ABSTRACT

Title of Dissertation: USABLE MACHINE LEARNING FOR REMOTE SENSING DATA

Ivan Zvonkov
Master of Science, 2023

Dissertation Directed by: Professor Abhinav Shrivastava
Department of Computer Science

The desired output for most real-world tasks using machine learning (ML) and remote sensing data is a set of dense predictions that form a predicted map for a geographic region. However, most prior work involving ML and remote sensing follows the traditional practice of reporting metrics on a set of independent, geographically-sparse samples and does not perform dense predictions. To reduce the labor of producing dense prediction maps, we present OpenMapFlow—an open-source python library for rapid map creation with ML and remote sensing data. OpenMapFlow provides 1) a data processing pipeline for users to create labeled datasets for any region, 2) code to train state-of-the-art deep learning models on custom or existing datasets, and 3) a cloud-based architecture to deploy models for efficient map prediction. We demonstrate the benefits of OpenMapFlow through experiments on three binary classification tasks: cropland, crop type (maize), and building mapping. We show that OpenMapFlow drastically reduces the time required for dense prediction compared to traditional workflows. To more broadly understand method adoption we present ML for Remote Sensing Usability Cards to assess usability

for machine learning with remote sensing data and use this framework to conduct a case study of a workflow developed with OpenMapFlow. We hope this library will stimulate novel research in areas such as domain shift, unsupervised learning, and societally-relevant applications and along with the usability framework lessen the barrier to adopting research methods for real-world tasks.

USABLE MACHINE LEARNING FOR REMOTE SENSING DATA

by

Ivan Zvonkov

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2023

Advisory Committee:
Professor Abhinav Shrivastava, Chair/Co-Advisor
Professor Hannah Kerner, Co-Advisor
Professor Hal Daumé III

© Copyright by
Ivan Zvonkov
2023

Acknowledgments

I am deeply grateful to all the individuals who have played a vital role in making my thesis work possible and have contributed to making my time in graduate school a wonderful experience.

First and foremost, I would like to express my gratitude to Professor Hannah Kerner for her guidance and support throughout my Master's degree. Your mentorship has played a vital role in my success as a graduate student, it has been a pleasure to learn from and work with you for the past two years. I would also like to extend my sincere appreciation to Professor Catherine Nakalembe for her invaluable assistance in navigating the challenging field of agricultural analysis using remote sensing data in resource-constrained regions. Her support has been instrumental in shaping my research.

I am grateful to Professor Abhinav Shrivastava for accepting the responsibility of heading my thesis committee and providing insightful feedback and guidance. I would also like to express my gratitude to Professor Hal Daumé III for agreeing to serve on my thesis committee and all members of the committee for taking the time to review my manuscript.

Finally, I would like to thank my close collaborators on the NASA Harvest team for their contributions that have enabled me to undertake this research. Without their cooperation and support, much of this work would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Remote sensing data (also referred to as Earth observation or satellite data) has become an increasingly popular modality for artificial intelligence research. This interest has largely been driven by the opportunities that remote sensing data present for contributing to challenges urgently important to society, such as climate change [1, 2, 3], food security [4, 5, 6], disasters [7], and poverty [8]. In recent years, many new datasets have been made available to facilitate research in ML methods for remote sensing data spanning diverse tasks such as image classification [9], segmentation [7], object detection [10] and diverse application areas such as agriculture [11], disasters [7], and land cover [9, 12]. The growing interest in ML research for remote sensing data is also driven by the challenges presented by its unique characteristics compared to other data modalities. Remote sensing datasets are very high-dimensional and often have spatial, temporal, and spectral dimensions more complex than traditional RGB images or videos. The diversity of instruments used for observing the Earth at different wavelengths, temporal cadences, and spatial resolutions has driven active research in domain adaptation [13, 14], data fusion [15], and other topic areas. Evidence for the growing popularity of remote sensing as a new data modality for ML research can be seen in the growing number of workshops, journals, and initiatives related to ML and remote sensing (e.g., Climate Change AI [2]).

Despite this growing body of novel ML methods for remote sensing, few of them have

seen adoption in consequential application areas. For example the Allen Coral Reef Atlas, the state-of-the-art coral reef mapping platform, utilizes a random forest classifier for classifying geomorphic and benthic classes [16]. Similarly, Global Forest Watch, the state-of-the-art global forest mapping platform, utilizes a decision tree classifier for mapping forest loss [17]. Google's Dynamic World product is one rare example where more expressive deep learning models are used for classifying land cover [18], however, this example is a concerted effort within a technology company and is not an example of an independent remote sensing research group adopting a novel ML method because it is best suited for their purposes.

Adoption of methods is a strong signal of legitimate novelty and innovation and therefore is important to understand. In this thesis we explore the usability challenges of ML for remote sensing data through a hierarchy of needs framework. We use the hierarchy of needs as a lens to explore the challenges of usable ML for remote sensing and draw comparisons to other popular data modalities. We propose ML for Remote Sensing Usability Cards to provide a straightforward way of assessing the usability of a particular ML method for the remote sensing data modality. As an example we fill out a Usability Card for an ML system we have developed for cropland classification.

One reason for the lack of ML method adoption for remote sensing is the absence of geographically-continuous dense predictions in the exploration of many novel ML methods. Traditional ML prediction tasks, such as those using image data or text data, are trained and evaluated on datasets containing samples that are assumed to be independently and identically distributed (i.i.d.). However, for most prediction tasks using remote sensing datasets, the desired output is a set of dense predictions that form a predicted map of a geographic region (Figure 1.1). The input samples used for generating the dense predictions are geographically contiguous and

