 2023-09-26 20:15:00
Duration: 15m

Valid = Anomalies not detected
Invalid = Anomalies detected

2023-09-26
20:00:22 UTC

0°

(S)atellite GOES-18 (West...
Se(c)tor Full Disk
(P)roduct Band 13: 10.3 µ...
Add (O)verlay Add (O)verlay
# of (I)mages 60
(T)ime Step 10 min

Band 13: 10.3 µm ("Clean" IR Longwave Window)

Default Borders
White

2023-09-26 20:00:22 UTC

Figure 3.5 Example of the DTAD installed on the EO-DT. (a) shows the DTAD state diagram for GOES-18 ABI channel 16 in the dashboard, where green indicates valid data and orange is invalid. At 2023-09-26 20:00 UTC, an anomaly was detected. (b) the real-time image, indicating an anomaly (Source: https://rammb-slider.cira.colostate.edu).

3.1.1.2 Training of NOAA-20 VIIRS Data

Leveraging the GOES anomaly model, we developed an analogous strategy for VIIRS imagery data. However, VIIRS had fewer real-world anomalies, and it was more challenging to build a training dataset. Instead, we created a synthetic training dataset that injected noise into images, mimicking real-world anomalies found in MODIS sensor data (Ren et al., 2010; Rakwatin et al., 2017). Two examples of striping patterns are shown in **Figure 3.6**. The "valid" directory was populated with 1,000 unmodified files from the NODD VIIRS S3 bucket using synthetic images. Then, the "invalid" directory was populated with the same batch of 1,000 files altered with artificial noise infused or rendered entirely in stark white or pitch-black shades.

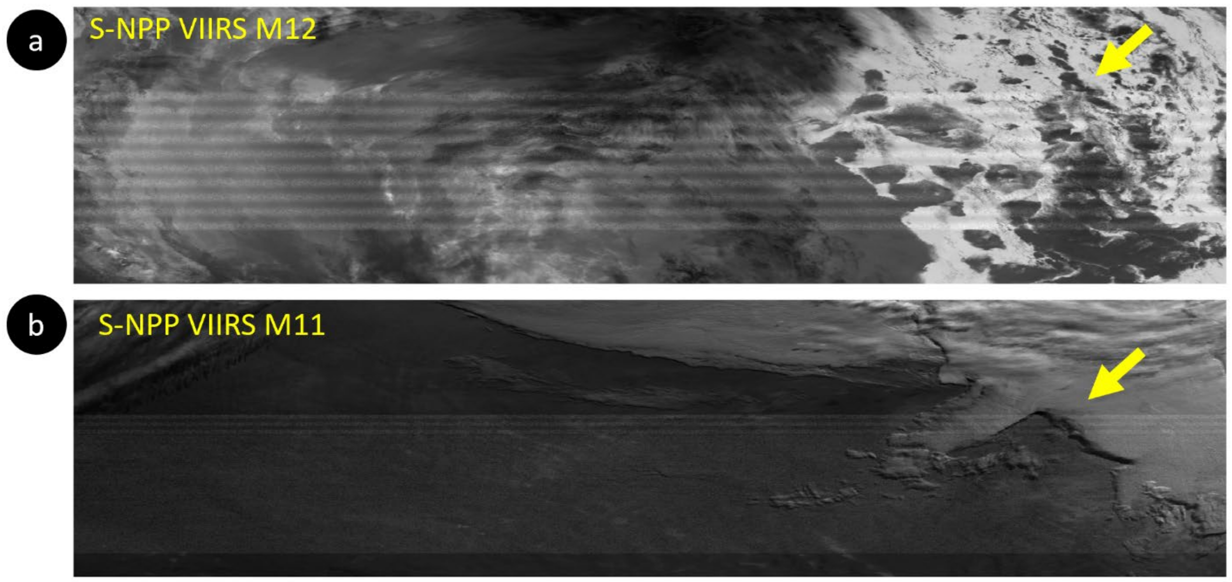


Figure 3.6 Examples of synthetic anomalies using noise injected into valid VIIRS data. (a) Shows stripes in the center of an M12 image (indicated by the yellow arrow), which the model must distinguish from natural features like clouds. (b) shows another example for the M11 band, with more closely spaced stripes.

Similar to the approach with the GOES data, we used the ResNet 18 model for the VIIRS data. However, the resulting model underperformed, attaining only a 50% accuracy rate. We identified a significant error in our training approach. The images labeled as "invalid" were the same as the "valid" images that had undergone noise injection. We recreated the training dataset, but this time, the “invalid” images’ base image layer was changed to be completely different from the “valid” images. With this rectification, there was a marked improvement in the model’s accuracy, reaching 89%, which was better than our goal of 75% accuracy for the model’s performance.

3.1.2 Multi-Label Classification

Once the trained binary classification model’s results met our accuracy goals, we explored using a multi-label classification framework, designed to test if ML between different types of anomalies in each dataset. Instead of using two labels, we developed a training dataset where the labels describe the type of anomaly they contained, such as “valid,” “blank,” “horizontal stripe,” and “missing.” While still simplistic, these labels could lead to more sophisticated anomaly detection. We again applied ResNet-18, with the final layer consisting of four distinct outputs corresponding to each of the expected labels in the training dataset.

The original dataset contained folders with the folder names corresponding to each label used for multi-label classification. This data was split into 80% for training and 20% for testing. After training, the model was evaluated against the test dataset but was notably suboptimal. When evaluating against the test dataset, the model’s accuracy for the “valid” label was 0%, and its predictions for “blanks,” “horizontal stripes,” and “missing” categories appear to be arbitrary, lacking any discernible pattern. However, we are optimistic that the model could achieve better results with refinement.

We have several strategies that might enhance model performance. Throughout the model training iterations, we observed a clear correlation between the quality of the training data and the model's accuracy. A potential avenue of exploration would involve leveraging a training dataset comprised of authentic, real-world examples instead of relying on synthesized data. This approach would better mirror real-world conditions, akin to the GOES dataset, and could significantly bolster the model's predictive capabilities. An alternative direction for enhancing the model involves architectural modifications.

While the Resnet-18 model has 18 layers, the Resnet family encompasses other variations like Resnet-34, Resnet-50, Resnet-101, and Resnet-152. By experimenting with these architectures, there could be potential for improved model performance.

Fine-tuning the ResNet-18 hyperparameters may improve model performance. One hyperparameter to tweak is the learning rate, which determines the step size of each iteration while moving toward a minimum of the loss function. Smaller learning rates converge the step size slowly, while larger ones might overshoot the minimum. Another is to change the batch size of the training dataset as the model could have been given more training data to increase the accuracy but might also have decreased the model's ability to generalize.

Finally, the project could have also explored further Epoch values, which determine the number of times the learning algorithm will work through the entire training dataset. Even though the project did tweak with different Epoch values for binary image classification, it was not a rigorous enough test for multi-label classification. Finally, the project could have also explored tweaking the Optimizer and trying out different optimization functions to see if they improve the model's performance.

3.1.3 Explainable AI with Heatmaps

Explainable Artificial Intelligence (XAI) is the concept of interpretability and transparency of an AI model. XAI refers to the methods and techniques that allow for a straightforward, understandable elucidation of the decision-making processes of ML models, particularly deep learning architectures and other complex models. XAI aims to demystify black-box models, increasing trust in ML.

Gradient-weighted Class Activation Mapping (Grad-CAM) is an XAI solution to understand better how the model is predicting the data it has predicted as "valid" or "invalid" in our DTAD subsystem. Grad-CAM is a technique designed to enhance the interpretability of CNNs by visualizing which parts of an input image contribute the most to the network's final decision. This is achieved by creating a heatmap highlighting the image's most influential regions.

The power of Grad-CAM lies in its ability to bridge the gap between model performance and human interpretability. Grad-CAM allows for this by visually pointing out the salient regions in the input data and offers the ability to make better decisions for debugging and refinement and developing trust with transparency into how the model functions.

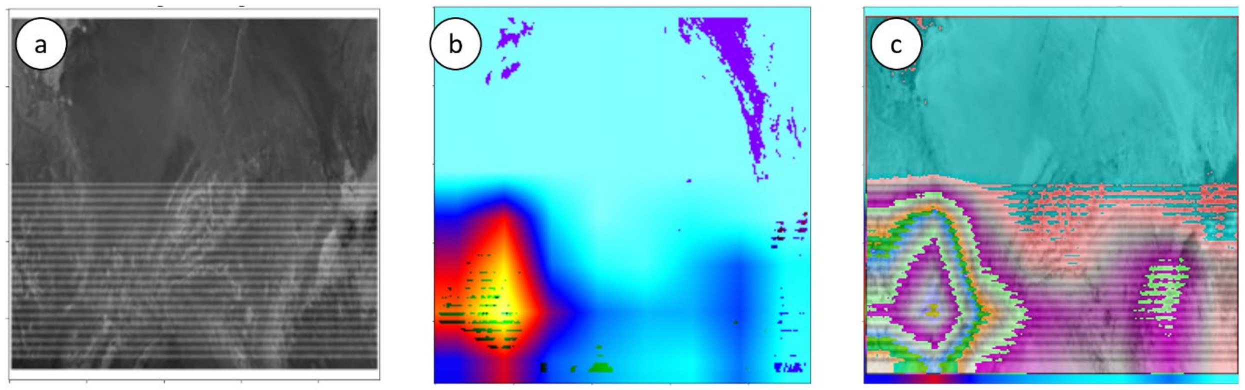


Figure 3.7 Invalid image classified by the DTAD subsystem. (a) shows the synthetic banding artifact added to VIIRS I-band 4. (b) is the heatmap showing regions of "interest" to the RESNET-18 model. (c) is the combination of the two images to show that banding region correctly led to classification

Figure 3.7 shows four examples of "invalid" VIIRS images, their associated heatmaps, and the combination. On the left, (a) shows an invalid image with synthetic stripes on the bottom half and (b) the associated Grad-CAM heat map. The redder colors indicate that the ResNet-18 is assigning more weight to the region of the image. By combining the two images in (c), we can see that the heatmap coincides with the location of the stripes. We found the heat maps helpful in refining our approach.

3.1.4 Summary and Lessons Learned

Our study showed a proof-of-concept for anomaly detection using ML. While our approach labels images as "valid" or "invalid," our methods can be extended to identify the type of anomaly that was detected. Transfer learning emerges as a more straightforward approach to image classification, leveraging pre-trained models to expedite the learning process and often achieving commendable results. This approach saves computational resources and significantly reduces the time required for model training. We also explored how explainable artificial intelligence methods can be incorporated into the anomaly detection workflow. In particular, we found heatmaps useful for understanding what image features contributed to the ResNet-18 model's classification.

In terms of lessons learned, it is evident that models tend to perform better in offline testing and training than when running in real-time. This underscores that even with long training periods, there is inherent unpredictability in real-time data, and the models may confuse natural phenomena as an anomaly and vice versa. Secondly, our analysis suggests that training ML models with natural GOES-16 anomalies yield better results than synthetic data with the VIIRS data. For ML-based anomaly detection to advance as a field, we recommend that NESDIS science teams help create large, labeled repositories of training data for the community to explore.

3.2 Deep Learning Methods to Enhance Data Fusion in a Digital Twin

Classical methods of data fusion are described in **Section 2.4**. Classical data fusion methods involve coding and well-understood mathematical algorithms. ML methods embody a slightly different approach. In an ML software system, the 'machine' is treated as a software black box and is trained on observational data. More than one data set type may be used in training (e.g., AOD and fire locations). The machine 'learns' the behavior of the data. The machine is then tested on data sets withheld from the training. The tests are evaluated using the root mean square error (RMSE), for example, and if the error rate is low enough, the ML system can be applied to new data sets confidently. The advantage of ML systems is that they can often outperform classic techniques regarding computational speed, taking advantage of GPU architectures. The disadvantage is that the ML devised scheme inside the black box is hidden from the user. This means the user often doesn't know what the ML is "seeing," and unusual results are sometimes inexplicably produced. Explainable ML described in the **Section 3.1.3** bridges the gap between these classical methods and ML methods.

We used two different ML systems described in **Table 3.2**. The first, convolutional long short-term memory (ConvLSTM), was tested with AOD measurements to determine if it could fill in the AOD gaps produced by clouds (Daniels et al., 2022). The second, the enhanced super-resolution generative adversarial network (ESRGAN; Ledig et al., 2017; Wang et al., 2018), was used to produce higher-resolution AOD data sets so that lower-resolution ABI data could be compared to high-resolution VIIRS measurements. **Table 3.2** provides definitions of the techniques and their analogous, classical equivalent.

Table 3.2 Definitions of two ML models that can facilitate data fusion of satellite observations.

ML Technique	Definition	Classic equivalent
ConvLSTM **Convolutional – long short-term memory**	This type of recurrent neural network for spatial-temporal prediction has convolutional structures in both the input-to-state and state-to-state transitions. The ConvLSTM determines the future state using the inputs and past states and their local neighbors.	Spatial/temporal back filling and inpainting
ESRGAN **Enhanced super-resolution generative adversarial network**	**ESRGAN** is the enhanced version of the SRGAN. Starting with SRCNN, ESRGAN is a generative adversarial network for single-image super-resolution. It uses a perceptual loss function, which consists of an adversarial loss and a content loss, to improve the image iteratively.	Interpolation using higher order polynomials such as cubic spline.

3.2.1 Gap filling with ConvLSTM

ConvLSTM integrates structures in both spatial and temporal dimensions, making it ideal for sequence prediction in multi-dimensional data (Shi et al., 2015). Unlike pixel-to-pixel straight backfill or inpainting, ConvLSTM focuses on spatial structures. In theory, ConvLSTM can efficiently handle spatial-temporal data, capturing spatial hierarchies and temporal dependencies better than classical data fusion methods.

When a sequence of images passes through ConvLSTM layers, filters compress the data by extracting important features from the pixel time series and their neighboring ones, thus retaining both the spatial and temporal characteristics. The ConvLSTM architecture has two network structures, which are the encoding network and the forecasting network, consisting of stacked with convLSTM layers (**Figure 3.8**). The encoding layer is composed of convolutional layers and LSTM cells, which compress the data into hidden layers and weigh features by their importance. The forecasting network copies the output state from the corresponding encoding layer and then decodes the hidden state to predict future values.

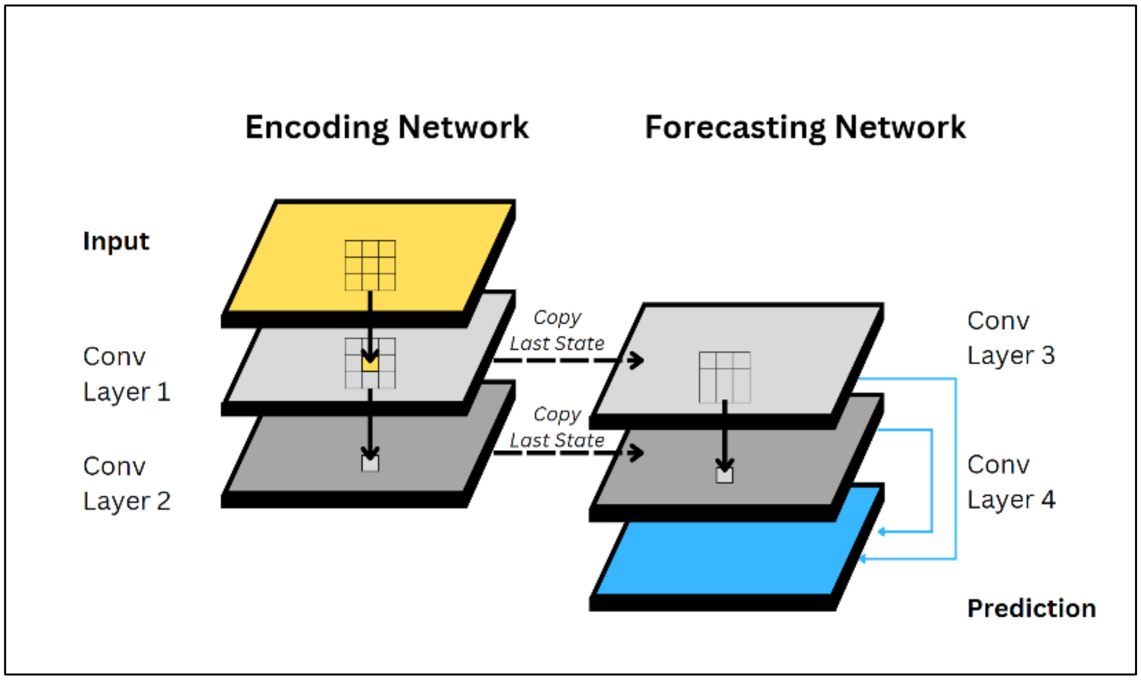


Figure 3.8 Network architecture for the ConvLSTM. The input layer is a 3D representation of a satellite dataset, e.g., an observation at location x, y, and time t. In the encoding network, each layer compresses the input to hidden states (ConvLSTM Layer 1) and hidden state to other hidden states (ConvLSTM Layer 2). In the forecast network, the previous hidden state (ConvLSTM Layer 1 and 2) is copied and decoded (ConvLSTM 3 and 4). Adapted from Shi et al. 2015.