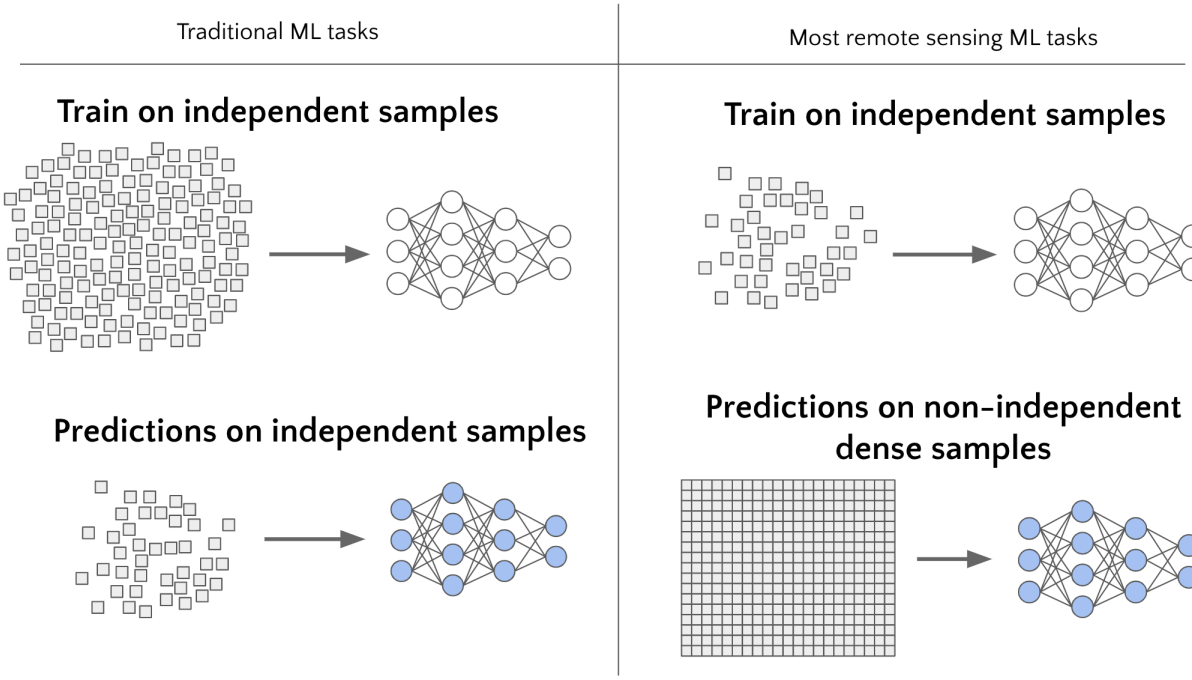

Figure 1.1: Sparse vs dense predictions

violate the i.i.d. assumption. For example, consider the task of crop type classification in which the goal is to predict locations where a specific crop is growing based on satellite observations of a given focus region—e.g., predicting where maize is growing in Kenya [11]. To accomplish this task, a model would be trained using a set of samples from different locations that are sparsely distributed throughout the focus region, then evaluated on a similarly sparsely-distributed test set. The majority of research studies conclude after this evaluation, even though the practical goal of remote sensing prediction tasks is to use the trained model to make dense predictions at every pixel location to form a complete map of the study region. Neglecting dense predictions has several consequences: it 1) makes model adoption significantly more difficult for the larger scientific community and downstream applications, 2) obfuscates potential model failure modes, and 3) precludes the investigation of related foundational research challenges.

In this thesis we also present OpenMapFlow (https://github.com/nasaharvest/openmapflow),

an open-source python library for rapid map creation with ML and remote sensing data. OpenMapFlow will enable researchers to train and deploy ML models with remote sensing data by reducing the effort and cost required to implement dense prediction. Outputs created by OpenMapFlow have already been used in downstream applications in agriculture by the NASA Harvest Program on Agriculture and Food Security. Further, studying dense predictions produced by OpenMapFlow can stimulate future work on foundational research challenges such as dataset shift, evaluation with unlabeled data, active learning, unsupervised learning, and semi-supervised learning.

We hope that the OpenMapFlow library and the Usability Cards will foster scientific exchange between researchers, practitioners, scientists, students, and engineers in AI and remote sensing and enable increased adoption of ML research methods for real-world tasks with social impact.

Chapter 2: Related Work

Despite the proliferation of ML libraries that help build and train novel ML models, there continues to be a gap between research experiments and deployed use of these models. This gap is likely due to the complexity involved in deploying, maintaining, and updating ML systems for solving real-world problems [19]. Research on ML deployment tools, such as OpenMapFlow, is thus critical to help bridge this gap and enable broader adoption of ML tools in society.

An understanding of ML model failure modes is another topic vital to model adoption and improvement [20]. One common method used to understand failure modes is analyzing model errors on a validation set and grouping the errors into distinct categories representing failure modes [21, 22]. This technique is limited to uncovering failure modes within the domain of the validation set. If the validation dataset sufficiently represents the real-world task, the metrics tracked and failure modes discovered using the validation set can be considered representative. However, it is challenging to make representative datasets that are aligned with real-world tasks [23]. Datasets that do not sufficiently represent the real-world task can result in inflated performance metrics and can obfuscate failure modes. OpenMapFlow helps identify failure modes for unlabeled data outside of the data distribution captured by sparse validation sets.

Several platforms and libraries have been developed to facilitate ML research with remote sensing datasets and to evaluate their performance [24], such as TorchGeo [25], Radiant Earth

MLHub [26], and CropHarvest [11]. However, these tools do not provide straightforward support for using trained models to generate maps via dense prediction, which is the goal of OpenMapFlow. Google Earth Engine [27], SEPAL [28], and Descartes Labs [29] platforms do enable map generation via dense prediction using trained machine learning models but have limitations that inhibit their use in the ML research community and adoption by real-world end-users. For example, Google Earth Engine [27] provides comprehensive data export, map storage, model training, and data exploration utilities. However, the availability of modern machine learning models and training algorithms in Google Earth Engine is limited. Google Earth Engine does support trained deep learning models for prediction but only for TensorFlow models deployed using the Google Cloud AI platform. SEPAL [28] is another platform that enables dense predictions to generate maps but is restricted to tree-based models and does not support deep learning. Another limitation of existing tools like Descartes Labs [29] and ArcGIS Pro (www.esri.com/en-us/arcgis/products/arcgis-pro) is that they require users to purchase a paid subscription which is a barrier to easy initial experimentation and adoption by users that lack large budgets. In addition, the aforementioned platforms are primarily designed for facilitating the analysis of remote sensing data broadly. In contrast, OpenMapFlow aims to make remote sensing data a more accessible modality for machine learning research.

In parallel, the machine learning community has begun to adopt guidelines to make machine learning methods broadly more accessible and usable. For example, FAIR Data Principles [30] have been proposed to increase findability, accessibility, interoperability, and reusability of data. Similarly, Datasheets for Datasets [31] have been proposed to help dataset creators reflect on the curation process and help dataset consumers make informed decisions about dataset usage. Beyond datasets, Model Cards [32] have been introduced to describe the intended usage of trained

models. Platforms such as Hugging Face (huggingface.co) have taken a significant step forward in democratizing machine learning models for popular data modalities such as images and text by making model weights and and an inference API accessible. Technology readiness levels for machine learning systems [33] have been proposed to encourage robust iteration and deployment.

Creating accessible and usable machine learning methods for the remote sensing data modality presents challenges not observed in other popular machine learning data modalities. Since these challenges are unique to remote sensing they are not sufficiently addressed in existing general machine learning guidelines and best practices. For example, the practice of sharing model weights has become more common and work such as Model Cards [32] discusses responsible usage of these trained models. In the remote sensing data modality however, sharing model weights is insufficient for usability. Additional attention must be paid to the method of obtaining input data, creating datasets, and using the model for dense predictions. Our proposed Machine Learning for Remote Sensing Usability Cards specifically focus on these challenges to give a comprehensive understanding of ML method adoption potential.

Chapter 3: OpenMapFlow: A Library for Rapid Map Creation

3.1 Library Overview

OpenMapFlow aims to facilitate ML research with remote sensing datasets by providing an accessible and extensible workflow that allows rapid iteration. The complexities of remote sensing data and geospatial map creation can deter new researchers from exploring this important data modality. In the design of OpenMapFlow, special attention was paid to reducing the start-up effort required to develop ML methods for remote sensing tasks by providing:

- a tutorial notebook for demonstrating the intricacies of remote sensing data and model training
- a template project generation function along with three example projects
- a simple method (two lines of code) for downloading existing training and evaluation data
- training and evaluation scripts that can be run on available datasets
- Github action scripts for automated integration testing and model deployment
- Google Colab notebooks for all components (adding data, model training, dense predictions) to remove the need for local environment setup

- A short configuration file to establish the connection between the OpenMapFlow library and the user's own Google Cloud project

The three components of OpenMapFlow—1) data processing pipeline, 2) ML model training, and 3) rapid map creation—can be used independently or end-to-end (Figure 3.1) depending on user needs. The library was designed such that the first iteration cycle of the ML and remote sensing project can be set up quickly.

Ability to evaluate results and iterate quickly is critical to consistent progress in the practice of data science [34]. In projects involving machine learning and remote sensing data, the natural places to iterate are on the labeled dataset, the ML model architecture, the model hyperparameters, and the training setup. A key factor in OpenMapFlow's design was to empower the user to have control over all these inputs and quickly see the result of changing any of them. Thus the library is kept light so users can determine additional dependencies based on their needs. To ensure a tight feedback loop from change to result, a specialized cloud architecture is developed to allow for parallelized predictions at a low cost. Lastly, the auto-generated project functionality is kept simple to allow users the freedom to implement more complex methods.

3.1.1 Project Generation

To start, the user must install the OpenMapFlow package from the Python Package Index (PyPI) and use the command `openmapflow generate` to generate a project structure. The command will prompt the user for project configuration arguments such as project name and Google Cloud Project ID (with several defaults provided). An existing Google Cloud project is a required prerequisite for the end goal of making efficient dense predictions. The data version

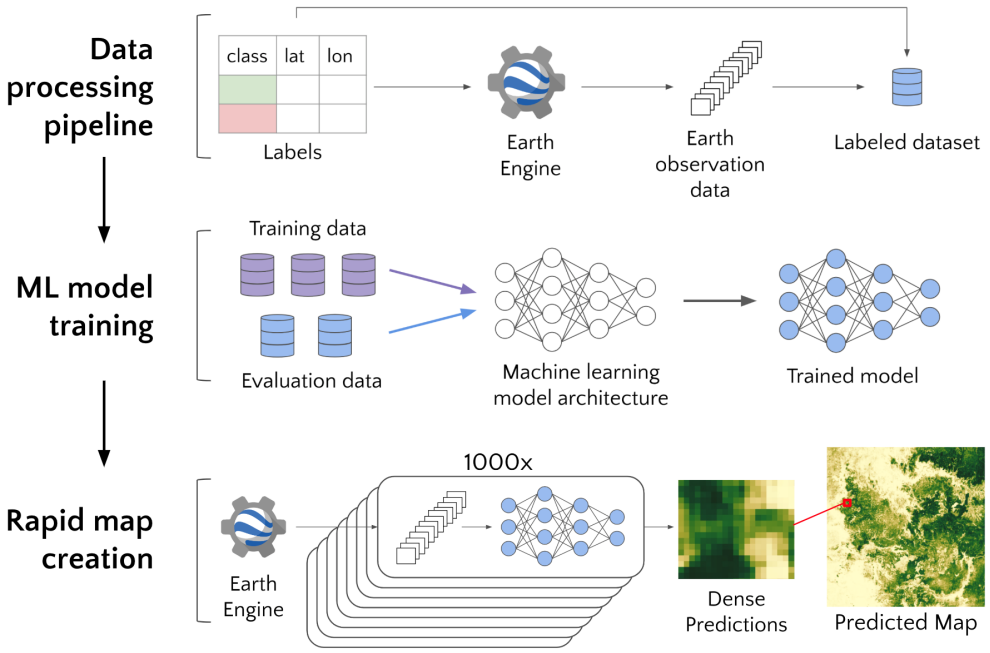

Figure 3.1: OpenMapFlow workflow

control (DVC) (https://dvc.org/) library is used for versioning and storing all data in remote storage. When all configuration is set, a lightweight project structure is generated consisting of:

- `openmapflow.yaml`, a configuration file that saves user inputs from the command `openmapflow generate`
- `datasets.py`, a python file containing a list of new or existing datasets and how they are generated from labels
- `train.py`, a template model training script
- `evaluate.py`, a template model evaluation script
- `.github/workflows/test.yaml`, a Github action script that tests training and evaluation scripts as well as dataset integrity

- `.github/workflows/deploy.yaml`, a Github action script that containerizes and deploys trained models
- `data/raw_labels`, a directory containing user-added labels for creating custom datasets
- `data/datasets`, a directory containing existing ML-ready datasets (consisting of labels and associated remote sensing data)
- `data/models`, a directory containing models trained using project datasets
- `.dvc/`, a folder created by the Data Version Control library for versioning all data

3.1.2 Adding New Data

New training data can be added using the OpenMapFlow remote sensing data processing pipeline, which can be run through a Google Colab notebook or using the command line interface. The pipeline requires a raw label file containing geospatial labels (CSV, Shapefile, or GeoJSON format) and a configuration in the `datasets.py` file to indicate which columns in the raw label file contain the label coordinates, class, and timestamp. Once the raw label file and configuration are set, the `openmapflow create-datasets` command can be run to generate a processed CSV file with labels and associated remote sensing data that can be used for training a model. The CSV is generated by standardizing the provided labels, addressing duplicate examples, and creating Google Earth Engine export tasks to fetch remote sensing time series data. We used Google Earth Engine to export data because it is available at no cost for research and other noncommercial projects. Each time series example contains 24 timesteps over 2 years. Each timestep represents aggregated satellite observations from one month. The remote sensing data

associated with each label has the same format as the CropHarvest dataset [11] where each timestep contains multi-spectral optical images from Sentinel-2, Synthetic Aperture Radar (SAR) data from Sentinel-1, meteorological data from ERA5, and topographic data from the Shuttle Radar Topography Mission (SRTM). Remote sensing data from Google Earth Engine is exported directly to a Google Cloud Storage bucket. Once the matching remote sensing data are available for a particular label; OpenMapFlow downloads and saves the data to the processed CSV file. To ensure data are added and processed correctly, several data integration tests are run automatically whenever an update to `datasets.py` is pushed to the project repository.

3.1.3 Model Training

After a new dataset has been generated and passes all integration tests, the dataset can be used for model training. A `PyTorchDataset` class is provided for representing the processed CSV as tensors which can then be used for training and evaluating PyTorch models. We chose to use PyTorch [35] instead of other machine learning libraries because it is widely used for ML research.

The model training script is meant to serve as starting point for users to quickly get a model running and visualize predictions. OpenMapFlow provides a Google Colab notebook demonstrating model training by using a Transformer model [36] implemented in PyTorch from the Time Series AI library (TSAI) [37].

3.1.4 Dense Predictions

Once a new model is trained, the command `openmapflow deploy` can be used to deploy the model at scale for efficient dense predictions. The deployment process begins by packaging TorchScript model files into a TorchServe server by building a Docker image. The deployment process then automatically sets up several Google Cloud services necessary for the dense prediction architecture:

- Artifact Registry, used for storing docker images.
- Cloud Run, used for running and scaling the TorchServe-based Docker containers.
- Cloud Buckets, used for storing all remote sensing data from Earth Engine and for storing all predictions made by a model.
- Cloud Functions, used for calling the Cloud Run service on every addition to the remote sensing data Cloud Bucket.
- Cloud Build, used for building the trigger Cloud Function.

The deploy command is run automatically when new models are pushed to Github through a deploy Github action. This ensures the latest models are used for predictions and alleviates the need for installing additional dependencies such as Docker.

Once the deploy command has run successfully, a Google Colab notebook `create_map.ipynb` can be used to interface with the Google Cloud architecture and generate dense predictions using the deployed models on any region of interest. The notebook is connected to the Google Cloud architecture by the `openmapflow.yaml` configuration file. A graphical user interface

is provided to select a deployed model, time frame, and region of interest. Once selected, Google Earth Engine is used to fetch the remote sensing data files for the region of interest into a Google Cloud bucket. Alternatively, the user can elect to make predictions on existing remote sensing data files and avoid fetching new data. Each new file in the remote sensing data bucket triggers a Cloud Function, which calls the Cloud Run service to generate a prediction for every pixel inside the remote sensing data file. The Cloud Run service then uploads the predictions inside a netCDF file to a separate Google Cloud bucket. When predictions have been made for all remote sensing data points in the region of interest, the individual prediction files must be merged into a single geospatial map. Compared to the input remote sensing data files, where each pixel contains band values over several time steps (228 float values per pixel), the output prediction files only contain a single float value for each pixel and therefore are much smaller. Therefore, it is feasible to download all prediction files into the Google Colab environment and merge them into a single geospatial map using the Geospatial Data Abstraction Library (GDAL). Lastly, the geospatial map of predictions is uploaded to Google Earth Engine to make the dense predictions easily shareable and accessible to project stakeholders and the public.