For our study, we used the "default" ConvLSTM encoder parameters for three layers, which had 64, 96, and 96 hidden states, respectively. We used a 5x5 filter on cells and a 3x3 kernel on all layers (see `convlstm_encoder_params` and `convlstm_decoder_params` in https://github.com/jhhuang96/ConvLSTM-PyTorch/blob/master/net_params.py).

To test the appropriate use of ConvLSTM, we evaluated its ability to predict AOD in a region where no data is available but previously available data. For example, imagine a fire-generated smoke region with high AOD moving from west to east. In this scenario, an AOD anomaly is observed, and then the region is covered by clouds. Based on the structure of the AOD field, can we try to predict the air quality below the clouds.

For our evaluation, we trained a ConvLSTM model using GOES-16 AOD. We had to construct a training dataset that reduced the full disk image (5424x5424 pixels) into small samples that were 64x64 pixels. Each smaller image consists of 12 timestep observations of AOD. We needed all the images to be nearly cloud-free, which was challenging given that the mean cloud fraction of the earth is 60% (King et al., 2013). Thus, building an extended training dataset was time-consuming. Using a year of data, we created 800 training samples (of 12 timesteps each) and validated/tested with 300 samples to refine the model. GOES-16 AOD is generated every 10 minutes, and we initially used all available observations to train out data. However, scene-to-scene changes in AOD can be small, so the ConvLSTM predictions were unrealistic. Instead, we switched to 20-minute timesteps and saw more realistic propagation of AOD plumes.

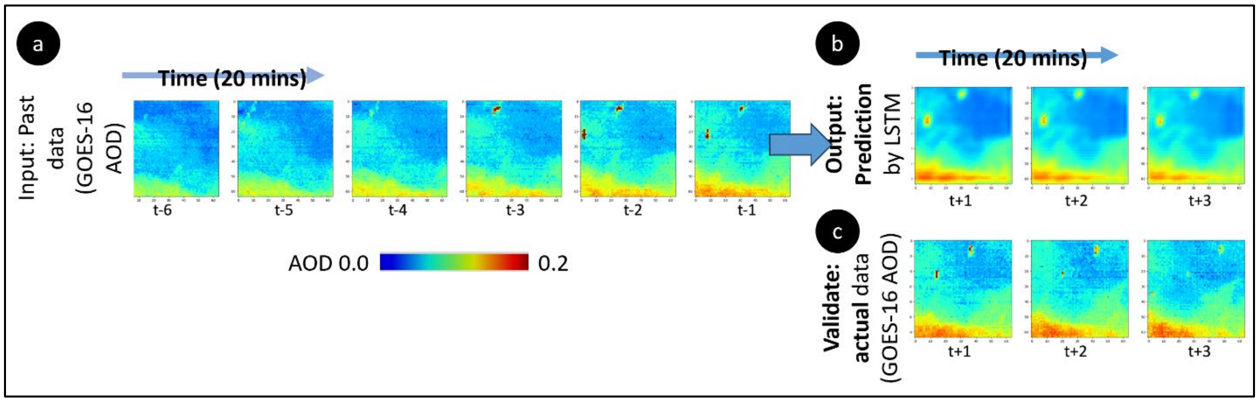


Figure 3.9 AOD prediction using ConvLSTM. Training data is at the top row. Images of AOD are separated by hours. Starting with image 5, ConvLSTM predicts 6-11 (bottom row). Validation is the middle row.

Figure 3.9 shows an example of AOD prediction of ConvLSTM for a cloud-free scene. In **a**, we used two hours (six, twenty-minute timesteps) of 64x64 pixels to predict the next several timesteps, shown in **b**. Compared with **c**, the observed AOD for the next three timesteps, the predicted results do an excellent job capturing areas of higher AOD. The model can make predictions further in time, but the results were unrealistic. Given that we are applying the ConvLSTM model to fill in clouds, not predict AOD, it is more critical that the next time step (t+1) agrees with the observed values.

We show how a ConvLSTM in the previous example can be used for gap filling in a digital twin in **Figure 3.10**. In ❶, the GOES-16 ABI data is used to construct a time series and predict the current time (t+1) in ❷. The observed values may contain clouds, sunglint, or another feature that reduces AOD retrieval quality, leading to missing values. We manually removed several clouds instead of using scenes with clouds so that we could evaluate the result. These missing pixels in the observed data can be filled with the predicted values. In ❸ the combined image is compared with the actual scene for validation purposes. We could repeat this process to evaluate ConvLSTM offline, and if the results are favorable, the approach could be installed into the EO-DT.

The ConvLSTM shows promise for gap filling. In our evaluation, we ignore advection and chemistry's effects on longer AOD time scales, so backfilling over a few hours can approximate the observations. However, incorporating other observations, such as GFS winds, will likely improve our results. Long-term validation is needed before incorporating into an EO-DT, which will likely require a multi-year training and test period using GOES AOD data.

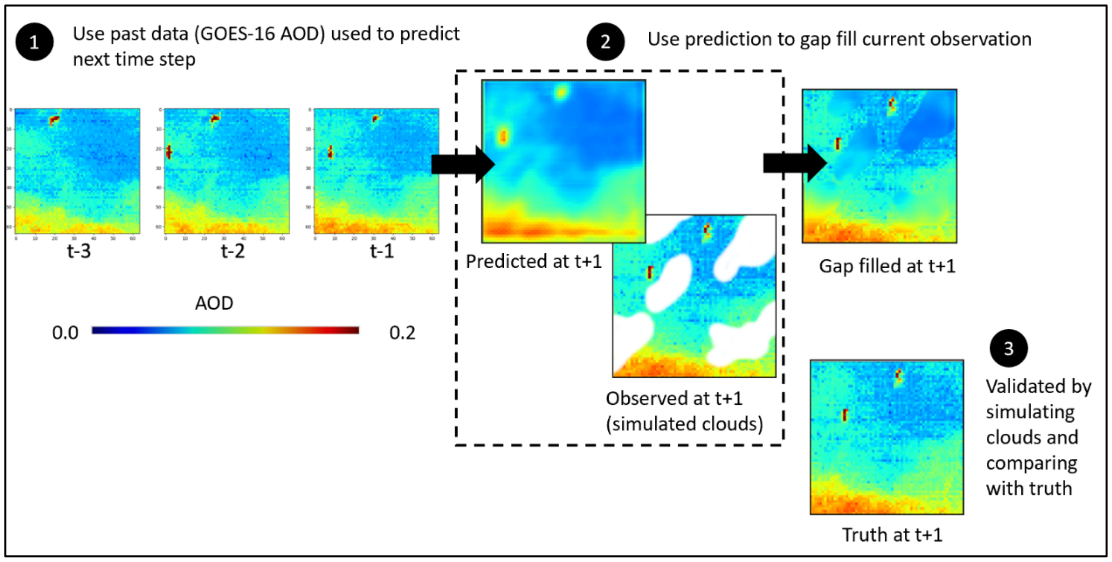

1
Use past data (GOES-16 AOD) used to predict next time step
2
Use prediction to gap fill current observation
t-3
t-2
t-1
AOD
0.0
0.2
Predicted at t+1
Observed at t+1 (simulated clouds)
Gap filled at t+1
3
Validated by simulating clouds and comparing with truth
Truth at t+1

Figure 3.10 Proof of concept for gap filling in an EO-DT. In 1, a convLSTM model predicts the next time step (t+1) using the past three-time steps of AOD (~1 hour of past data). In 2, the predicted AOD is compared with AOD observed at t+1, regions observed with simulated missing clouds can be replaced with the predicted values. Results can be validated using the unmodified t+1 observed values, where the clouds have not been removed.

3.2.2 Enhancing image resolution using ESRGAN

A higher-resolution data set often provides more insights into geophysical phenomena. However, the optics and detector signal-to-noise set the spatial resolution of most satellite data sets. GEO data sets often cannot match the kilometer-scale granularity needed for the location of fires or small pollution sources, limiting the potential of these data.

One approach to overcome this limitation is using super-resolution techniques. Here we used a variant of SRCNN called ESRGAN (Wang, et al., 2018; Tsang, 2018). **Figure 3.11** illustrates how ESRGAN can be applied to the same retrieval algorithm (e.g. AOD) when available from two different sensors at different times and resolutions. The ESRGAN model can be trained using collocated AOD from NOAA-20 VIIRS, which has a higher spatial resolution but has lower temporal resolution, and the lower spatial resolution/high temporal resolution AOD products from GOES-16/18. Then, the model could be given a GOES image, which is then downscaled to the resolution of VIIRS, even if a VIIRS dataset is not present.

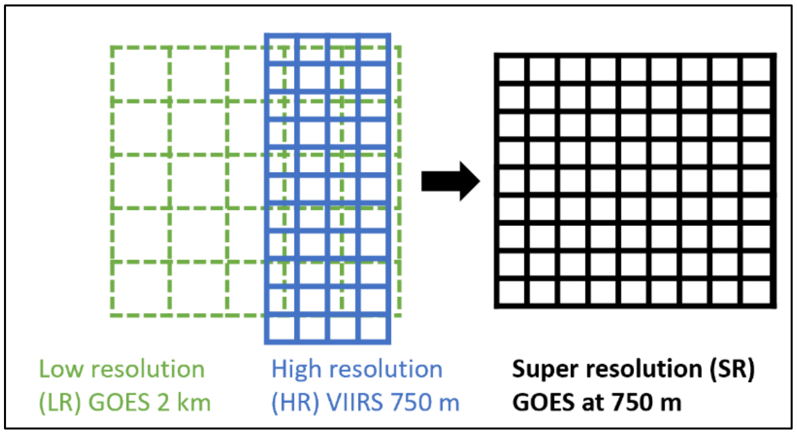

Low resolution (LR) GOES 2 km
High resolution (HR) VIIRS 750 m
Super resolution (SR) GOES at 750 m

Figure 3.11 illustrates how super resolution uses a generative adversarial network (GAN) can be trained to increase the scale when there are two similar datasets with different resolutions.

ESRGAN employs deep learning to upscale and improve low-resolution images, leveraging the adversarial relationship between a generator (creates high-resolution images) and a discriminator (distinguishes between real and generated images). By iterating this process, the generator improves its outputs. Training the system produces a better resolution of coastlines and other geographical features and sharpens the gradients near the edges of the AOD distribution. Like any interpolation, ESRGAN is "creating data" and can occasionally produce unrealistic results. Another challenge is that ESRGAN requires extensive training datasets and GPUs to process the data. As a result, we used a pre-trained model to evaluate the off-the-shelf version and see if it was feasible for an EO-DT.

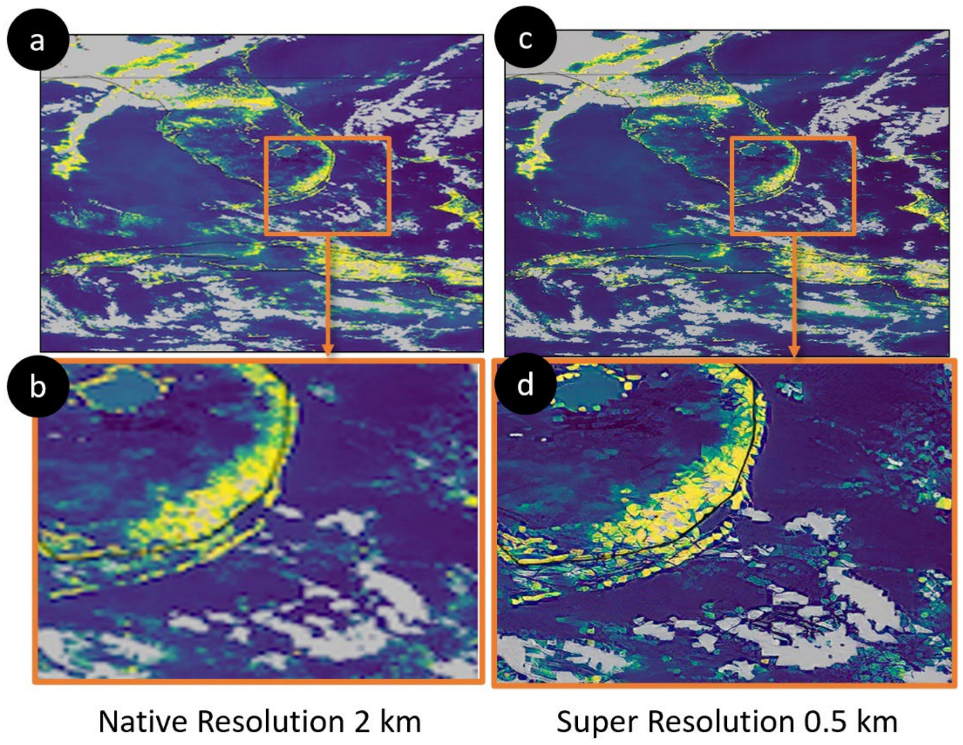


Figure 3.12 Example for GOES-16 AOD L2 Feb 23, 2023, 14:30 UTC using an off-the-shelf pre-trained model that doubles the resolution. Grey pixels are missing values. (a) shows the zoomed out original GOES-16 AOD image and (b) zooms in on the finer details. (c) is the same as in a but processed using the SRGAN model. (d) shows the zoomed in view, where pixels are clearer.

Figure 3.12 shows an example of the pre-trained ESRGAN sharpened resolution of a sample GOES-16 AOD image. On the left (**a** and **b**), the GOES-16 AOD values are coarse when zooming into the focus region over Florida. The features are sharpened on the right-hand side (**c** and **d**) after processing with the ESRGAN model. The results show good visual agreement but need further validation with VIIRS AOD and AERONET before installing into the EO-DT. In the pre-trained model output, the pixel shapes take on a granular geometry, which may be controlled by developing a custom-trained model. Overall, ESRGAN shows promise for improving the spatial resolution of earth observations, especially for products available on different platforms.

A challenge of using this approach is that the ESRGAN will sharpen noise in the datasets. For example, coastal regions are an area of uncertainty in AOD models because of bright surfaces and a mixture of land and water scenes, which use different retrieval models. ESRGAN does not ensure thermodynamic consistency of AI/ML based and traditional based methods to understand the quality. Thus, it's important to vigorously validate datasets using in situ and high-quality model data to understand and apply quality control to the image, otherwise ESRGAN will enhance the underlying noise. In the future, a large, multi-year training dataset of collocated VIIRS and ABI would allow us to train a custom ESRGAN model specifically for Earth Science data.

3.2.3 Summary

Our results suggested that ML systems have the potential to produce excellent data-fused products, but the approach must be nuanced. Users cannot just submit training data to an ML package and expect good results. Understanding the physics behind the data is required, and the ML package may need to be trained using multiple variables. In our example, AOD predictions in regions where data gaps occur require some knowledge of the processes that change AOD, such as pollution sources, winds, and chemistry.

The unique characteristics of ABI data meant that our ML models were continually at risk of overfitting to specific patterns. This could potentially jeopardize their effectiveness when introduced to new, unseen data. Another concern revolved around the need for additional training data. While ConvLSTM shows promise in filling data gaps, it is evident that supplementary training datasets are essential for generating satisfactory outcomes. The foundation for selecting this additional training data should be rooted in a comprehensive understanding of the processes under simulation.

Additionally, despite the remarkable capabilities of ML techniques, ML models tend to produce artifacts. These anomalies can misrepresent or distort the data. To address this, rigorous testing of the methods is imperative. Moreover, a post-simulation review of datasets generated by ML can identify and rectify most of these discrepancies.