3.2 Experimental Results

3.2.1 Dense Predictions Time and Cost

The OpenMapFlow parallelized architecture offers a drastic time reduction over commonly used sequential prediction pipelines. To quantify this reduction, we measured dense prediction time for several focus regions using OpenMapFlow compared to the prediction pipeline used in Kerner et al. [5], which represents a traditional inference approach. These regions are a subset

Bounding Box	**Area**	**Input data**	**Sequential prediction time**	**Our prediction time**	**Merging time**	**Our cost**
Rwanda	63,018 km^2	0.331 TB	458 mins	12 mins	7 mins	$9.19
Western Ethiopia	160,036 km^2	1.38 TB	1908 mins	19 mins	40 mins	$42.50
Uganda	375,755 km^2	3.06 TB	4232 mins	39 mins	101 mins	$93.25
	1000 km^2	8.83 GB	12 mins	1 min	$<$1 min	$0.30

Table 3.1: Dense Predictions Time and Cost

chosen from the NASA Harvest project focus regions and cover a range of regional areas (63,000 to 376,000 km^2) and input data sizes (0.3 to 3 TB). Specifically we measured 1) the time to generate individual predictions for each remote sensing data file for a region, and 2) the time to merge all individual predictions into a single predicted map file. For OpenMapFlow these times were measured by running the `create_map.ipynb` Colab notebook for each region. For the traditional (sequential) approach, we used a Virtual Machine (VM) to make predictions sequentially on a small region (1000km^2). We then used the input dataset file size (8.83GB) and prediction time (12 minutes and 13 seconds) to compute a rate for sequential predictions by dividing the dataset storage size by prediction time, resulting in 12.05 MB/s. We used this prediction rate to estimate the sequential prediction times for the larger regions. We used a Google Cloud e2-standard-4 (4 vCPUs, 16 GB memory) Virtual Machine for the sequential prediction to be comparable to the default Cloud Run setting of 4 vCPUs used in OpenMapFlow. Cost was computed by summing:

- US regional storage cost for one month for the remote sensing data files and predictions
- Cloud Function CPU and memory cost for the request execution time

- Cloud Run CPU and memory cost for the prediction execution time

The results are shown in Table 3.1. Generally, sequential prediction time will increase linearly with increased data size. In contrast using OpenMapFlow allows scaling compute to ensure prediction time is manageable.

3.2.2 Evaluation on Example Tasks

OpenMapFlow can be used to predict binary classification maps using ML models and remote sensing data for any real-world application domain including agriculture, forestry, oceans, urban areas, and more. To demonstrate the utility of using OpenMapFlow for various societally-relevant applications, we created complete OpenMapFlow projects for three different tasks: cropland classification, crop type classification, and building detection. We used these OpenMapFlow projects to generate geospatial prediction maps and evaluation metrics for each task. These examples are included in the open-source Github repository along with their corresponding datasets, evaluation metrics, and predicted geospatial maps. We describe each task below.

3.2.2.1 Cropland Classification in Togo

Cropland classification consists of classifying each pixel in a satellite image as containing cropland or not. A cropland map indicates where crops are being grown for a particular time period and region and are important datasets needed for agriculture and food security monitoring efforts [38]. Each label for this task has metadata consisting of a coordinate location (latitude/longitude) and a date or time range; the binary class label indicates whether crops were growing in that location during the specified time range or not. We used a total of 1577 labels (991 train, 277 val,

309 test) in Togo obtained from Kerner et al. [5] and an additional 34,270 globally-distributed training labels obtained from the Geowiki dataset [39].

3.2.2.2 Crop Type Classification in Kenya

Binary crop type classification consists of classifying each pixel in a satellite image as containing a specific crop type (e.g., maize) or not. Crop type classification is also often posed as a multi-class classification problem in which each pixel is classified as containing one of N crop types, but we focus on binary classification here. Thereby, labels follow the aforementioned structure but with binary class: maize/non-maize. Like cropland maps, crop type maps are needed for a wide variety of agriculture and food security applications [38]. For this task, we used 2,080 maize/non-maize labels from the CropHarvest dataset [11] in Kenya (1,229 train, 438 val, 413 test).

3.2.2.3 Building Detection in Uganda

Mapping buildings is useful for a variety of applications including population mapping, humanitarian response efforts, and urban planning [40]. For this task, we sampled 8,117 Uganda specific positive (building) labels from the Google Open Buildings dataset [40], 13,993 negative (non-building) labels from the Geowiki dataset [39] and 1,455 negative (non-building) labels from the Uganda Protected Planet dataset [41] (19,944 train, 1,863 val, 1,758 test).

We used OpenMapFlow to train time series classification models for each task. In all experiments, models were trained for 10 epochs with the default hyperparameters specified in the TSAI library. The model with the lowest validation loss was used to report test set metrics