Synergistic utilization of both classic and ML techniques will provide superior data fusion. While classical techniques like backfilling or polynomial resampling provides a simplified approach to changes in resolution and data gaps, ML can provide superior results under the right circumstances. That said, ML techniques cannot be applied haphazardly and must be thoroughly tested before implementation.

3.3 References

Daniels, J., Bailey, C. P., & Liang, L. (2022). Filling Cloud Gaps in Satellite AOD Retrievals Using an LSTM CNN-Autoencoder Model. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium* (pp. 2758–2761). https://doi.org/10.1109/IGARSS46834.2022.9884482

Gibert, K., Horsburgh, J. S., Athanasiadis, I. N., & Holmes, G. (2018). Environmental Data Science. *Environmental Modelling & Software*, *106*, 4–12. https://doi.org/10.1016/j.envsoft.2018.04.005

King, M. D., Platnick, S., Menzel, W. P., Ackerman, S. A., & Hubanks, P. A. (2013). Spatial and Temporal Distribution of Clouds Observed by MODIS Onboard the Terra and Aqua Satellites. *IEEE Transactions on Geoscience and Remote Sensing*, *51*(7), 3826–3852. https://doi.org/10.1109/TGRS.2012.2227333

Laszlo, I., & Liu, H. (2022). EPS Aerosol Optical Depth (AOD) Algorithm Theoretical Basis Document (p. 91). NOAA.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., et al. (2017, May 25). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. arXiv. https://doi.org/10.48550/arXiv.1609.04802

NOAA. (2019). GOES-R SERIES PRODUCT DEFINITION AND USERS' GUIDE (No. 416- R- PUG- L2 Plus- 0349 Vol 5) (p. 726). Retrieved from https://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf

Rakwatin, P., Takeuchi, W., & Yasuoka, Y. (2007). Stripe Noise Reduction in MODIS Data by Combining Histogram Matching With Facet Filter. *IEEE Transactions on Geoscience and Remote Sensing*, *45*(6), 1844–1856. https://doi.org/10.1109/TGRS.2007.895841

Ren, R., Guo, S., Gu, L., & Shao, X. (2010). Stripe noise removal method for MODIS remote sensing imagery. In 2010 2nd International Conference on Computer Engineering and Technology (Vol. 1, pp. V1-565-V1-569). https://doi.org/10.1109/ICCET.2010.5485965

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W., & Woo, W. (2015, September 19). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. arXiv. https://doi.org/10.48550/arXiv.1506.04214

Tsang, S.-H. (2018, Sept 5). Review: SRCNN (Super Resolution). Retrieved October 30, 2023, from https://medium.com/coinmonks/review-srcnn-super-resolution-3cb3a4f67a7c

Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., et al. (2018, September 17). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. arXiv. https://doi.org/10.48550/arXiv.1809.00219

4 Optimizing Data for Interoperability

In this chapter, we explore why knowledge graphs and cloud-optimized data are essential for data exploitation because these enhancements can improve data discoverability and usability. Knowledge graphs can semantically represent relationships between phenomena in the Earth Sciences, such as linking datasets with certain types of weather phenomena. Knowledge graphs can also connect standard processes performed on data, such as regridding and gap filling. This structured representation could also form the basis for an integrated digital twin system. Cloud storage ensures accessibility and scalability of data, making it more efficient to disseminate and use datasets. The two enhancements are interlinked because knowledge graphs can link different twins, and the cloud-optimized formats can facilitate data exchange.

4.1 Metadata and Knowledge Graphs

Much focus is made on the usage and formats of data in a digital twin. However, each file generates additional data describing the data, which is coined metadata. Most people are familiar with metadata, thanks to public libraries. A person can find a specific book in the library because it has properties like a title, author, and publication year. It is also possible to browse books in a library because they are organized by genre. Each book is given a main class and then further divided into subclasses. For example, under the Dewey Decimal System, 500 is for Natural Science and Mathematics, and 520 is for Astronomy.

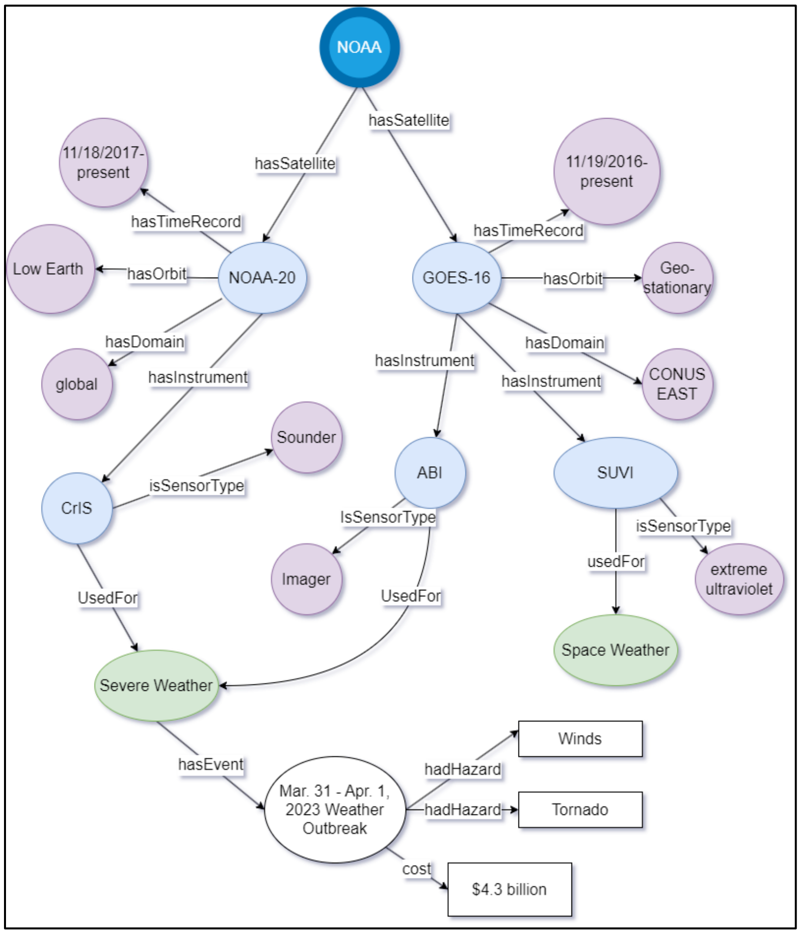


Figure 4.1 Illustration of knowledge graph connecting NOAA satellite assets ("NOAA-20"), their properties ("global"), and the relationships between them ("hasDomain"). Knowledge graphs can improve the searchability and discovery of datasets as well as link to other databases to create a larger, interconnected federation. Consistent metadata formats are a key requirement for success.

Much like a library, it is important for users to find a specific dataset and discover similar and related datasets. This is where models like knowledge graphs can connect relationships between datasets using metadata (**Figure 4.1**). Knowledge graphs describe the relationship between objects following a subject-object-verb format. For example, VIIRS could be mapped by (`VIIRS`, `is_a`, `imager`). While this example is simplistic, we could use knowledge graphs to link concepts like sensors to high-level data products, like AOD, and then link to natural hazards (wildfires). This can help users interested in natural hazards access a catalog of data.

Metadata is key to digital twin semantic interoperability because there is no centralized warehouse for all Earth observations. Instead, major data providers - NOAA, NASA, ESA, ECMWF, and CNES - all have their large data warehouse stores. These stores must operate as a data mesh to leverage all these data sources. Data meshes treat data as a product and distribute ownership and responsibility across multiple domain-specific teams. Returning to our library example, a data mesh would be the equivalent of having access to another library's book through an interlibrary loan. The NSF and other international organizations have championed open knowledge, which they have coined the knowledge commons (McGranaghan et al., 2023).

Libraries successfully share their book collections because they have a common framework to catalog their items, such as the Dewey Decimal system. The earth science community is working towards establishing a universal metadata format and access method across all organizations. A common metadata framework and robust API can enable a better "handshake" and data exchange between digital twins.

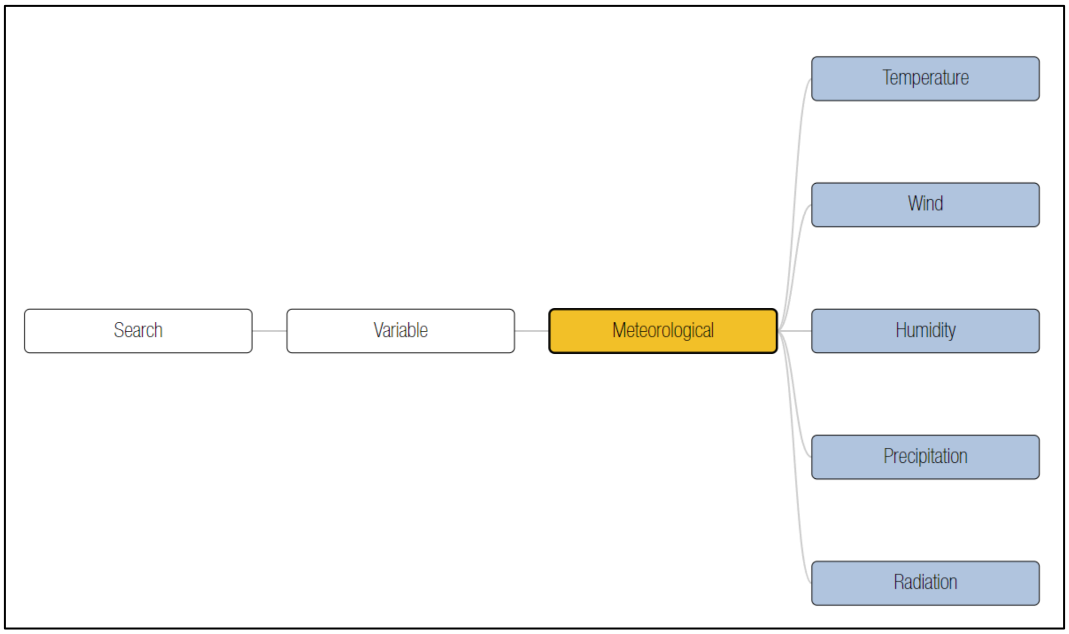


Figure 4.2 Example of a data catalog for fire weather on WIFIRE commons (https://wifire.ucsd.edu/). Users can browse nearly 3,000 datasets

One popular catalog model for geospatial data is STAC. In STAC, specifications are intended to make data more searchable. The core catalog fields included properties like the bounding box and the datetime of the scan, identifiers like constellation and platform, and other information on loading and processing the data. However, data providers can add hundreds of additional metadata fields using an Asset Object. The Asset Object could be where NESDIS adds additional data product tags, such as the thematic area, associated hazard, and even models that can process the data. For example, VIIRS and ABI AOD could be associated with atmospheric composition (thematic area), air quality (hazard) and wildfires (hazard), and the HYSPLIT trajectory model (simulation). Then, AOD could be linked to online catalogs of wildfire events to extract dates, geolocations, and even economic costs. **Figure 4.2** shows a real-world example of a knowledge graph-enhanced data catalog, the WIFIRE commons (https://wifire.ucsd.edu/). Users can search by datasets or variables; their ultimate selection will provide a list of all available products, their sources, and how to access them.

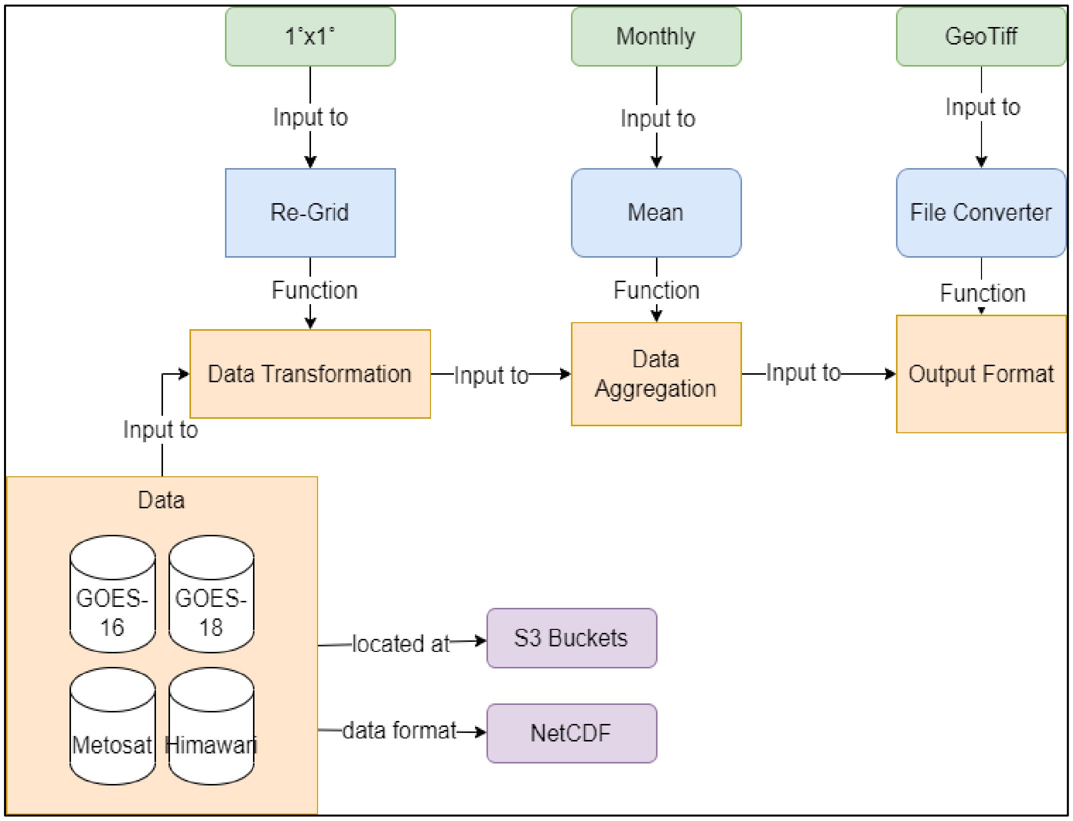


Figure 4.3 Simplified illustration of how knowledge graphs can map actions performed on datasets. This can create greater transparency for NOAA open science goals and better error and uncertainty tracking within processes.

In addition to linking the datasets, knowledge graphs can be used to connect data workflows, such as extractions, transformations, gap filling, and resolution enhancement (Berkheimer et al., 2023). **Figure 4.3** shows an example where a researcher examines monthly mean SST using a combination of GOES, Meteosat, and Himawari satellites and regrid to a 1˚ cartesian grid. Data could be extracted from a catalog, transformed into a new grid, and aggregated into a monthly mean. The researcher can also save the resulting data in their format of choice, GeoTiff.

If applied to our EO-DT, knowledge graphs can improve data discoverability. Searching by data product requires users to know the data exists and identify a specific use care they are interested in. With knowledge graphs, a user could start with the event they are interested in and find the relevant data products without prior knowledge of what those products are.

In our EO-DT prototype, users were allowed to regrid their data to a standard grid for data fusion. With knowledge graphs, users could create workflows using predefined transformations. Using the STC EO-DT as an example, a user could request data that is easily compared with the GFS model on a regularly spaced grid. The EO-DT could use ontology to chain processes, such as regridding to a 1x1 grid and perform gap filling and combine the result with GFS. The user would see the final product without having to understand and code the full complexity of the data transformations.

The concept of data workflows is an ongoing area of interest in the Earth Sciences. For example, Kepler (https://kepler-project.org/index.html) is a software tool that provides a GUI for scientists to chain containerized processes with limited coding. Keppler has been used to build a successful workflow for fire weather (Nguyen et al., 2017). While open source, a drawback is that the software was developed for the pre-cloud era and would not be viable in a digital twin. One major goal of NASA AIST is the development of the Analytics Collaborative Framework (ACF) to permit scientists to harmonize data management and perform analysis. This framework was successfully deployed for an air quality focused ACF (Huang et al., 2022). The project provides a one-stop sandbox for scientists to study air quality, with access to relevant datasets from satellites, models, and in situ. Scientists can use AWS SageMaker to run ML and popular models like GEOS-Chem. At the time of writing, the ACF can build workflows but only supports data transformations and does not appear to incorporate knowledge graphs to improve data or process searchability.

The most fully realized example of using knowledge graphs for EO-DT-like goals was by Shimizu et al., 2023 via a space weather use case. The authors developed an ontological framework that links Solar Flare events to power grid disturbances. They developed a model that initiated a workflow if a Solar Flare Event was detected, which would respond by pulling together NOAA, SWPC, and GOES satellite observations, use that data to run a model to estimate ionospheric currents and simulate the geomagnetic field of the earth and the resulting induced currents. Their study did not combine and run simulations but defined how a theoretical workflow could be labeled.