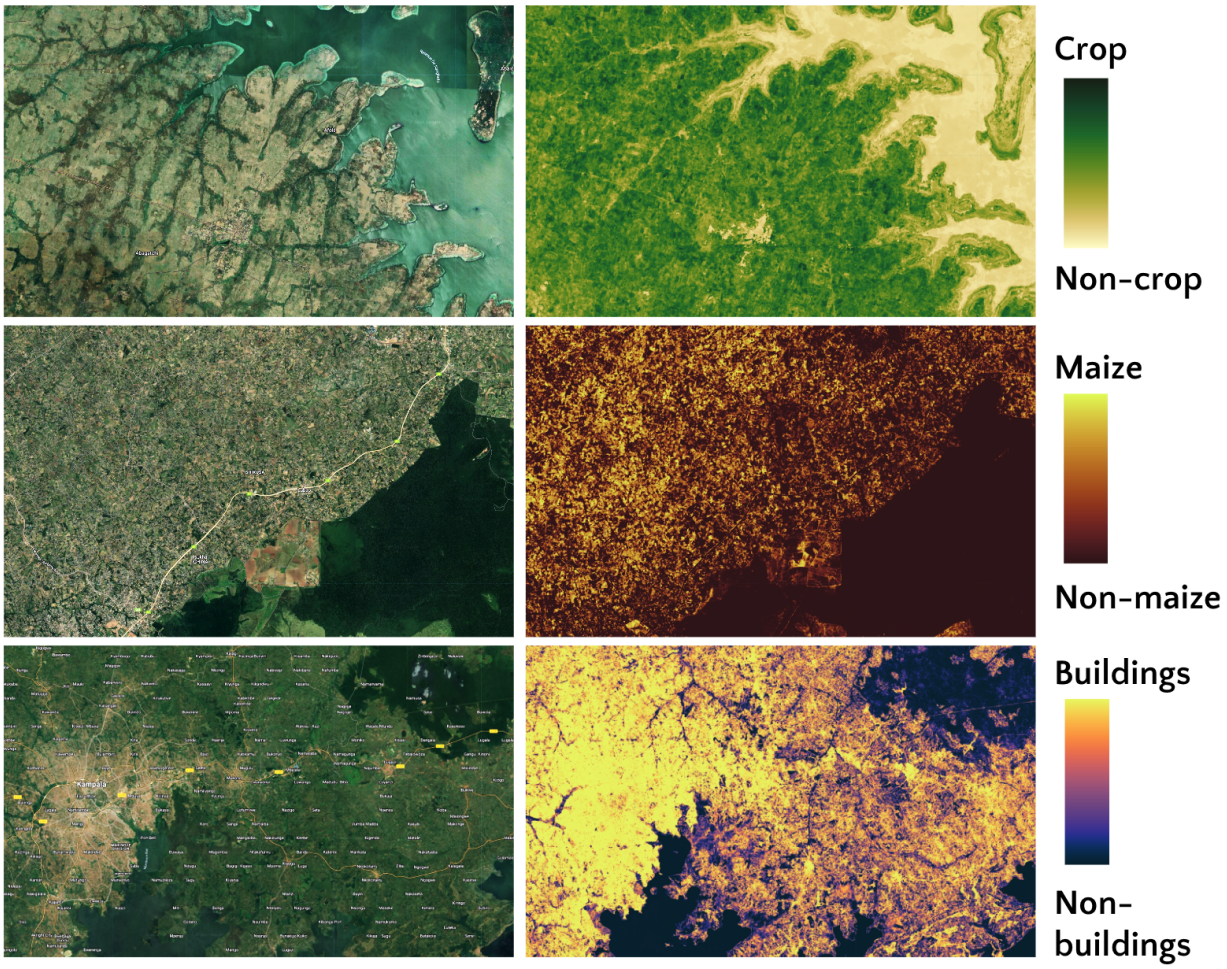

Figure 3.2: Predicted cropland map for region near Abagatchi, Togo (top), predicted crop type (maize) map for region in Kakamega County, Kenya (middle), predicted buildings map for Kampala, Uganda and surrounding region (bottom).

Task	Model Architecture	Batch size	Accuracy	F1-Score
Cropland classification	RNN [42]	64	0.76	0.70
Cropland classification	RNN	256	0.80	0.75
Cropland classification	LSTM [43]	64	0.73	0.67
Cropland classification	LSTM	256	0.74	0.65
Cropland classification	MLP [44]	64	0.78	0.69
Cropland classification	MLP	256	0.74	0.70
Cropland classification	ResNet [44]	64	0.80	0.73
Cropland classification	ResNet	256	0.79	0.74
Cropland classification	ResCNN [45]	64	0.77	0.69
Cropland classification	ResCNN	256	0.71	0.69
Cropland classification	Transformer [36]	64	0.76	0.69
Cropland classification	Transformer	256	0.79	0.71
Cropland classification	gMLP [46]	64	0.80	0.74
Cropland classification	gMLP	256	0.75	0.70
Building detection	Transformer	64	0.97	0.97
Crop type classification (maize)	Transformer	64	0.91	0.64

Table 3.2: Test set metrics for cropland classification, building detection, and crop type classification example tasks.

and make dense predictions. Predicted maps were generated for each task's evaluation set region. Figure 3.2 shows a subset of the prediction maps for each task zoomed in to an example region that illustrates the detection of the class of interest.

We report the accuracy and F1 score for each application task using the default project configuration in Table 3.2. We used accuracy for easy interpretability and F1 score because it is a more representative metric for imbalanced data. For the cropland classification task, we used OpenMapFlow to run additional experiments for six TSAI models in addition to the Transformer and two batch size hyperparameter settings to illustrate how OpenMapFlow enables rapid experimentation with different model architectures and hyperparameter settings. The goal of these results is to illustrate how OpenMapFlow can be used to easily compute metrics and benchmark multiple models for a given task. The focus of this paper is not on optimizing the

results for these three example tasks, but rather on the novel contribution of the OpenMapFlow library which enables ML models to be easily implemented and deployed for a wide variety of applications that use remote sensing data. Thus a detailed evaluation of the predicted maps and exploration of models and hyperparameters to optimize evaluation metrics for each application dataset is out of the scope of this study.

3.3 Limitations

The combination of remote sensing bands used in the CropHarvest dataset (Sentinel-1, Sentinel-2, ERA5, SRTM) [11] is currently the only option available for remote sensing input datasets created using OpenMapFlow. Though this is sufficient for many applications, some applications may require custom data sources for machine learning model inputs. In addition, OpenMapFlow currently only supports single-pixel time series classification and does not currently support datasets with spatial dimensions (e.g. image patch classification, or image patch segmentation). We chose to focus on single pixel time series classification, because classifiers trained on image patches do not allow for high resolution mapping [47] and image patch segmentation requires expensive labels (Table 4.1) which are not always available. In addition, the temporal dimension contains the most relevant information for some remote sensing tasks (particularly for agriculture applications). Though the temporal dimension is important, including spatial inputs would enable the use of a wider array of deep learning models and is therefore important especially as higher resolution remote sensing sensors become publicly available. Lastly, creating new datasets and performing dense predictions with OpenMapFlow currently requires Google Earth Engine and several Google Cloud dependencies, which may limit its flexibility for some users. Future work

includes addressing the data limitations by including other data sources and allowing spatial data as input. Due to the project's open-source nature, community contributions to address these limitations in future work are also possible.

3.4 Research Topics Motivated by Dense Predictions

As discussed in the introduction and illustrated in Figure 1.1, an essential difference between remote sensing datasets and other modalities (e.g., images, videos, text) is that trained models are used to make dense predictions. Together, these predictions form a geospatial map that reveals patterns and errors that may not be apparent from predictions on a set of geographically-sparse samples. Therefore, ML models should be evaluated based on the fidelity of the entire predicted map (the intended result for an end-user), not only a geographically-sparse validation or test set as is the current practice. Evaluating model performance on dense prediction maps is an open area of research that has been under-studied in existing research, despite its importance for real-world adoption of ML models. Dense prediction maps are extremely large (millions of samples) compared to the sparse datasets currently used for training or evaluation (hundreds or thousands of samples). Evaluating the large dense prediction maps that OpenMapFlow enables researchers to produce will require advances in topics such as evaluating models on unlabeled and/or out-of-distribution data (e.g., Deng and Zheng [48], Sun et al. [49]), robustness and adaptation to multiple types of distribution shift (e.g., Guillory et al. [50], Taori et al. [51], Raghunathan et al. [52]), active learning, and error analysis. Techniques for addressing dataset shift and out-of-distribution data are especially important because trained models may be used to make predictions for time periods beyond the training dataset (e.g., predicting crop type maps for a