Shimuza et al., 2023 developed their ontology though the following questions (1) What datasets are available to view? (2) What does dataset X contain? (3) In what ways is dataset X used? (4) What is the result of dataset X transformed by Algorithm A? (5) What dataset X was used for input to Simulation S? Satellite, in situ, and model data, as well as algorithms and models, can be better connected by working with science teams to label their resources better and incorporate that into the existing catalogs.

When people think of interoperable digital twins, they may naively assume that the exchange only occurs on a data level. Data exchange between providers like NOAA and NASA is invaluable, but to truly support interoperability, processes also must be exchanged. A genuinely interoperable digital twin should be able to pass data from a NASA digital twin, preprocess the data on a NOAA digital twin, and process it using an EUMETSAT forecast model. This is a challenging software problem, but the importance of cataloging data and processes cannot be overlooked.

4.2 Cloud Optimized Data Formats

Digital twin data access must meet the needs of two major end users: (1) data providers who must archive, maintain, and disseminate vast amounts of data, and (2) the scientific community, who need to access the data in a consistent and analysis-ready format to ingest the data into models and simulations. The first community requires compression, stability, and reduced computing costs; the latter requires timeliness, findability, and ease-of-use in their software ecosystem. System architecture requirements sometimes clash with the community's needs, as data would be costly to store in multiple formats or in every conceivable option.

The two formats that serve the data archive community are the netCDF4 and HDF5 formats. HDF was developed in the 1980s as a cross-platform data format to address big data challenges in the scientific community (Folk, 2010). NetCDF (Network Common Data Form) is a set of libraries that can read multiple binary formats, including the netCDF4 format, which builds on HDF5 but with additional data requirements. Presently, netCDF4 is one of several international standard formats of the Open Geospatial Consortium.

Other formats, such as GRIB and BUFR, address big data challenges and were adopted by the WMO for modeling purposes (Caron, 2011). These formats are less common in newer satellite data products because, among other reasons, they require local tables to read the contents and thus are not entirely self-contained as netCDF4. Additionally, these formats are more rigidly organized, which is challenging for Level 2 and higher data products.

NetCDF4 files provide excellent compression, readers/writers are readily available, and most data is stored in these formats. While these formats are well-understood by more advanced users, they are not cloud-optimized.

Under previous data access paradigms, users would search and download the data using data portals, such as NASA's Earthdata and NOAA CLASS. The NOAA NODD program allows users to access, analyze, and utilize large datasets directly in the cloud without local downloads. This approach enhances the user experience, ensuring rapid data access, seamless integration with analytical tools, and efficient data manipulation in a cloud-based environment. By shifting towards this cloud-centric model, data archive communities and researchers benefit from the streamlined process, minimizing the challenges traditional data access methods pose.

What makes a data format "cloud-optimized?" For one, cloud-optimized data is designed to be processed across multiple nodes. The STC EO-DT explores this using serverless technology, which enables handling of multiple search queries and user data fusion requests. Under the current paradigms, NetCDF4 metadata is not centrally located in the file. As a result, users can download the entire file, even if only a single field is needed (Ambatipudi and Byna, 2023).

Cloud-optimized data keeps metadata in a central location so the user can download just a chunk of the data (e.g., only GFS *u/v* vectors, not the entire 20 GB GFS file). It is more challenging to search and subset when metadata and variables are scattered across the file.

Chunking is a technique that divides large datasets into smaller, more manageable pieces called "chunks." Optimizing chunk sizes is crucial for cloud-based data processing and storage because retrieving many small files concurrently is often faster than one large file sequentially in a cloud system. With cloud storage solutions like Amazon S3, you pay for the storage space and the number of read/write requests. Chunking can reduce the number of requests by ensuring that only the necessary chunks are accessed. The benefit to the end user is that they have less data to transfer.

Table 4.1 summarizes some of the popular cloud optimized formats and file system solutions. Zarr is a format for storing chunked, compressed, N-dimensional arrays. These characteristics make Zarr especially suitable for scenarios where data are frequently accessed non-contiguously, such as in cloud-based storage systems. Zarr divides arrays into equally sized chunks, which can be read or written independently. This is particularly beneficial for parallel and distributed computing. Zarr also stores metadata related to the array, including information about its shape, data type, and chunks. This metadata is stored in a human-readable format, typically JSON. As a result, end users who use the cloud increasingly request that data providers like NOAA provide their data in Zarr (Abernathey et al., 2018).

Table 4.1 Comparison of cloud-optimized approaches and data formats

Solution	Strengths	Weaknesses
"Retrofit" NetCDF4 Kerchunk NCZarr	Uses existing computing standard	NCZarr stores are not fully compatible and interoperable with Zarr V2, might be addressed in Zarr V3. Slower access than using Zarr
Zarr	Cloud optimized Supports data filters OGC endorsed on 6/30/2022	Must re-process processing/duplicating the original data Introducing new format to community
GeoTiff (Cloud-Optimized)	Community Standard Cloud Optimized	Too simple for multi-dimensional/complex data datasets
Parquet, ORC, other database formats	Cloud and ML optimized	Works best for tabulated data formats (However, GeoParquet beta available as of Dec 2022)

However, adopting Zarr in the Earth sciences presents its own set of challenges. As evidenced by the AWS registry of Open Data, out of 479 datasets, only 11 are in Zarr, whereas 20 are formatted in netCDF (20), HDF (6), or GRIB2 (7). This preference for traditional legacy formats highlights that it would be costly to reformat archives to Zarr. Instead, modifying filesystems like NetCDF4 to incorporate the principles underlying Zarr may be more pragmatic. Fortunately, existing libraries, such as Kerchunk, allow the creation of virtual Zarr datasets (Sterzinger, 2023). The performance trade-off of this approach is typically minimal. As illustrated in **Figure 4.4**, while Zarr processing times were only slightly faster for single file processes, they were notably quicker — up to twice as fast — for multifile processes (Augspurher, 2022).

Another popular format, HDF, similarly upgraded the HDF5 library to be cloud-optimized (Jelenak, 2023). One hurdle for HDF is that the default chunk size is 1MB, whereas AWS best practices recommend 8-16 MB chunk sizes. Like netCDF, metadata are spread throughout the file. The metadata can be consolidated in paged aggregation, combining all the metadata on a single “page” in the file. Unfortunately, this solution requires the entire archive to be reprocessed.

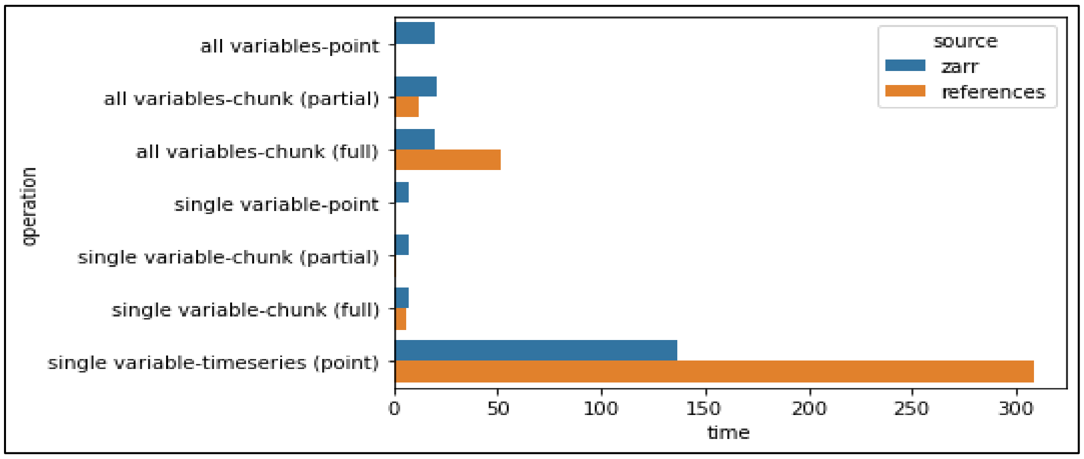


Figure 4.4 Benchmark tests for the NASA CMI6 dataset comparing processing speeds of accessing (1) a Zarr Archive and (2) NetCDF4 archive using a Zarr-like file system (Kerchunk) [Image credited to Augspurger, 2022]

GeoTIFF, Parquet, and ORC are all file formats designed or adapted with specific optimizations that make them well-suited for cloud-based applications. GeoTIFFs can be made into cloud-optimized GeoTIFFs (COG) by organizing the file's internal structure to enhance the efficiency of partial reads. While COGs are great for imagery, they might not always be ideal for other types of multidimensional remote sensing datasets that may have multiple variables and time steps. Parquet and ORC are both cloud-optimized and primarily designed for tabular data. Earth science datasets often consist of multi-dimensional arrays, which do not map naturally to Parquet's columnar format.

In summary, while newer formats have cloud-specific advantages, legacy formats like netCDF and HDF remain deeply entrenched in the community's practices and toolchains. Both Unidata and the HDF group, which manage the two filesystems, are actively studying and implementing solutions to libraries for these formats to reinforce their resilience and adaptability as systems evolve to the cloud. Considering this, it is pragmatic for institutions like NOAA and NASA to continue leveraging these formats. Transitioning to a new format would not only entail considerable resources for converting large archives but may also introduce complexities and unforeseen challenges in the community. Staying the course with netCDF and HDF ensures stability, continuity, and the ability to benefit from ongoing innovations in their ecosystems.

4.3 References

Abernathey, R. P., Hamman, J., & Miles, A. (2018). Beyond netCDF: Cloud Native Climate Data with Zarr and XArray, 2018, IN33A-06. Presented at the AGU Fall Meeting Abstracts.

Ambatipudi, S., & Byna, S. (2023, February 5). A Comparison of HDF5, Zarr, and netCDF4 in Performing Common I/O Operations. arXiv. Retrieved from http://arxiv.org/abs/2207.09503

Augspurger, T. (2022, January 12). Recommendation for hosting cloud-optimized data - Data. Retrieved October 30, 2023, from https://discourse.pangeo.io/t/recommendation-for-hosting-cloud-optimized-data/2063

Berkheimer, R. (2023, April). NOAA Steps to the Geoverse: Shaping the Next Generation of Earth Systems Compute. Presented at the Committee on Earth Observation Satellites, Cordoba, Argentina. Retrieved from https://ceos.org/document_management/Working_Groups/WGISS/Meetings/WGISS-55/1/2023.04.18_15.10_NOAA%20Steps%20to%20the%20Geoverse.pdf

Caron, J. (2011, April 25). On the suitability of BUFR and GRIB for archiving data. Retrieved October 30, 2023, from https://www.unidata.ucar.edu/blogs/developer/en//entry/on_the_suitability_of_grib

Folk, Mike (2010). About Us. Retrieved October 30, 2023, from https://www.hdfgroup.org/about-us/

Huang, T., Chung, N., Dunn, A., Hovland, E., Kang, J., Loubrieu, T., et al. (2022). An Advanced Open-Source Platform for Air Quality Analysis, Visualization, and Prediction. In IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium (pp. 6574–6577). https://doi.org/10.1109/IGARSS46834.2022.9883227

Jelenak, Aleksandar (2023). Cloud-Optimized HDF5 Files, The HDF Group #HUG23. Retrieved from https://www.youtube.com/watch?v=bDH59YTXpkc

McGranaghan, R., Klein, S., Cameron, A., Young, E., Schonfeld, S., Higginson, A., et al. (2021). The need for a Space Data Knowledge Commons. Structuring Collective Knowledge. Retrieved from https://knowledgestructure.pubpub.org/pub/space-knowledge-commons/release/4

Nguyen, M. H., Crawl, D., Li, J., Uys, D., & Altintas, I. (2017). Automated scalable detection of location-specific Santa Ana conditions from weather data using unsupervised learning. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 1203–1212). https://doi.org/10.1109/BigData.2017.8258046

Shimizu, C., McGranaghan, R., Eberhart, A., & Kellerman, A. C. (2020, September 28). Towards a Modular Ontology for Space Weather Research. arXiv. https://doi.org/10.48550/arXiv.2009.12285

Sterzinger, L. (2023, October 25). Fake it until you make it — Reading GOES NetCDF4 data on AWS S3 as Zarr for rapid data access. Retrieved October 30, 2023, from https://medium.com/pangeo/fake-it-until-you-make-it-reading-goes-netcdf4-data-on-aws-s3-as-zarr-for-rapid-data-access-61e33f8fe685

5 Transitioning the Prototype to an Operational System

The STC EO-DT prototype was a foundation for evaluating the technologies and services for building a digital twin. NESDIS's goal for the prototype EO-DT was to "enhance our ability to process, monitor, quality-control, consolidate, fuse, and assimilate environmental observations." NESDIS's prototype goals are operationally focused and closely aligned with NESDIS' mission, which is to provide secure and timely access to global environmental data and information from satellites. Following our assessment, an operational EO-DT can meet these goals and become a basis for a more extensive Earth system digital twin, especially if combined with other digital twin efforts.

In this section, we discuss how to scale up our prototype infrastructure, identify essential milestones for interoperability, and provide an estimate of the cost. Per the report disclaimer (**page 4**), NOAA is not planning for activity beyond the demonstration projects. This section is written for informational purposes only.

5.1 Operational Architecture

Figure 5.1 shows the architecture diagram of our EO-DT when scaled from a prototype for demonstration to operations. The architecture diagram incorporates many of the same on-demand processes used in the prototype diagram (**Figure 2.1**). A fundamental change is that we replaced our EC2 with containerization to improve scalability. In the top right corner of the diagram, you can see that the regridding and the anomaly detection are stored as images in the Elastic Container Registry, which is run using AWS Fargate and Batch. AWS Batch dynamically provisions an optimal quantity and type of computing resource for containers based on the requirements of the submitted job. The container runs on Fargate, which handles the provisioning and managing of servers. This combination is more straightforward than Kubernetes because you will not need to choose server types, decide when to scale your clusters, and optimize cluster packing. AWS step functions are used to manage which batch workflow is run. For example, if a new SNS indicates a new GOES-16 file is available, the AWS step function would run the anomaly detection container using batch and Fargate. If the user ordered a regridded data file, the regridding container would run and save the output file to the S3.

A benefit of this approach is that it can be expanded to incorporate any number of processes so long as they can be stored in a container. For example, a scientist can quickly write code to combine multiple files and save it on a container. A developer could then easily update the step function and batch for this new process, and Fargate would automatically manage and provision the resources in the EO-DT.

The alternative to using a serverless combination of Batch and Fargate is a self-managed service like Elastic Kubernetes Service (EKS). EKS is more cost-effective and could be used with AWS Stepwise, but it would require manually provisioning compute resources. The serverless approach would allow science teams to manage their operational deployments, whereas using Kubernetes would require an engineer to oversee the deployment process. So, we recommend the serverless approach if it is within an organization's budget.