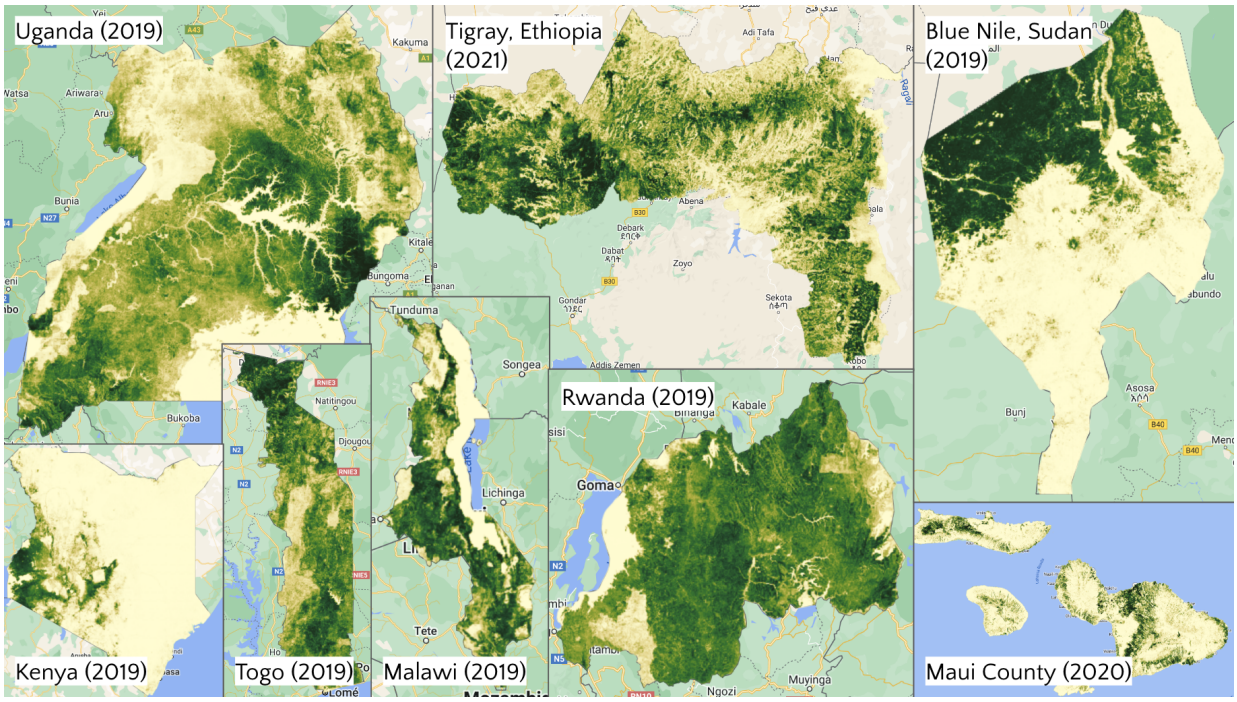

Figure 3.3: Cropland maps developed as part of the crop-mask project

new agricultural season), which exhibit different patterns as weather patterns vary, the climate changes, and land use evolves. OpenMapFlow will also enable these research topics to be studied using real-world, societally-relevant datasets rather than being limited to benchmark datasets.

3.5 Application

OpenMapFlow is currently being used to train and deploy state-of-the-art cropland models [5] as part of NASA Harvest's crop-mask project (https://github.com/nasaharvest/crop-mask). Within the crop-mask project, OpenMapFlow has been used to generate high-resolution, multi-year cropland and crop type maps in multiple countries, including Rwanda, Malawi, Namibia, Sudan, Tanzania, and Zambia (Figure 3.3). These dense prediction maps are being used by stakeholders for socially-impactful downstream applications including cultivated area estimation, crop yield forecasting, and crop conditions assessments.

Chapter 4: A Framework for Usability

4.1 Hierarchy of Needs for Machine Learning for Remote Sensing

The opportunity to contribute towards important societal challenges has led to a surge in the development of novel deep learning ML methods for remote sensing data (also referred to as Earth observation or satellite data). Despite this surge in research, few of the novel ML methods have been adopted in significant application areas. For example, the Allen Coral Reef Atlas, a leading coral reef mapping platform uses a more traditional ML algorithm, the random forest classifier, for classifying geomorphic and benthic classes [16]. Similarly, Global Forest Watch, the state-of-the-art global forest mapping platform, uses a decision tree classifier for mapping forest loss [17].

We believe this lack of adoption stems from novel ML methods for remote sensing data generally overlooking usability. Novelty in the machine learning field is demonstrated by showing improved performance over prior methods through more accurate predictions or decisions on a test set. This paradigm can be observed in the most cited ML papers in recent years [53, 54, 55, 56]. An underlying assumption in this paradigm is that improved performance of an ML method directly correlates with utility, where utility is an actual strong signal of legitimate novelty. However, this assumption only holds if the ML method is usable (it can be easily adopted for the presented use case). If the ML method is difficult or impossible to use its utility will

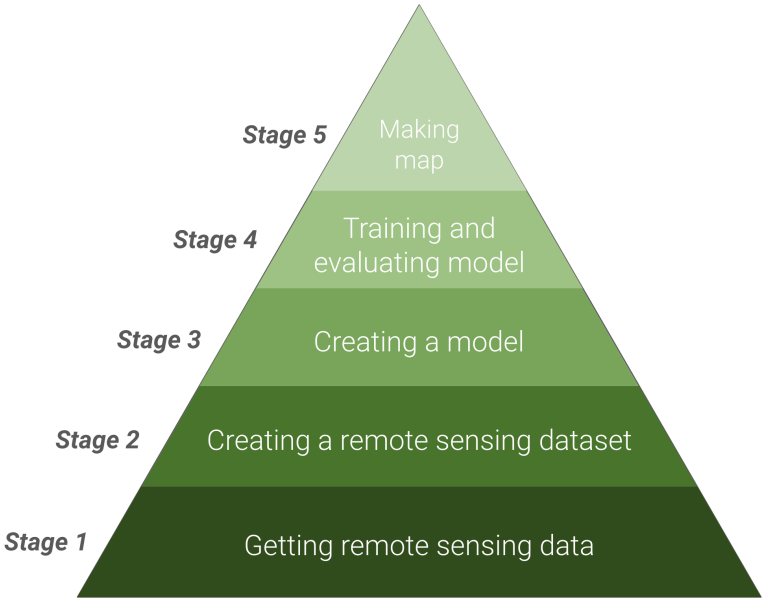

Figure 4.1: Hierarchy of Needs for Usable ML for Remote Sensing

be low regardless of the demonstrated performance. In this chapter we show that the remote sensing data modality in particular presents usability challenges which are critical to consider when developing novel methods.

We propose to view ML for remote sensing data through the lens of a hierarchy of needs pyramid to better understand its unique usability challenges and adoption costs (Figure 4.1). The hierarchy of needs pyramid indicates the stages required for fully adopting a particular ML method for a real world use case. Beginning from the base, our pyramid consists of five levels: 1) getting remote sensing data, 2) creating a remote sensing dataset, 3) creating a model, 4) training and evaluating a model, and 5) making a map. The key idea of the pyramid is that the top stage: 5) making a map, can only be achieved through fulfilling the needs of all prior stages. The ability to make a map with ML and remote sensing data is not always equivalent to usefulness or "model self-actualization", but it does make a method significantly easier to adopt for a real-world use case.

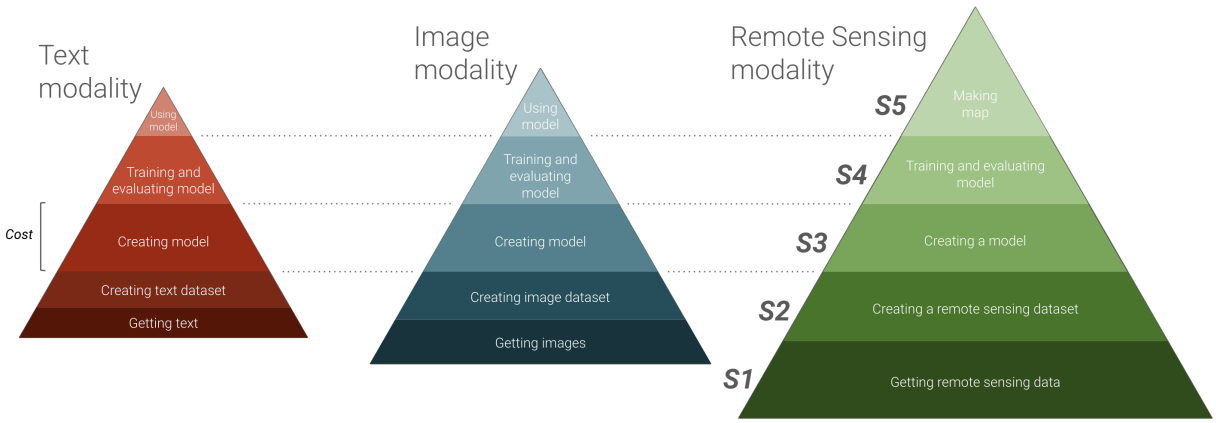

Figure 4.2: Usable ML hierarchy of needs comparison across different data modalities