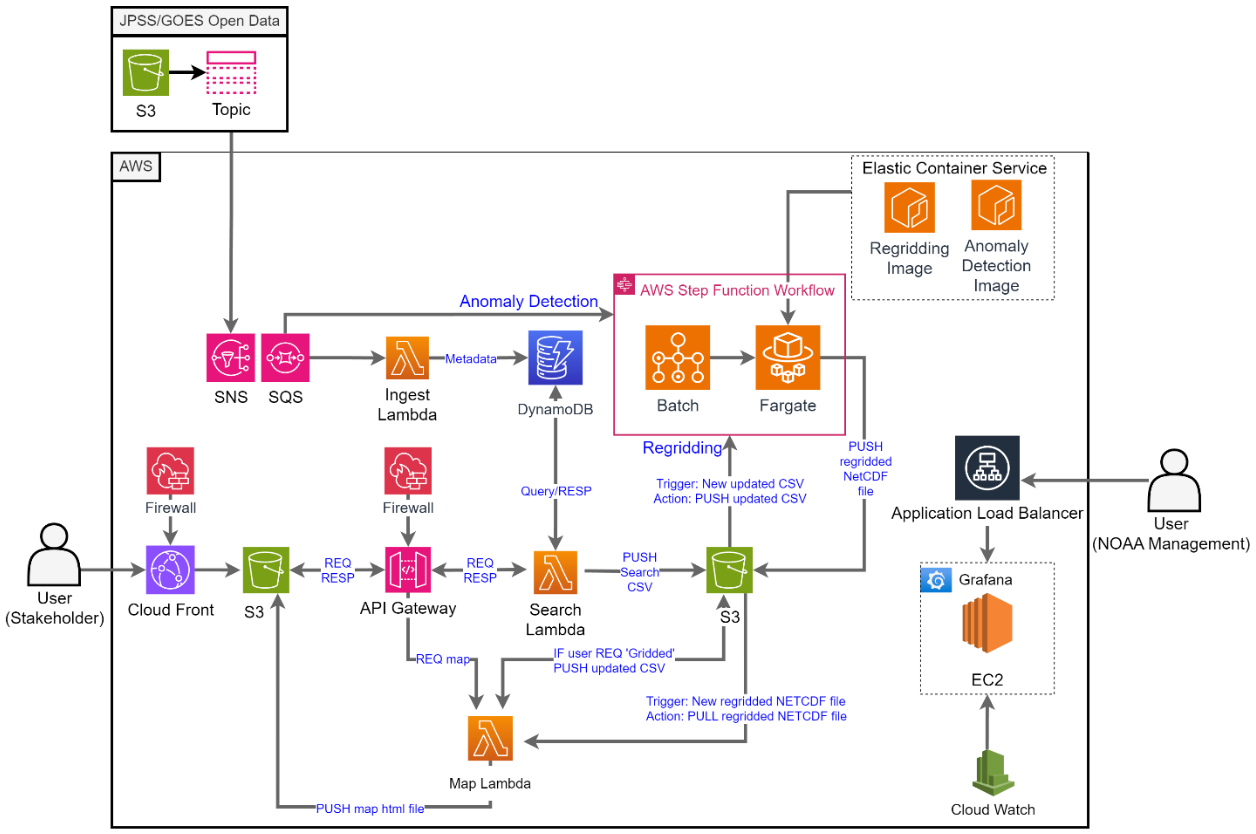


Figure 5.1 The architecture diagram for a fully operational version of the EO-DT in this project.

We recommend adding a Web Application Firewall (WAF) to Cloud Front and API Gateway for additional security. WAFs can filter which users can access the digital twin and block or limit access if a user uses the system excessively. In addition to blocking threats, the firewall can help reduce costs.

Finally, we added Grafana to the operational system. The Grafana dashboard requires an EC2 instance to run (m6g.4xlarge) and can be accessed by NOAA management using a public IP address. For added security, we recommend putting the EC2 behind an Application Load Balancer, which can have WAF rules applied to it. The added benefit of fully modularizing our system is that it improves transparency. In the prototype, we wrote python scripts to provide custom monitoring, but in the operational system, these requests can be monitored by CloudFront. If additional custom scripts are needed, they can be containerized and added as a stepwise function to execute.

5.2 Interoperability with External Digital Twins

A major concern of the community is how to leverage the numerous earth science digital twin efforts worldwide and open them to the scientific community. NASA, ESA, and EUMETSAT are building digital twins with different goals and unique approaches. The capabilities of individual digital twin efforts can be improved by seamlessly exchanging data and information with other digital twins, a process known as interoperability. Opening the interoperable, federated digital twins to external users benefits the scientific community by improving the findability, accessibility, interoperability, and reuse of digital assets (GO FAIR, 2016) and can deliver Analysis-Ready Data (ARD; Dwyer et al., 2018).

Interoperability between digital twins involves adopting standards, data formats, and protocols. While there are numerous ways a system can achieve interoperability, some major ones identified during discussion sessions at the 4th NOAA AI Workshop (https://www.noaa.gov/ai/events/4th-noaa-ai-workshop-2022) include syntactic interoperability, which is the ability to communicate and exchange data through a standard data format and communication protocols. Structural interoperability refers to the different information technology systems and software applications that exchange, interpret, and present data in an understandable and usable way by the recipient system or application. Semantic interoperability is the ability to transfer and interpret meaningful information between digital twins to build knowledge. Other types of interoperability (for example, legal) are also essential and need to be part of the ongoing discussion, but specific actions to address them become more relevant as digital twins reach higher maturity levels.

5.2.1 Major Digital Twin Efforts

There is an explosion of investment in digital twin research in the earth sciences. Here, we summarize some of the larger efforts, their goals, and their approach. Destination Earth (DestinE, https://destination-earth.eu) is building a digital twin of the Earth, "*a highly accurate digital model of the Earth to model, monitor and simulate natural phenomena, hazards and the related human activities.*" DestinE is taking a top-down approach, as in they are planning the complete system from the start and building components. For example, DestinE is investing in building the core platform, infrastructure, and functions. Another European effort, the Digital Twins of the Ocean (DITTO, https://ditto-oceandecade.org), seeks to "*enable users to address "what if" questions based on shared and relevant data, models, and knowledge*" (Bahurel et al., 2023). Like NESDIS' approach, DITTO aims to advance specific use case prototypes within the first two years.

NASA's Advanced Information Systems Technology (AIST, https://esto.nasa.gov/earth-system-digital-twin) program's Earth System Digital Twin to "*integrating diverse Earth and human activity models, continuous observations, and information system capabilities to provide unified, comprehensive representations and predictions that can be utilized for monitoring as well as for developing actionable information and supporting decision making.*" DestinE and NASA seek to integrate climate and human activity models and are primarily (but not exclusively) focused on research goals. NASA is adopting a bottom-up approach to its ESDT by investing in relevant technologies along thematic lines and building up its capabilities over time (Le Moigne, 2022; Le Moigne and Smith, 2022). While the differences are subtle, all efforts require close coordination; otherwise, their development trajectories may eventually diverge.

Despite these differences, these efforts have significant potential to share resources, particularly those looking to develop from the bottom up. For example, models and processes used by the EO-DT for data fusion and EO-DT data can be incorporated into the ESDT to make socioeconomic predictions. Interoperability in digital twins is an achievable goal. Digital twins are essentially software platforms, and software frequently integrates with other software. Most importantly, there is consensus in the community around interoperability and ample opportunities to collaborate.

By attending conferences, workshops, and meetings with leaders in the digital twin community, we make the following recommendations to encourage interoperability if NESDIS chooses to build an operational EO-DT. We recommend funding an integration effort with another digital twin at a similar maturity level (point-to-point integrations). As major top-down digital twin projects mature, a “middleware” solution may establish the standards for other digital twins to follow. DestinE’s core platform or NASA’s IDEAS framework are viable candidates once they reach a higher maturity level. However, a focused twin-to-twin effort can provide early lessons learned for future, larger efforts.

Like our project, we recommend starting small, such as developing APIs to exchange data between digital twins. Then, we recommend building up processes and developing the ontological interoperability framework.

5.2.2 Connecting with a Sibling Digital Twin

Our recommendations assume that NOAA builds an operational system based on or like our recommendations in **Section 5.1**. While building the operational EO-DT, we recommend that NESDIS identify a partner digital twin effort to work towards interoperability actively. Several ESDT-related projects are mature or will be completed within a few years. Below are some existing candidate projects that NESDIS could closely work with:

- The CNES’ FloodDAM-DT project (https://www.spaceclimateobservatory.org/flooddam-dt) plans to have an end-to-end demonstration by June 2024. FloodDAM is an automated service to detect, monitor, and assess global flood events using ML and computational fluid dynamics. The team has partnered with JPL to use their IDEAS (Integrated Digital Earth Analysis System) platform and has reported successful data and process transfer (Huang et al., 2022; Huang et al, 2023).
- Pixels for Public Health (https://pixels-for-public-health-digital-twin-odu-gis.hub.arcgis.com/) project focuses on predicting coastal hazards and human health in Hampton Roads area in Virginia, USA. Their recent work successfully merges various measurements from satellites, models, and in-situ observations. The team has an open data portal and several available geospatial models (Allen et al., 2023).
- The European Commission’s Horizon 2020 Research and Innovation program effort funds Project Iliad (https://www.ocean-twin.eu/digital-twins). Project Iliad has a mature information model for their ocean digital twin (Palma, 2023).

The partnership is intended to understand the technology needed to achieve interoperability rapidly; it is less important that the sibling effort meet a critical NESDIS use case or need. The most fruitful partnership will be with a group actively building a prototype or a mature end-to-end system for a use case. Then, the two groups can focus on building an ecosystem of tools that leverage both investments. DestinE may have partnership opportunities, but according to their project timeline (DestinE, 2023), they plan to incorporate other digital twins around 2027.

5.2.3 Recommended Steps for Interoperability

After identifying a sibling digital twin partnership, we recommend that the two efforts make small but necessary steps toward interoperability. Their efforts can be a focal point of digital twin community conversation and a path for developing the architecture, data formats, software tools, and methodologies for interoperability between other digital twins.

If an operational EO-DT is built, we recommend the following “action items” to kick-start the process of interoperability (**Figure 5.2**):

1. Exchange data between the EO-DT and the sibling DT

2. Run processes between the EO-DT and the sibling DT
3. Run models between EO-DT and the sibling DT

We recommend that these processes be fully automated, as in, there is no help from a human computer operator.

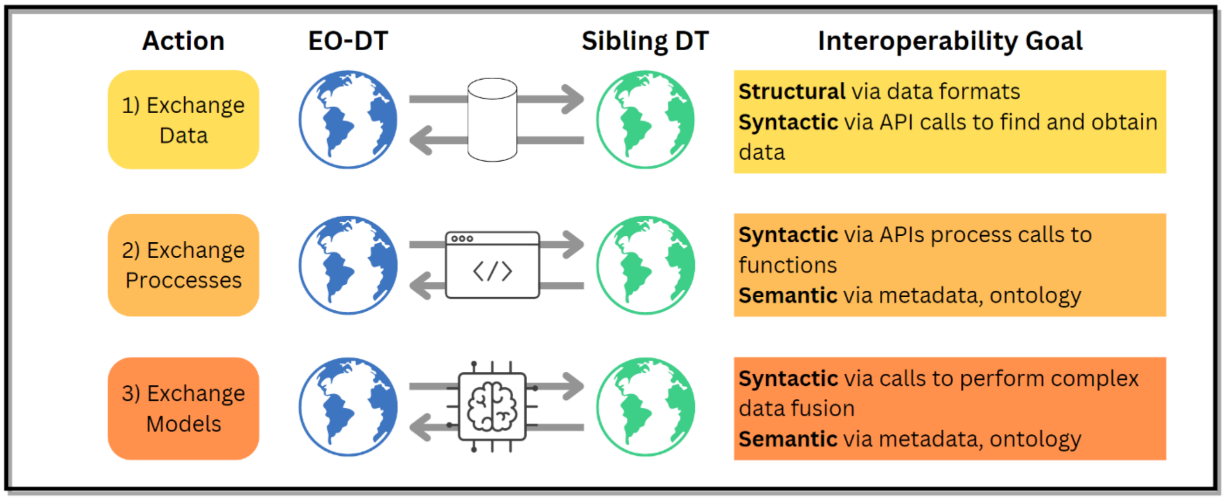

Action
EO-DT
Sibling DT
Interoperability Goal
1) Exchange Data
Structural via data formats
Syntactic via API calls to find and obtain data
2) Exchange Proccesses
Syntactic via APIs process calls to functions
Semantic via metadata, ontology
3) Exchange Models
Syntactic via calls to perform complex data fusion
Semantic via metadata, ontology

Figure 5.2 Three milestones to better understand structural, syntactic, and semantic interoperability between digital twins. After partnering with a sibling effort, we recommend that NESDIS work toward exchanging data, processes, and models.

As a first milestone, we recommend NESDIS begin by establishing data exchange with their sibling digital twin effort. To do so, the EO-DT and sibling DT would need to jointly develop APIs to search for specific data and download it into the other digital twin's system. The STC EO-DT already has an API interface that can serve as the starting point for the exchange. In addition to developing the infrastructure to query and download the data, the two groups can begin to develop an ontological system and identify necessary metadata. These early conversations can help pave the way for a future knowledge graph.

A second milestone is the exchange of processes. We recommend that an operational EO-DT have a function catalog to complete this milestone. The function catalog would consist of containerized code with common data transformations, starting with data fusion operations (e.g., various regridding schemes, ML-based gap filling, and data combination). These processes can be extended to encompass processes that prepare data for assimilation, analysis, and decision support. If successful, the sibling DT effort should be able to download processed data to prepare the data for ingestion into the other digital twin's system and vice versa. The key to success in this second milestone is (1) early identification of the sibling efforts' processing needs and (2) a high degree of containerization of processes in the EO-DT. Our proposed operational EO-DT cloud diagram in **Figure 5.1** explicitly containerizes data fusion processes to be easily updated.

A third milestone is running models on another digital twin. Running models is more challenging than running a single process because more elements are involved, including data exchange and additional processing. For example, let us suppose the sibling digital twin has a microwave sounder that MIIDAPS-AI could use to predict the 850 hPa air temperature. A successful exchange would allow the sibling DT to send its data to the EO-DT, process it with MIIDAPS-AI, and return the result. Another example would be for the EO-DT to send NESDIS data to the sibling EO-DT, have the data assimilated into their prediction model, and download the result.

Milestones 2 and 3 are also points for collaboration with AIST's ESDT effort. Several PIs funded from 2021-2023 are developing useful processes and models that could be incorporated into the EO-DT. For example, AIST invested in updating the GEOS Composition Forecast system (GEOS-CF) for real-time use in a digital twin (PI Keller). AIST also funded a project to harmonize data from heterogeneous sources for deep learning from multi-sensor, multi-temporal data (PI Prasad). The EO-DT could also try to utilize one of the visualization tools, such as the NASA open-source extended reality (PI Grubb). We also recommend further investment in ML-based simulations in the EO-DT, which can help ensure that NESDIS data can be exchanged with an ever-growing number of ML-based models.

Much like the prototype, connecting with a sibling effort is intended to be a pathfinder exercise to evaluate the technology and methods needed to make digital twins interoperable. Given the number of digital twin projects, we foresee a high likelihood of success in leveraging these many projects. Interoperability will benefit the user community, improve data exploitation, and ultimately enhance the value of Earth system digital twins.

5.3 Cost Estimates

One of the challenges of developing cloud systems is that the costs vary depending on the system usage. Cost calculators are invaluable tools designed to provide users with an estimated expense for cloud services based on their projected usage patterns. These calculators, such as the AWS Pricing Calculator, consider several parameters: the type and number of resources (like EC2 instances or Lambda functions), expected data transfer and storage (for services like S3), number of requests (pertinent for services like API Gateway), and more. Customers receive a detailed breakdown of potential costs by inputting these parameters, allowing them to budget effectively. For the services mentioned, costs can vary. EC2 charges are typically based on the type and duration of instances run, while Lambda incurs costs for each execution and the compute time consumed. DynamoDB's pricing considers the amount of read and write capacity units, and S3 costs are contingent on the amount of stored data and requests made. API Gateway generally bills users for the number of API calls made and the amount of data transferred, while CloudFront's pricing considers data transfer and requests, factoring in the geographic region. Utilizing cost calculators helps understand these nuances, enabling businesses to make informed decisions about scaling and optimizing their cloud infrastructure.

5.3.1 Prototype EO-DT

For our demonstration, we built and tested our EO-DT prototype for up to two simultaneous users, 100 user requests per month, and 1 GB per user request. **Figure 5.3** shows a time series of the STC's EO-DT prototype monthly costs for the US East-2 (Ohio) region. This time series illustrates that the most cost-intensive resources are "always on," such as EC2. However, the costs are much more predictable, as our EC2 instances were consistently \$500 a month and another \$400 for additional storage. On the other hand, Lambda was a much smaller fraction of our costs even when invoked thousands of times a month, between \$10-\$100. Because of our data-in-place model, we did not need to duplicate the NODD onto our system, and our S3 costs were relatively low (\$50-\$70/month).