4.2 Comparing Machine Learning Data Modalities

To broadly understand the adoption challenges of the remote sensing data modality we compare it with the two most popular data modalities in ML: images and text. At the time of writing, the top 10 most cited ML datasets (according to Papers With Code [57]) were all image datasets (e.g., CIFAR-10 [58], ImageNet [59], COCO [60], MNIST [61]). The next most cited datasets were of the text modality (e.g., GLUE [62], SQuAD [63], SST [64]). Further evidence for the dominance of these two data modalities can be seen through the type of data used by the most cited ML papers in recent years such as "Adam: A Method for Stochastic Optimization" [53], "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" [54], "Attention Is All You Need" [55], and "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" [56].

We propose a hierarchy of needs pyramid for each modality to facilitate usability comparison across data modatilies (Figure 4.2). The height of each stage corresponds to the implementation cost which encompasses the time, compute, and storage costs to implement and execute code for the stage. For the purposes of this thesis, cost is estimated solely as a relative qualitative

measure to compare data modalities. In the hierarchies for each of the compared modalities, stage 3 (creating a model) and stage 4 (training and evaluating a model) have equivalent height (representing equivalent implementation cost) across text, images, and remote sensing. Though each modality has unique nuances for model creation, training, and evaluation, for simplicity we claim that implementation cost is roughly equivalent in all three. The key differences in implementation cost arise in the remaining stages: obtaining input data (stage 1), creating a dataset (stage 2), and using the model (stage 5).

4.2.1 Obtaining Input Data

To use an ML method one needs the ability to obtain input data beyond the original dataset but in the same format. If it is not possible to obtain input data beyond the original dataset, the method cannot be used to make predictions for any purpose beyond the dataset. Ability to obtain input data is also critical for creating new or additional datasets for training or fine-tuning. Perhaps the easiest way to obtain input data for a specific task is to generate it yourself. This is often straightforward for text and image data in which consumer devices can be used to generate text or take photos. Remote sensing data cannot be generated on consumer devices but must be obtained from the appropriate data source for a specific location and time which requires significantly more effort. Beyond just obtaining input data, it is critical to ensure it has a consistent format with the original dataset. For example if an image model was trained on color images with a height and width of 64 pixels, and the model is being adopted for a use case where input images have a height and width of 224 pixels, either the model needs to be re-trained (or fine-tuned) on the higher resolution images or the new images need to be transformed to the

same size as the input. While both options are relatively straightforward for the image modality the same cannot be said about the remote sensing data modality. To better understand these differences, we will discuss data input spaces. A data input space can be described as all the sets of values that can comprise a data input sample. Mathematically, an input space can be expressed as

$$\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_k}$$

where $\mathcal{X}$ is the input space, and $\boldsymbol{x}_i$ is a specific vector in that space with k dimensions.

Natural color image datasets the input space can be described as:

$$\mathcal{X}_{images} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in 256^{d_{height} \times d_{width} \times 3}$$

While the input spaces for various image datasets are different, it is fairly trivial to transform any image into the dimensions required by a particular method by scaling and/or cropping the height and/or width dimensions.

In the case of the text data modality, each input sample consists of a sequence of words which are converted to tokens. In the simplest case, each word is assigned a number associated with the index of that word in a vocabulary (V). Given a max input token length $d_{sequence}$, the text modality input space can thereby be expressed as:

$$\mathcal{X}_{text} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in V^{d_{sequence}}$$

Given this dimensionality, the text data modality often requires even less effort than the image data modality to transform any raw input (given it is in the same language) into the

appropriate format for a particular method. This is done by tokenizing the raw text using a common vocabulary and either padding it to match the sequence length $d_{sequence}$ or splitting into several sequences of size $d_{sequence}$.

The input space for the remote sensing data modality poses challenges that are not typically observed in image and text data modalities. Remote sensing datasets often include values from a variety of sensors where each sensor has unique dimensionality, which adds substantial complexity [65]. Even a single remote sensing sensor, for example Sentinel-2, can have varying spatial resolutions for each spectral band. Additionally, the temporal dimension provides critical information for many applications but varies highly among different sensors, requiring specific temporal alignment design decisions. Finally, values from different sensors have different ranges that require unique processing and scaling. Assuming values for each sensor (indexed by j) are scaled to range a to b represented as the interval S_j where $S_j = \{x \in \mathbb{R} : a \leq x \leq b\}$ and the dimensions for each sensor are denoted as d_j the input space for remote sensing data can be expressed as:

$$\mathcal{X}_{rs} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\},$$

$$\boldsymbol{x_i} = [\boldsymbol{x_{i_1}}, \boldsymbol{x_{i_2}}, ..., \boldsymbol{x_{i_s}}], \boldsymbol{x_{i_j}} \in S_j^{d_{j_{height}} \times d_{j_{width}} \times d_{j_{time}} \times d_{channel}}$$

Given this large dimensionality, users of remote sensing datasets often use a subspace of the entire remote sensing data input space. Example strategies for reducing the input dimensionality include using a single sensor, using a single time step, using a single pixel (reducing the height and width dimensions to one), using a subset of the channels available, or using a subset of the temporal dimension. Given this complexity, obtaining and transforming remote sensing data into

Data Modality	Labeling Task	Price Per Label
Text	Classification	$0.012 [66]
	Named Entity Recognition	$0.024 [66]
Image	Classification	$0.012 [66]
	Bounding Box	$0.036 [66]
	Semantic Segmentation	$0.84 [66]
Remote Sensing	Classification	$0.035 [67]
	Bounding Box	$0.4375 [67]
	Segmentation	$8.75 [67]

Table 4.1: Cost of labeling across modalities

Data Modality	Dataset	Size	Num. Samples	Sample Size
Text	SQuAD [63]	125 MB	98,169	1.27 KB
	SST [64]	20.4 MB	11,855	1.72 KB
Image	MNIST [61]	15.7 MB	70,000	0.22 KB
	CIFAR-10 [58]	170 MB	60,000	2.83 KB
	ImageNet [59]	155 GB	590,326	10.92 KB
Remote Sensing	BigEarthNet [68]	121 GB	590,326	262.57 KB
	PASTIS [69]	29 GB	2,433	11.92 MB

Table 4.2: Sample storage size across modalities

the format required by a particular model requires explicit documentation and code, and generally has a significantly higher implementation cost than image and text data modalities.

4.2.2 Creating Datasets

After a pipeline for obtaining data has been established, the next stage to adoption often involves creating additional labeled datasets for training (or fine-tuning) and evaluation for the specific use case at hand. The cost of creating datasets can be broken down into labeling cost and storage cost. We estimated the typical labeling cost for each data modality and associated tasks by checking the price per labeling task on popular labeling platforms: Amazon Mechanical Turk

[66] for images and text and Development Seed [67] for remote sensing data. Table 4.1 provides a simple overview of label pricing for common tasks across different modalities. The Price Per Label column shows that on average the text modality has a lower labeling cost than the image modality which in turn has a lower labeling cost than the remote sensing data modality.