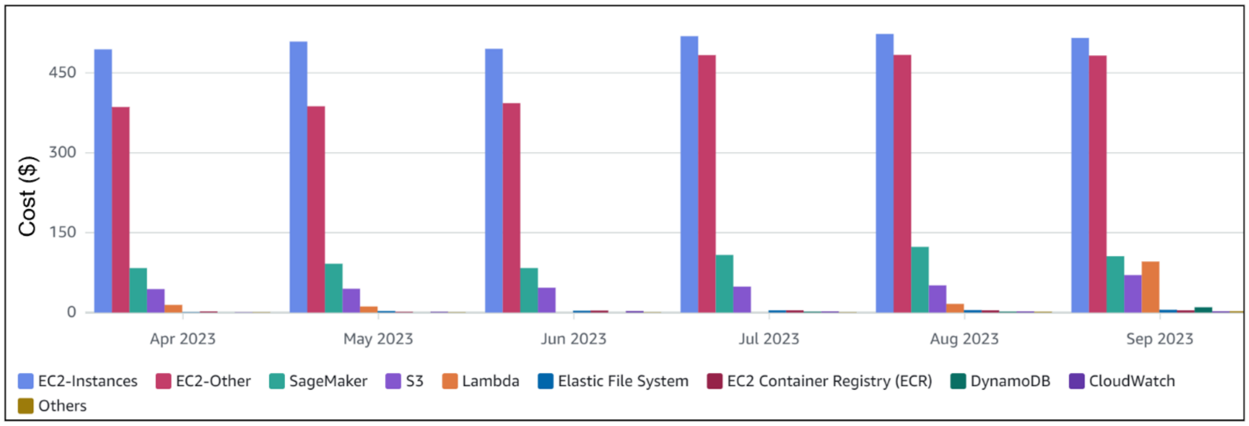


Figure 5.3 A time series of the costs of running our prototype EO-DT. The above does not reflect operational cost, but shows which resources are the most expensive.

We had two EC2 c5.2xlarge instances with 8 vCPUs and 16GB of memory. The cost for a Linux system is $0.36 per hour. One of our EC2s was the development environment and shared by the team for R&D, and the other was the production environment which is used to run more complex code and operations on the EO-DT, such as the regridding code (**Section 2.4**), ML-based anomaly detection code (**Section 3.1**), and MIIDAPS-AI (**Section 1.2.2.2**). Smaller, single-purpose codes were installed in Lambda, such as the ingest code (**Section 2.3.3**), the search code (**Section 2.3.4**), and the geospatial mapping code (**Section 2.3.5**).

As shown in **Figure 1.13**, typical processing required 150MB of memory and our system had 8GB of memory, so we had more than enough memory for the prototype. Thus, we selected an “overpowered” EC2 for our use case. This highlights the benefit of using on-demand resources, which would only run using the needed resources and without excess.

We estimated the cost assuming up to two simultaneous users, 100 user requests per month, and 1 GB per user request. Under these usage conditions, the monthly cost for a prototype EO-DT is estimated to be $1,292 per month (**Table 5.1**). The fees will depend on various factors, including your actual system usage. Any potential taxes are also not included. Our estimate is the runtime cost on AWS and does not include development costs to get the system up and running and ongoing maintenance.

While not a run-time component of the prototype EO-DT, SageMaker was particularly cost-saving because our team could start a GPU-enabled instance for training and testing ML models and shut them down when idle. The g4dn.xlarge (4vCPU, 4GPUs, 16GB mem) is a general-purpose GPU-enabled instance at $0.558 per hour. Assuming one data scientist, an always-on EC2 would cost $1,463 a month (not including storage), whereas SageMaker would cost $117.82 monthly for 160 hours of use (e.g., 8 hours a day for 20 workdays). We utilized Grafana Cloud for the prototype, which was free for up to three users.

Table 5.1 Cost estimates for a prototype EO-DT assuming up to two simultaneous users, 100 user requests per month, and 1 GB per user request.

Monthly cost
$1,292

Detailed Cost

Service	Description	Monthly
Amazon CloudFront	Host for User Interface	$0
AWS Lambda	Search Lambda, Map Lambda, Ingest Lambda	$96
Amazon Simple Notification Service (SNS)	Metadata Ingest	<$1
Amazon Simple Queue Service (SQS)	Metadata Ingest	$2
DynamoDB provisioned capacity	Metadata Catalog/Stores metadata from NOAA OpenData	$10
Amazon EC2	Development EC2s	$989
S3 Standard	Persistent storage for Lambda, Hosting the User Interface, data fusion downloads, map hosting	$70
Amazon CloudWatch	Observability for Grafana	$3
Amazon Elastic Container Registry	Containers for processes	$16
API Gateway	For searching DynamoDB	$0
SageMaker	Development environment for machine learning	$105

5.3.2 Operational EO-DT

We estimated the cost, assuming up to 100 simultaneous users and an average of 1,000 user requests over a month, 10 minutes per request, and 1 GB per request. Under these usage conditions, the cost is estimated to be $4,641 per month ($58,155 per year) for an operational EO-DT (**Table 5.2**). The actual fees will depend on various factors, including the usage of the EO-DT. Our estimate is the runtime cost on AWS and does not include development costs to get the system up and running and ongoing maintenance. Any potential taxes are also not included.

Table 5.2 Cost estimates for an operational EO-DT assuming up to 100 simultaneous users and an average of 1,000 user requests over a month, 10 minutes per request, and 1 GB per request. Estimate available online until Nov 26, 2024, at https://calculator.aws/#/estimate?id=2a3a6d9a142aadb1af918c46c06fdbb90f021454

Estimate summary		
Upfront cost	**Monthly cost**	**Total 12 months cost (Includes upfront costs)**
$2,460	$4,641	$58,155

Detailed Estimate					
Service	**Description**	**Upfront**	**Monthly**	**Annual**	**Configuration Summary**
Amazon CloudFront	Host for User Interface	-	$95	$1,138	Number of requests (HTTPS) (1000000 per month), Data transfer out to internet (1 TB per month), Data transfer out to origin (.33 TB per month)
AWS Lambda	Search Lambda	-	$2	$28	Architecture (x86), Invoke Mode (Buffered), Number of requests (28187 per month), Amount of ephemeral storage allocated (512 MB), Amount of memory allocated (512 MB)
AWS Lambda	Map Lambda	-	$400	$4,802	Architecture (x86), Invoke Mode (Buffered), Number of requests (1000000 per month), Amount of ephemeral storage allocated (512 MB) , Amount of memory allocated (8,192 MB)
AWS Lambda	Ingest Lambda	-	$85	$1,015	Architecture (x86), Invoke Mode (Buffered), Number of requests (1000000 per month), Amount of ephemeral storage allocated (4096 MB), Amount of memory allocated (512 MB)

Amazon Simple Notification Service (SNS)	Metadata Ingest	-	$1	$11	Data transfer Inbound: All other regions (1 TB per month), Data transfer Outbound: Not selected (0 TB per month), Requests (1 million per month), SQS Notifications (1 million per month), The amount of outbound payload data scanned per month (10 GB)
Amazon Simple Queue Service (SQS)	Metadata Ingest	-	$0	$0	Data transfer Inbound: Internet (1 TB per month), Data transfer Outbound: Not selected (0 TB per month), FIFO queue requests (1 million per month)
DynamoDB provisioned capacity	Metadata Catalog/Stores metadata from NOAA OpenData	$2,460	$408	$7,361	Table class (Standard), Average item size (all attributes) (16 KB), Write reserved capacity term (1 year), Read reserved capacity term (1 year), Data storage size (100 GB)
Amazon EC2	Grafana EC2	-	$362	$4,342	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (m6g.4xlarge), Pricing strategy (On-Demand Utilization: 80 %Utilized/Month), Enable monitoring (enabled), Data transfer Inbound: Not selected (0 TB per month), Data transfer Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)
S3 Standard	Persistent storage for Lambda	-	$24	$283	S3 Standard storage (1 TB per month), PUT, COPY, POST, LIST requests to S3 Standard (3000)
S3 Public	Hosting the User Interface, data	-	$116	$1,389	S3 Standard storage (1 TB per month), PUT,

	fusion downloads, map hosting				COPY, POST, LIST requests to S3 Standard (3000) Data transfer Inbound: Internet (1 TB per month), Data transfer Outbound: Internet (1 TB per month)
AWS Fargate	Compute environment for data fusion containers	-	$3,002	$36,027	Operating system (Linux), CPU Architecture (x86), Average duration (10 minutes), Number of tasks or pods (100 per hour), Amount of ephemeral storage allocated for Amazon ECS (100 GB), Amount of memory allocated (16 GB)
Amazon CloudWatch	Observability for Grafana	-	$52	$619	Number of Metrics (includes detailed and custom metrics) (100), GetMetricData: Number of metrics requested (1000), GetMetricWidgetImage: Number of metrics requested (1000), Number of other API requests (1000), Standard Logs: Data Ingested (1 GB), Logs Delivered to CloudWatch Logs: Data Ingested (1 GB), Logs Delivered to S3: Data Ingested (1 GB), Number of Custom/Cross-account events (100), Number of Dashboards (1), Number of Standard Resolution Alarm Metrics (100), Number of High Resolution Alarm Metrics (10), Number of Lambda functions (3), Number of requests per function (100 per hour)
Amazon Elastic Container Registry	Containers for processes (e.g., regridding,	-	$10	$120	Amount of data stored (100 GB per month)

	anomaly detection)				
Step Functions - Standard Workflows	Handler for AWS Batch	-	$9	$108	Workflow requests (1000 per hour), State transitions per workflow (4)
AWS Batch	Handler for processes in ECR using Fargate compute	-	$0	$0	No charge
AWS Web Application Firewall (WAF)	Security for User Interface and API	-	$37	$444	Number of Web Access Control Lists (Web ACLs) utilized (2 per month), Number of Rules added per Web ACL (5 per month), Number of Rule Groups per Web ACL (1 per month), Number of Rules inside each Rule Group (5 per month), Number of Managed Rule Groups per Web ACL (1 per month)
Elastic Load Balancing	Security for Grafana Dashboard	-	$17	$199	Number of Application Load Balancers (1)
API Gateway	Searching DynamoDB, triggering Data Fusion		$2	$24	REST API request units (thousands), Cache memory size (GB) (None), WebSocket message units (thousands), HTTP API requests units (thousands), Average size of each request (10 MB), Average message size (32 KB), Requests (100 per month), Requests (100 per month)
Data Transfer	Transferring data out of EO-DT		$20	$240	Data transfer Inbound: All other regions (1000 TB per month), Data transfer Outbound: All other regions (1000 GB per month), Data transfer Intra-Region: (0 TB per month), Data transfer cost (20)

The most considerable cost is related to Fargate ($36,027/year), the compute engine for the data fusion containers. EKS is a cheaper solution, estimated at $17,059/year for ~ 600 hours per month using an on-demand c5.2xlarge EC2 and one EKS cluster. While the runtime costs may be lower, this EKS solution requires more hands-on management of the EC2 systems by a cloud engineer, whereas Fargate will automatically allocate resources. These two systems are interchangeable in the architecture plan in **Figure 5.1**; one would remove AWS Fargate and replace it with EKS.

The three Lambdas have substantially different costs due to their compute environment's size and frequency of invocation. The Ingest Lambda is run frequently but only needs 512MB of memory, whereas the map lambda uses 8GB because the latter needs to read several large files into memory. If specific processes need high amounts of memory, they may need to be containerized and managed using Fargate.

There are two pricing options available for Amazon DynamoDB, which are on-demand capacity mode and provisioned capacity mode. The provisioned capacity mode is more cost-effective for consistent traffic, which is the case if a steady stream of incoming data is being ingested. This pricing option incurs a partial upfront cost. In an operational system, it may be beneficial to briefly use on-demand capacity mode to establish a baseline and switch to provisional capacity later.

Grafana Cloud is relatively inexpensive (<$200/month) for a limited number of users. For an operational digital twin, we recommend using the self-hosted Grafana option for tighter integration with cloud computing resources, improved performance, and allowing for unlimited users. The minimum computing overhead for a self-hosted Grafana is small (t4g.nano, 512 MB RAM, 1 CPU), which is roughly $1.50 a month, but we recommend an upgraded instance (m6g.4xlarge) which would improve the performance and cost $362 per month.

The final highly variable cost is transferring data from the EO-DT to another system, such as a scientist's working station or another digital twin. Inbound data into the EO-DT is free, but 1000 GB of data transfer out of the EO-DT would cost $20. If interoperable digital twins proliferate and the data transfer rate increases (e.g., from 1 Terabyte to 100 Terabytes), this cost scales linearly from $20 to $2,000 monthly.

5.4 References

Allen, T., Chen, Y.-H., Terry, B., Baptist, J., Steiner, B., Tahvildari, N., et al. (2023). A Digital Twin to Link Flood Models, Sensors, and Earth Observations for Coastal Resilience in Hampton Roads, Virginia USA. In Big Data and AI Technologies for Earth System Digital Twins I. Pasadena, California, USA. Retrieved from https://2023.ieeeigarss.org/view_paper.php?PaperNum=2968

Bahurel, P., Brönner, U., Buttigieg, P.-L., Chai, F., Chassignet, E., Fanjul, E. A., et al. (2023). DITTO Programme White Paper. Retrieved from https://ditto-oceandecade.org/wp-content/uploads/2023/07/DITTO-Whitepaper_FINAL-July-2023.pdf

DestinE. (2023, July 4). Destination Earth Timeline. Retrieved November 24, 2023, from https://digital-strategy.ec.europa.eu/en/policies/destination-earth

Dwyer, J. L., Roy, D. P., Sauer, B., Jenkerson, C. B., Zhang, H. K., & Lymburner, L. (2018). Analysis Ready Data: Enabling Analysis of the Landsat Archive. Remote Sensing, 10(9), 1363. https://doi.org/10.3390/rs10091363

GO FAIR. (2016). FAIR Principles. Retrieved November 24, 2023, from https://www.go-fair.org/fair-principles/

Huang, T., David, C., Oadia, C., Roberts, J. T., Kumar, S. V., Stackhouse, P., et al. (2022). An Earth System Digital Twin for Flood Prediction and Analysis. In IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium (pp. 4735–4738). Kuala Lumpur, Malaysia: IEEE. https://doi.org/10.1109/IGARSS46834.2022.9884830

Huang, T., David, C., Perez, S., Taglialatela, C. et al. (2023). Big Data Smart: Federated Earth System Digital Twins. In Big Data and AI Technologies for Earth System Digital Twins I. Pasadena, California, USA. Retrieved from https://2023.ieeeigarss.org/view_paper.php?PaperNum=2968

Le Moigne, J. (2022). NASA'S Advanced Information Systems Technology (AIST): Combining New Observing Strategies and Analytics Frameworks to Build Earth System Digital Twins. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium* (pp. 4724–4727). Kuala Lumpur, Malaysia: IEEE. https://doi.org/10.1109/IGARSS46834.2022.9883640

Le Moigne, J., & Smith, B. (2022). Earth Systems Digital Twin (ESDT) Workshop Report, 66. Retrieved from https://esto.nasa.gov/files/ESDT_Workshop_Report.pdf.

Palma, R. (2023, November). ILIAD approach for semantic interoperability. Presented at the Standards and Best Practices for Digital Twins. DITTO Summit satellite event. Retrieved from https://www.ocean-twin.eu/event/standards-and-best-practices-for-digital-twins-ditto-summit-satellite-event

6 Recommendations

In the original BAA, NOAA was interested in exploring how digital twins could (1) enhance our ability to process, monitor, quality-control, consolidate, fuse, and assimilate environmental observations; (2) streamline the satellite data ground processing and dissemination to users and applications; and (3) serve as the next generation ground enterprise system in operations. The BAA also sought to identify (4) the benefits of interfacing with the Earth System approach modeling effort in NOAA and (5) the best approaches for achieving an agile, scalable EO-DT. A complete discussion of the BAA goals is available online: https://www.nesdis.noaa.gov/events/digital-twin-earth-observations-eo-dt-using-artificial-intelligence.

Through our demonstration and this study, we found concrete examples of how an EO-DT can enhance the NESDIS data monitoring, processing, delivery, data fusion, and quality control. Using on-demand cloud resources, we found that an EO-DT can cost-effectively scale data processing and dissemination for a next-generation ground system. We identified commercial and open-source tools we recommend and others we do not. We also documented the many lessons learned in the process. Overall, we found that an EO-DT can meet the BAA exploratory requirements and recommend developing an operational EO-DT[4].

Section 1.1.2 identified several questions we sought to answer through this project. Table 6.1 is a high-level summary of our answers and recommendations.

Table 6.1 Summary of questions in Section 1.1.2 and a shorthand reference.

Shorthand	Question	Quick Summary
Cost	How much will it cost to develop a fully capable digital twin?	Prototype: $1,292 per month Operational: $4,641 per month
Tools	What are the best commercial and open-source tools to use?	We recommend several serverless AWS resources, Python scripting and ML libraries, Grafana Dashboards.
ML	Can ML improve the performance of some of the ground system components?	We found ML useful for detecting dataset anomalies in our proof-of-concept study.
Formats	What data formats are optimal within the digital twin?	We do not recommend reformatting to cloud optimized data formats. However, we recommend considering cloud-optimized filesystems.
Access	Can the digital twin improve data access to end users?	We found that scalable, on-demand AWS resources can provide data in analysis-ready format and reduce data wrangling.

[4] There is no planning for activity beyond the demonstration projects. See Disclaimer on page 4.

Latency	Can the digital twin meet or exceed the latency requirements?	We found that scalable, on-demand AWS resources can quickly deliver data to end users.
Data Fusion	How can the digital twin enable data fusion and ML?	We found that classical methods can be used for gridding operations in scalable, on-demand cloud resources. We found that ML techniques can be used to fill gaps and improve the spatial resolution of satellite datasets, making them more useful for data fusion.

Below is a summary of specific recommendations lessons learned, and where their detailed discussion can be found in the report. We link the recommendation back to the shorthand question notation in **Table 6.1.**

1) Approach to Building an EO-DT
a) **Recommendation:** “Keep It Simple” and build incrementally [Cost]
(**Section 1.1.4**) We incorporated a “Keep It Simple” philosophy on the user interface. While users may need complex features in a fully operational digital twin, we aimed to minimize clutter from the interface. (**Section 5.3.1**) We estimated the cost assuming up to two simultaneous users, 100 user requests per month, and 1 GB per user request. Under these usage conditions, the monthly cost for a prototype EO-DT is estimated to be $1,292 per month. Our estimate is the runtime cost on AWS and does not include development costs to get the system up and running and ongoing maintenance. (**Section 5.3.2**) We estimated the cost, assuming up to 100 simultaneous users and an average of 1,000 user requests over a month, 10 minutes per request, and 1 GB per request. Under these usage conditions, the cost is estimated to be $58,155 per year for an operational EO-DT. Our estimate is the runtime cost on AWS and does not include development costs to get the system up and running and ongoing maintenance.
b) **Lesson Learned**: On-demand resources are operationally robust once installed. However, they have a steep learning curve and their limitations need to be considered. [Tools]
(**Lesson Learned 2c**) See “Consider container size and runtime when choosing Lambda over other containerized services” for an example
c) **Lesson Learned**: TLEs need to be up-to-date otherwise, the file geolocation will incorrectly drift over time [Tools]
(**Section 2.3.3**) An important consideration is that a satellite’s TLE changes over time to reflect satellite orbit changes due to drag. So, the TLE table must be updated once a day to

reflect these changes. We learned this lesson quite painfully when our initial testing did not return results in a specified bounding box when we tested it in March.

Lambda deployment involves zipping the entire contents of a working directory and uploading, so we included a static copy of the TLE in the package. The query results were thousands of miles away from our bounding box by September because we were using an outdated TLE. Because we did not want Lambda to download the current TLE file every time it was invoked (which may lead to our IP being blocked by the host website), we wrote a script to download the TLE once a day to our local S3 and Lambda, then imports it each time it is invoked.

d) **Recommendation**: Geohash improved processing efficiency in catalog at the expense of precision [Tools, Latency]

(**Section 2.3.2**) Using geohash, there is no longer a dependency on data-specific readers like HDF libraries or specialized file readers, making the system more flexible for accommodating new data. A drawback is that the file position is not exact but is the nadir point of the center of the granule. To address this, we designed the search to return more results than needed, which can be further filtered in later processing steps once the file has been opened.

e) **Lesson Learned**: Incorporate flexibility in SNS filters because S3 resources may be reorganized by large data providers [Tools]

(**Section 2.1**) Overall, flexibility, maintainability, and scalability are the key strengths of our architecture. Our system was tested when the JPSS part of the NODD underwent reorganization, and our data paths no longer pointed to the files needed to populate the catalog - as seen in reduced data flow into our catalog database. Fortunately, our SNS filters are easy to update, and when we saw the reduced data flow, we could make changes. The system remained online through this episode, albeit with a short data outage for some products. Some future improvements would include incorporating containerization, adding security measures, and load balancing so the system can handle more simultaneous users

2) Evaluation of Commercial Software and Services

a) **Recommendation**: AWS TwinMaker not recommended for Earth digital twins (at this time) [Tools]

(**Section 2.1**) During the project, we also evaluated several services that we did not ultimately decide to deploy in the EO-DT. For instance, we explored using TwinMaker, Amazon's new digital twin service that provides a framework to integrate data streams from IoT sensors. An appealing characteristic of TwinMaker was that it fully managed the messaging and data flow within the system, and we could expand to accommodate new sensors and datasets and remove components if features were retired. However, upon testing, TwinMaker did not easily ingest NESDIS data sources, which included satellite sensor data, retrieval products, dataset production data, and user inputs. Instead, we found that a combination of SNS/SQS and Lambda functions carried out many of the same functions as TwinMaker and were able to leverage Python packages that can read geospatial data formats.

b) **Recommendation**: AWS Fargate useful for a robust operational EO-DT (if built) but not for rapid prototype development [Tools, Latency, Cost]

(**Section 2.1**) We also considered using containers and resources such as AWS Fargate, a serverless compute engine for containers that work with Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Fargate would have allowed us to deploy containerized applications without managing the EC2 instances. We ultimately decided that using EC2 with Python virtual environments was sufficient for the demonstration. However, if one is built, we recommend Fargate or another container orchestration approach for an operational EO-DT.

(**Section 5.3.2**) The most considerable cost is related to Fargate (~$3,000/month), the compute engine for data fusion containers. EKS is a cheaper solution. While the runtime costs may be lower, EKS requires more hands-on management of the EC2 systems by a cloud engineer, whereas Fargate will automatically allocate resources.

c) **Lesson Learned**: Consider container size and runtime when choosing Lambda over other on-demand services [Tools]

(**Section 2.1**) Early in the project, we considered using Lambda to perform data fusion. However, Lambda has strict size limitations for the entire package (1GB) and strict processing time limits (<15 minutes to run). The data fusion processing time may exceed the time limit for large requests. Instead, we installed the code on an EC2. For a fully operational EO-DT, we recommend creating a docker image and deploying using a service like Amazon Elastic Container Service (ECS) to fully scalable the resources. Once the file is regridded, it is saved to another S3 directory linked to the map Lambda. If a new file is present, the map Lambda triggers. Lambda then reads the file and displays the regridded data on the leaflet map, which is then opened as a new tab on the user's browser.

d) **Lesson Learned**: Recommend DynamoDB, but utilize Z-order indexing in operational system instead of the GSI approach in the prototype [Tools, Latency]

(**Section 2.3.1**) The fastest searches will query based on the partition and sort keys; querying the other fields is significantly slower. We needed users to be able to search the table based on multiple parameters: time, location, and product. A solution is to create Global Secondary Indexes (GSI), which copies the main table with different partition combinations and sort keys. GSI makes a copy of the main table, once for each different sort key.

(**Section 2.3.1**) Toward the end of the project, we showed our approach to a DynamoDB subject matter expert who proposed an alternative, faster, and more cost-effective approach using Z-Order Indexing. This approach would require us to create a new column with a unique value by combining multiple columns' values and setting that field as the sort key. Then, we can use a non-unique field as the partition key. This approach would be more effective because it would require fewer queries and copies of the main table.

e) **Recommendation:** Use SageMaker for ML development teams [Tools, Cost]

(**Section 5.3.1**) While not a run-time component of the prototype EO-DT, SageMaker was particularly cost-saving because our team could start a GPU-enabled instance for training and testing machine learning models and shut them down when idle.

(**Lesson Learned 4d**) Also see "Need on-demand access to GPUs for some of the newest techniques" for examples

3) Evaluation of Open-Source Software and Services

a) **Recommendation:** Grafana suitable platform for dashboard [Tools]

(**Section 2.5**) Grafana met our selection criteria for an EO-DT dashboard, which included (1) how well the platform interfaced with our EO-DT system and data streams, (2) if the platform produced a variety of customizable visualizations, including from geospatial data, and (3) if the platform operating costs were within our budget. Since our project focused on open-source solutions, we did not evaluate commercial platforms, like Tableau or Microsoft PowerBI.

b) **Recommendation:** Used Grafana Cloud for demonstration (fully managed service by Grafana) but recommend migrating to AWS system to support more users and have complete ownership over the service. [Tools, Cost]

(**Section 5.3.2**) While we chose to use the fully managed Grafana Cloud, Grafana is an open-source dashboard that can be self-hosted for free. An advantage of Grafana Cloud is that the developer only must focus on setting up the dashboard and metrics and less on the infrastructure to support it. Grafana Cloud is relatively inexpensive for a limited number of users. For a full digital twin, we recommend using the self-hosted Grafana option for tighter integration with cloud computing resources, cheaper hosting costs, and allowing for additional users.

4) Machine Learning Improvements to Ground System

a) **Recommendation:** Use PyTorch machine learning libraries and packages [Tools, ML, Data Fusion, Access]

(**Section 3.1.1**) Early in the project, we had to decide which ML Python package to use for training. TensorFlow and PyTorch are both free, easily accessible, open-source software libraries. TensorFlow is older and thus has more online documentation and community support. PyTorch is a popular tool because of its user-friendly interface and flexible design. Despite PyTorch's comparatively smaller ecosystem and restricted multi-GPU support, it was designated as the chosen framework for DTAD. This decision was due to several factors: PyTorch's seamless integration with Python and its more intuitive API, the provision of dynamic computational graphs that render it particularly conducive for intuitive processing and experimental endeavors, its robust foothold within academic circles, and its native support for the Open Neural Network (ONNX) format.

b) **Recommendation**: ResNet-18 performed better than custom-tuned CNN for anomaly detection [Tools, ML]

(**Section 3.1.1**) After segregating the images into their respective directories, labeled as "valid" or "invalid," we initiate model training. We initially chose to use CNN with a custom-trained model. However, the model's accuracy did not achieve the desired benchmarks. The most proficient model trained under this paradigm attained an accuracy of 75% in the binary classification of satellite images, which was lower than expected, given how large some of the anomalies were.

c) **Recommendation**: Classical methods adequate for regridding, but continue to explore ConvLSTM and ESRGAN techniques to better exploit more satellite observations [Data Fusion, Access]

(Section 2.4.1) We developed a fast, simple regridding method that is agnostic of the data source and can be applied to all NESDIS datasets without re-training. We successfully tested and installed this code into our EO-DT prototype, and it successfully transformed satellite datasets from an irregular grid to a regular grid at a user's requested spacing. The code processed a data granule on one CPU core within 30 seconds to 3 minutes for a 50 km and 750 m VIIRS granule, respectively. Processing speeds can be improved by utilizing multithreading and containerization.

(Section 3.2.1) We show an AOD prediction of ConvLSTM for a cloud-free scene. We used two hours (six, twenty-minute timesteps) AOD images to predict the next several timesteps. The observed AOD and the predicted results do an excellent job capturing areas of higher AOD. The model can make predictions further in time, but the results were unrealistic. Given that we are applying the ConvLSTM model to fill in clouds, not predict AOD, it is more critical that the next time step (t+1) agrees with the observed values.

(Section 3.2.2) We show an example of the pre-trained ESRGAN sharpened resolution of a sample GOES-16 AOD image. In the native resolution image, GOES-16 AOD values are coarse when zooming into the focus region over Florida. The features are sharpened after processing with the ESRGAN model. The results show good visual agreement but need further validation with VIIRS AOD and AERONET before adopting in an EO-DT. In the pre-trained model output, the pixel shapes take on a granular geometry, which may be controlled by developing a custom-trained model. Overall, ESRGAN shows promise for improving the spatial resolution of earth observations, especially for products available on different platforms.

d) **Lesson Learned**: Training datasets remain a challenge and are very time consuming to construct [ML, Data Fusion, Access]

(**Section 3.2.1**) We had to construct a training dataset that reduced the full disk image (5424x5424 pixels) into small samples that were 64x64 pixels. Each smaller image consists of 12 timestep observations of AOD. We needed all the images to be nearly cloud-free, which was challenging given that the mean cloud fraction of the earth is 60% (King et al., 2023). Thus, building an extended training dataset was time-consuming.

Using a year of data, we created 800 training samples (of 12 timesteps each) and validated/tested with 300 samples to refine the model. GOES-16 AOD is generated every 10 minutes, and we initially used all available observations to train out data. However, scene-to-scene changes in AOD can be small, so the ConvLSTM predictions were unrealistic. Instead, we switched to 20-minute timesteps and saw more realistic propagation of AOD plumes.

e) **Lesson Learned**: Models perform better during the validation period than when implemented in operations [ML, Data Fusion]

(**Section 3.1.4**) It is evident that models tend to perform better in offline testing and training than when running in real-time. This underscores that even with long training periods, there is inherent unpredictability in real-time data, and the models may confuse natural phenomena as an anomaly and vice versa.

f) **Lesson Learned**: Real training data better than synthetic [ML, Data Fusion]

(**Section 3.1.4**) Our analysis suggests that training ML models with natural GOES-16 anomalies yield better results than synthetic data with the VIIRS data. For machine learning-based anomaly detection to advance as a field, we recommend that NESDIS science teams help create large, labeled repositories of training data for the community to explore.

g) **Lesson Learned**: Need access to GPU-enabled systems for some of the newest techniques [ML, Data Fusion, Access]

(**Section 2.4.3**) While slower, many classical regridding techniques are less computationally intensive than their ML counterparts, as some ML models require GPU-equipped instances for training.

(**Section 3.1.1**) PyTorch is a popular tool because of its user-friendly interface and flexible design. Despite PyTorch's comparatively smaller ecosystem and restricted multi-GPU support, it was designated as the chosen framework for DTAD.

(**Section 3.2**) The advantage of ML systems is that they can often outperform classic techniques regarding computational speed, taking advantage of GPU architectures.

(**Section 3.2**) ESRGAN requires extensive training datasets and GPUs to process the data. As a result, we used a pre-trained model to evaluate the off-the-shelf version and see if it was feasible for an EO-DT.

(**Lesson Learned 2e**) Also see SageMaker for a recommended on-demand solution.

h) **Lesson Learned**: Still relying on offline dataset pre-processing and storage for training models, not fully implementing into memory [Formats, ML, Data Fusion]

(**Lesson Learned 5a**) See cloud-optimized file system recommendations

5) Data Formats

a) **Recommendation:** Keep data in native format (for now), but update the filesystems to be cloud-optimized [Formats, Access]

(**Section 2.4.3**) We recommend adopting a cloud-optimized file system to expedite the processes of opening, downloading, and further processing the files.

(**Section 4.2**) While newer formats have cloud-specific advantages, legacy formats like netCDF and HDF remain deeply entrenched in the community's practices and toolchains. Both Unidata and the HDF group, which manage the two filesystems, are actively studying and implementing solutions to libraries for these formats to reinforce their resilience and adaptability as systems evolve to the cloud. Considering this, it is pragmatic for institutions like NOAA and NASA to continue leveraging these formats. Transitioning to a new format would not only entail considerable resources for converting large archives but may also introduce complexities and unforeseen challenges in the community. Staying the course with netCDF and HDF ensures stability, continuity, and the ability to benefit from ongoing innovations in their ecosystems.

b) **Lesson Learned:** Still need custom readers to open/read data [Formats, Access]

(**Section 2.3.4**) Note that in a fully functional EO-DT, we envision the user will be able to select which files they wish to display. It is here that we open the files themselves for the first time. Each product requires a unique reader to parse the file contents. We only extracted one variable for simplicity, even though numerous fields are in the files. In the future, we imagine the end user could use the file contents as a search parameter. Because the quick look is intended to display data rapidly, we downsampled the data at a ratio of 10:1 for some of the larger files, such as those from the full disk ABI.

(**Section 2.4.3**) Our analysis identified several bottlenecks that made data processing and usage difficult. A primary challenge arose from the unpacking and utilization of typical NetCDF4 files, which contain numerous variables helpful to an algorithm developer but may not apply to the average user. Unfortunately, the user must download the entire file to access only a small part of its contents. For instance, we only needed to access four variables (AOD550, QCAll, Latitude, and Longitude) out of the 21 geospatial variables in the VIIRS AOD.

7 Appendix

7.1 Accessing Digital Resources

All technological assets and documentation for the STC EO-DT prototype and this study are available online: https://gitlab.com/stc-ai.

7.2 Meeting Summaries

7.2.1 Biweekly Meeting Discussion

- October 2022
  - Checking in with the timeline, STC is currently on schedule and focusing on setting up the EO-DT infrastructure on AWS and scheduling meetings with the NASA DT team.
  - STC completed training on AWS TwinMaker and scheduled a meeting with the TwinMaker lead application engineer to discuss the project.
  - STC picked an example use case around fire weather to work towards when building the EO-DT. In the use case, the user orders and combines multiple datasets relevant to assessing fire weather, air quality, and the presence of active fires. This could utilize the active fire, AOD, NUCAPS, and GFS datasets.
  - STC discussed data access pathways with the DEEVA team. Options included private FTP feed for the project or public access via AWS, CLASS. DEEVA team recommends using the public channels. The DEEVA team will follow up on datasets not in STAR's portfolio (e.g., SUVI).
  - STC requests data readers where available, and the DEEVA team will provide them by the next biweekly.
  - STC states that the underlying philosophy is to use off-the-shelf tools, including existing NOAA resources.
- November 2022
  - STC created an architecture map for the DT prototype. Collaborating with the AWS team to validate the system and get upgrade recommendations.
  - One challenge of implementing a DT is the data bottleneck. Getting data from CLASS can be slow, which could impact the processing times of implementing a full DT.
  - STC wants to keep the user experience in mind, STC team is concurrently examining the end use cases (from both user and management) with AWS setup and development.
  - STC and DEEVA discussed incorporating the MIIDAPS-AI into the DT at a later time. Not currently routinely run at STAR, but could be run on STC system. Can schedule a follow up TIM at a later time.
- January 2023
  - Eric Maddy noted that when benchmarking, test speed performance is better when batching multiple files along with single-file tests. Performance can decline exponentially (for example) with increasing file sizes.
- February 2023
  - Scheduled midterm demo for April 12, 2023. Planned for 1.5 hours (*post mortem: we recommend in the future we schedule for 2 hours*), each team member will talk for 10-15 mins. Will discuss our approach with slides and show a live, hands-on demo.

- Discussed the value of super-resolution method for achieving fine scales in the EODT datasets. Can train ABI data to match VIIRS resolution.
  - Eric Maddy: Planning to use SR or GAN? Yi - we are evaluating both.
  - Beau: Will there be one model or multiple models, depending on the resolution? Yi - anticipate training one model/resolution scale.
  - Ramesh: Will you consider datasets other than AOD?
- Ramesh: How are we implementing agile in this project? Through small, incremental installations into the digital twin, utilizing CI/CD in GitLab. Stand-up meetings sprint into our workflow. Feedback from customers (like biweekly meetings) is also part of the agile process.
- Sid asked if multiple DynamoDBs indicated multiple digital twins. No, to the user, they only see a single database. If implemented, this would be a backend solution, not one transparent to the user.
- Discussed different gridding options for queries, such as h3 and geohash. Beau asked if this was for the data fusion step. STC clarified that this was just a query-only scheme and it was intended to optimize the system.
- Sid shared some of his vision:
  - He is most interested in a flexible system where ML components could be swapped. Possibly one ML model per sensor/product.
  - AI models do not have to be 100% accurate, 60% is acceptable. The models are placeholders for a full-blown EODT if implemented by NOAA. The models can be fine-tuned later.
  - Very interested in the infrastructure and feasibility of meeting requirements. EODT could become the next-generation architecture and possibly replace the existing ground system. Looking for possible improvements to the value chain.
  - Considers success/wins:
    - Demo with a hook into UFS/GFS
    - Couples with another digital twin
  - Wants STC team to stay in the research loop/NASA activities
    - NASA organized around science themes interfaced with DTs that are part of a federation
    - Engage in various activities, e.g., NAS (attended), DestinationEarth meetings

- March 2023
  - Discussed AGU session on digital twins - followed up with NASA. They already planned to organize a session.
  - Ramesh would look into an AMS session, and NOAA would organize
    - Discussed Sid's abstract submission to EUMETSAT 2023, "Informing NOAA's Next-Generation Space and Ground Architectures: Example Concepts of Hyperspectral Microwave Sensor and Earth System Digital Twin"
    - Preparing for Midterm Review on April 12, 2023
- April 2023
  - Midterm review and retrospective meeting
- May 2023

    - Clarified that the final report is intended to be the roadmap to building a digital twin.
    - Ramesh pointed out that it was not clear that we are not moving any data around in our plan (data are “in place”). This is advantageous from a file storage perspective. STC should highlight this more.
    - Ryan B. said our plans make sense, and vision is coming together.
    - Ramesh pointed out that the Grafana cloud is appropriate for the demo, but NOAA would likely need self-managed Grafana for a full EODT for additional security. STC confirmed it will include this info in the final report.
    - The NOAA team is interested in on-prem versus cloud service costs demo versus total system costs.
    - Ramesh recommends we check back on the STC proposal, ensuring all items are discussed. If there are unexpected challenges or changes to the plan, they are interested in the why and the reasons for the changes.
    - Had a side discussion on the definition of digital twin: often, the first thing discussed in a meeting, there are conflicting definitions. Fundamental similarities are decision support and making sense of a “bewildering” amount of data.
    -
- June 2023
    - STC is adding three additional months to Analytics and ML augmentation of the EO-DT for anomalies/fusion knowledge graph study.
    - The anomaly study is complete, and there is additional time to study other sensors (VIIRS, CrIS).
    - Previously concerned about S3 reorganization, NODD reorganization is in progress, so some of the ingest links have to be updated or may break in the future.
    - Ramesh and Beau want to space out the three reviews in September/October/ STC is working to schedule a meeting room for a hybrid format.
    - Ramesh and Beau want to review abstracts before submission to AMS/AGU
    - Ramesh requested a draft of the final report before the final demonstration. Noted that the final report is intended to be a road map to building a digital twin, want to know if NOAA’s needs are met through a digital, the path forward, and the dead ends.
    - Ryan Berkheimer discussed interest in learning about the value that knowledge graphs may have in an operational paradigm. Knowledge graphs are a way to unlock the metadata within NESDIS data.
- July 2023
    - STC showed results from backfilling data. Eric Maddy suggested that we have some QC to show the age of the pixel. STC implemented this already but did not include it in the presentation (will show at another meeting).
    - Ryan Berkheimer discussed NCEI's activity in using knowledge graphs and shared a helpful report on cloud-optimized file formats.
    - Ramesh encouraged us to submit AMS abstract to EODT sessions
    - Ramesh wanted clarification on the current schedule, as many items have an Aug 31 due date. STC stated that some tasks (the data fusion step) were

bigger for the demo while others (knowledge graphs, file formats) were smaller and intended for the final report. The data query task for the demo is nearly complete. STC has no concerns about the current schedule.
    - STC attended the digital twin meeting/NASA town hall at the IGARSS 2023 meeting.
- August 2023
    - Provided an overview of remaining tasks and their status
    - Shows partial demonstration of the user interface for EO-DT to Eric Maddy
- September 2023
    - Final demonstration
- October and November 2023
    - Technical report preparation, review, and submission

7.2.2 Kickoff Meeting Discussion

- STC team's overarching goal is to produce a study with recommendations on building a fully operational EODT. The STC team will accomplish this by building a prototype.
- The NOAA team requests EODT demonstrations, and the STC team will provide two, one at the midpoint and again at the final meeting.
- Sid emphasizes the value of documenting sister DT efforts at NASA/Destination Earth to ensure an operational NOAA EODT is interoperable with those efforts.
- Sid reiterates that processing time requirements are not fixed but wants to ensure the system is fast (in contrast, data can take hours and days to process in the current ground system).
- Sid requests that the STC team consider adding NWP to the data fusion portfolio in the EO-DT to increase gap-filling/augment satellite data products in the product catalog; STC team agrees.
- The STC team emphasizes that risk is mitigated via good communication with the customer and leveraging existing services, technologies, and peer-reviewed methodologies.
- The DEEVA team within NOAA can provide sample files and readers for various data formats to assist the STC team.

Recommendations and Action Items:

- The NOAA team asks what kinds of events will constitute an anomaly. STC team will use both injected noise and simulated anomalies that have happened before. Will work with Flavio's team for these tests.
- In addition to working with Jacqueline, STC will establish a POC within the European digital twin project.
- Sid recommends that the final report include findings on whether (1) NESDIS's goals can be achieved via an EODT and (2) recommendations of what needs to be done to scale up and allow NESDIS to go from prototype to full operational model.
- Sid recommends that the web portal incorporate a data visualization component.
- Beau requests monthly reporting to help keep track of progress during bi-weekly agile meetings.

7.2.3 Midterm Meeting Discussion

- Slide 7: Why is Grafana outside of AWS?
    - STC uses the Grafana cloud, but Grafana can be deployed within AWS.

- Slide 9: Would like STC to summarize challenges/solutions in the final report, and how issues would play out in a fully deployed EO-DT
- Slide 15: Some clarification about the data ingest step:
  - Catalogs the data in table on slide 14
  - Ingest assumes data integrity, paired with anomaly detection to explore how AI can enhance NESDIS processing system.
  - The focus is searchability. Data are left in their native formats
  - Data ingest does not re-grid to a standard grid, which is completed in the data fusion step (slides 35-44)
- Slide 23: What is the computational cost of running the model?
  - Training: 2-3 hours/~4,000 files, running: 10 seconds/file
- Slide 23: Does it say where/why the image fails? Would like to see STC continue to explore the path to trust.
  - CNN does not do that, but it is something STC will explore in the future, either another technique or by pairing with an explainable AI method
- Slide 24: How does the confusion matrix stack up against what forecasters see?
  - Seems reasonable from STC experience. STC can follow up with NWS users or validation teams at NOAA
- Slide 24: GOES-16/-17/-18: How did the GOES-17 ABI anomalies (which were obvious) influence detection in GOES-16/-18?
  - 60% of the training data was GOES-17 ABI, which was helpful because it produced many “invalid” images. All satellites were used for training/test.
- Slide 24: How is it different if you remove the word “digital twin” from the workflow?
  - The ML-approach in the anomaly detection step is not different but is tailored to the EO-DT problem at hand/adapted for a specific use case
    - The definition of a digital twin and what a digital twin should do varies within the earth sciences community.
  - What is the core component/heart of the EO-DT? Would like this to be emphasized in the final demo
    - The data fusion step combines the NESDIS assets. That is the core engine of the digital twin (slides 35-44)
  - How can an EO-DT predict and replicate Earth processes? What is quality control?
    - STC was focused on enhancing the user experience and modeling the ground system (not the earth system). BAA did not call explicitly for an Earth System Digital Twin (slides 4-5)
    - STC can explore an ESDT through the fire weather/air quality science use case (slide 8 and slide 38) and include a summary of results in final demonstration

7.2.4 Final Review Discussion

Q: Why did you use the NOAA NODD instead of the CLASS archive?

A: NODD is more cloud-friendly/represents a better approximation of the 'future state' archive. Better to develop against the NODD.

Q: In addition to collocating the different types of data to the same temporal and spatial spot, data fusion also should 'fuses' or merges the different sources of the same data into a single value. I get a sense that the merging aspect of the data fusion in not present in this demo. Am I misunderstanding this aspect?

A: For the demonstration, we did not explicitly merge multiple datasets of the same type. Instead, we focused on the pre-requisite steps to combining different types of observations. However, the tools we developed can be updated to combine data in a fully operational digital twin.

Q: Are there any plans to experiment with pixel-level labels via image segmentation, so images that are only partially invalid can be more effectively used?

A: We didn’t explore pixel labels, but the invalid portions could be identified using multi-label classification. The flagged regions could be combined with other ML techniques, like ConvLSTM, to fill in invalid regions.

Q: If NOAA implemented an operational EO-DT, what is the biggest thing to consider if they decide to implement it?

A: We recommend defining the requirements, starting with smaller goals that can be scaled into bigger project needs. [STC will include more recommendations in the final report]

7.3 Acronyms

Below is a list of acronyms that were used in this report.

Acronyms	Definition
ABI	Advanced Baseline Imager
ACF	Analytic Collaborative Framework
AERONET	Aerosol Robotic Network
AGU	American Geophysical Union
AIST	Advanced Information Systems Technology
AMS	American Meteorological Society
AMSU	Advanced Microwave Sounding Unit
AOD	Aerosol Optical Depth
API	Application Programming Interface
ATMS	Advanced Technology Microwave Sounder
AVHRR	Advanced Very High Radiometer
AWIPS	Advanced Weather Interactive Processing System
AWS	Amazon Web Services
AWS	Amazon Web Services
BAA	Broad Agency Announcement
CDN	Content Delivery Network
CI/CD	Continuous Integration/Continuous Deployment
CLASS	Comprehensive Large Array-data Stewardship System
CLI	Command Line Interface
CMR	Common Metadata Repository
CNES	Centre National d'Etudes Spatiales
COG	Cloud-optimized GeoTIFFs
CONOPS	Concept of Operations
CONUS	Continental United States
ConvLSTM	Convolutional Long short-term memory
CrIS	Cross-track Infrared Sounder
DAACS	Distributed Active Archive Centers (DAAC)
DB	Database
DestinE	Destination Earth

DITTO	Digital Twins of the Ocean
DQF	Data Quality Flags
DT	Digital Twin
DTAD	Digital Twin Anomaly Detection
EC2	Elastic Cloud Computing
ECMWF	European Centre for Medium-Range Weather Forecasts
ECS	Elastic Container Service
EKS	Elastic Kubernetes Service
EO-DT	Earth Observing Digital Twin
ESA	European Space Agency
ESA	European Space Agency
ESDT	Earth System Digital Twins
ESRGAN	Super-Resolution Generative Adversarial Networks
ESTO	Earth Science Technology Office
EUMETSAT	European Meteorological Satellites
Geo	Geostationary
GEOS	Goddard Earth Observing System
GEOS-CF	GEOS Composition Forecast system
GOES	Geostationary Operational Environmental Satellites
GOES-R	GOES – Series R
GradCAM	Gradient-weighted Class Activation Mapping
GRIB	GRIdded Binary or General Regularly-distributed Information in Binary form
GSI	Global Secondary Indexes
HDF	Hierarchical Data Format
HPC	High Performance Computing
HYSPLIT	Hybrid Single-Particle Lagrangian Integrated Trajectory model
IASI	Infrared Atmospheric Sounding Interferometer
IDEAS	Integrated Digital Earth Analysis System
IoT	Internet of Things
JPSS	Joint Polar Satellite System
JSON	JavaScript Object Notation
JV	Joint Venture

LSTM	Long short-term memory
MetOp	Meteorological Operational Satellite Program of Europe
MIIDAPS-AI	multi-instrument inversion and data assimilation preprocessing system, artificial intelligence version
ML	Machine Learning
NASA	National Aeronautics and Space Administration
NCCF	NESDIS Common Cloud Framework
NCEI	National Centers for Environmental Information
NCEP	National Centers for Environmental Prediction
NESDIS	National Environmental Satellite, Data, and Information Service
NetCDF	Network Common Data Form
NOAA	National Oceanic and Atmospheric Administration
NODD	NOAA Open Data Dissemination
NoSQL	No Structured Query Language
NSOF	NOAA Satellite Operations Facility
NUCAPS	NOAA Unique Combined Atmospheric Processing System
OMPS	Ozone Mapping and Profiler Suite
ONNX	Open Neural Network
OPPA	Office of Projects, Planning, and Analysis
OSPO	Office of Satellite and Product Operations
PDA	Product Distribution and Access
REQ	Request
ResNet-18	Residual Neural Network
RESP	Response
RESTful	REpresentational State Transfer (compliant)
RMSE	Root Mean Square Error
RNN	recurrent neural network
RO	Radio Occultation
S3	Simple Storage Service
SAE	Systems Architecture and Engineering
SNPP	Suomi National Polar-orbiting Partnership
SNS	Simple Notification Service

SQS	Simple Queue Service
SRGAN	Super-Resolution Generative Adversarial Networks
SST	Sea Surface Temperature
STAC	Spatio-Temporal Asset Catalog
STAR	Center for Satellite Applications and Research
TLE	Two Line Elements
UCAR	University Corporation of Atmospheric Research
VIIRS	Visible Infrared Imaging Radiometer Suite
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WBS	Work Breakdown Structure
XAI	Explainable Artificial Intelligence