Since storage cost depends on storage size, comparisons of storage cost can be done by comparing typical file sizes across different data modalities. In table 4.2 we list typical sample sizes used in popular ML datasets. Similar to labeling cost, the text modality has a lower average sample size than images, which have a lower sample size than remote sensing data. Both tables demonstrate the higher implementation cost of creating remote sensing datasets over the comparison modalities.

4.2.3 Using Models

Once an ML model is trained and evaluated, it can be used to make predictions. Here we consider cost of using a model as the cost per prediction multiplied by the amount of predictions necessary for utility, where utility is defined as ability to inform a subsequent action. The cost per prediction is closely correlated with model size—a model with more parameters requires more compute and therefore higher cost. Since model sizes within data modalities vary widely we will consider a uniform model size when comparing different data modalities and instead focus on the amount of predictions necessary for utility.

Example applications in the text data modality include spam detection [70], translation [71], and text summarization [72]. In all of these applications a prediction on a single text data sample can lead to a subsequent action and therefore be considered useful. For example,

classifying an email as spam can lead to the action of the email being filtered out from a user's inbox. For the image data modality, common applications include self-driving cars [73], face recognition [73], and medical imaging [74]. Similar to the text data modality, in the image modality, predictions on a single image sample can lead to a subsequent action. For example in medical imaging, detecting the presence of a tumor in single image sample can lead to further analysis and potential surgery. In some cases predictions on several image samples across time are needed for determining subsequent actions such as in self-driving applications.

In the remote sensing data modality, example applications include land cover classification [18], crop yield prediction [75], and disaster response [7]. In this modality, a prediction on a single sample is almost always insufficient for a subsequent action. Take the example of land cover classification in which pixels on the Earth's surface are classified by land cover. A common subsequent action of land cover classification is environmental change detection [76] which involves analyzing where certain types of land cover change occur (e.g., forest to cropland). This subsequent action *can only be performed on a map* (predictions across every location in a region of interest). Therefore unlike the text and image data modalities, in the remote sensing data modality predictions often need to be made on thousands to millions of samples (depending on the sample size and size of the study region) for a subsequent action. Returning to our proposed model usage cost as the cost per prediction multiplied by the amount of predictions necessary for utility, the distinction in amount of predictions necessary for utility between different modalities has a massive impact on the cost. Assuming a constant model size, the text data modality usually has the lowest cost of using the model with the image data modality following. The remote sensing data modality has a far higher cost of using a particular model than other modalities due to the number of predictions needed for utility.

4.3 Machine Learning for Remote Sensing Usability Cards

Trained ML models are increasingly released and adopted for real world use-cases which has resulted in the creation of Model Cards for Model Reporting [32] to ensure responsible usage. While this is a common practice in the text and image modality, in the remote sensing data modality sharing trained model weights is insufficient for usage. In the prior section we showed that the cost of adopting an ML method for the remote sensing data modality is significantly higher than other data modalities due to unique challenges in obtaining input data, creating datasets and using the model. These challenges are a major factor in the low adoption rate of novel ML methods even when they are critically needed for many real world applications. Before responsible usage can be assessed, usage alone must be possible. Therefore to help the community better gauge the usability of novel ML work for remote sensing data we propose Usability Cards which are meant to answer the question: how difficult will it be to adopt and use the ML method presented in this paper or system? Our Usability Cards consist of a checklist for each stage of the hierarchy of needs pyramid (Figure 4.3 and Table 4.3). Once filled out, it is meant to provide a clear view of how much additional work must be done for the adoption of a particular ML for remote sensing method presented in a paper or system.

Stage 1 contains four checklist items related to obtaining remote sensing data. We make a deliberate distinction between the remote sensing data format (sensors, resolutions) and remote sensing data processing (e.g. cleaning, harmonization) as the processing can have a significant effect on the data. For example, multispectral optical images from the Sentinel-2 satellite contain 13 bands at spatial resolutions between 10-60 meters and temporal revisit rate of 5 days (i.e., every location on the planet is imaged every 5 days). Cloud artifacts (occlusions) often appear in

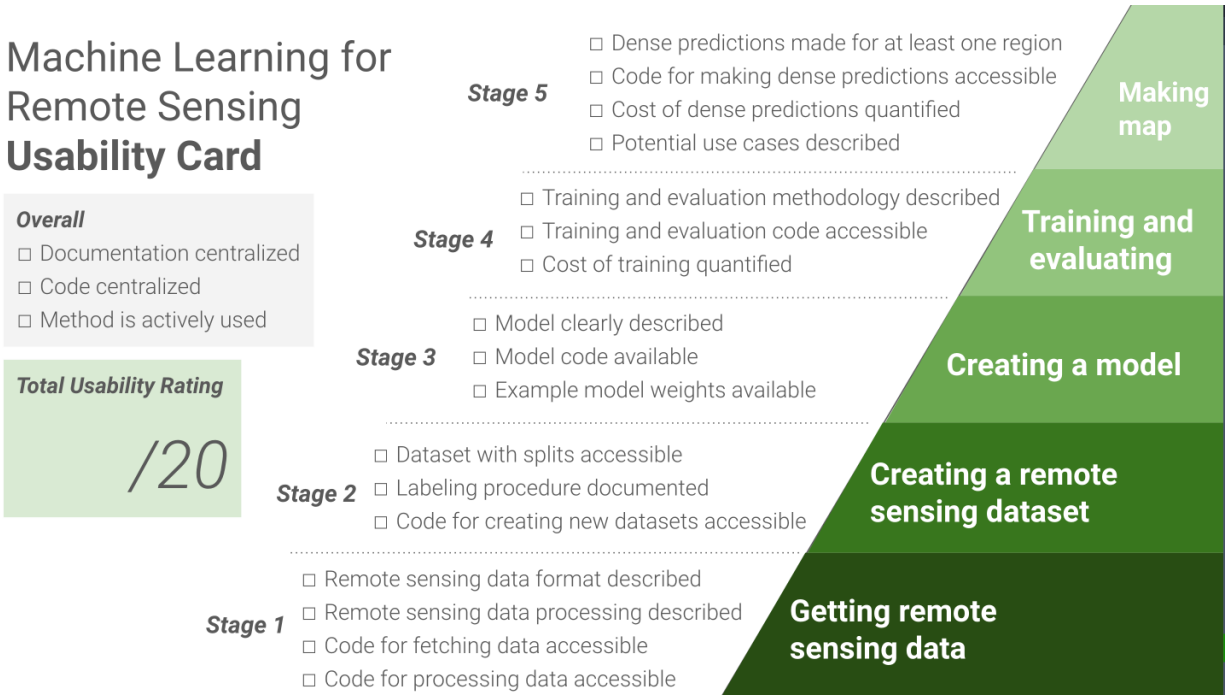

Figure 4.3: ML for Remote Sensing Usability Card

these images and thus processing steps such as masking out and/or filling in cloud-covered pixels, or selecting the least cloudy observation within a time frame [11], are almost always applied. To obtain full points for this stage, the remote sensing data format and processing must be described and the code for obtaining and processing the data must be available.

An ML paper or system either creates a new dataset or uses an existing dataset for training and evaluation (stage 2). To ensure reproducibility, the dataset used along with dataset splits should be made available. To ensure usability, the process of creating additional datasets should also be reproducible. If it is not possible to create additional datasets, then it is not possible to evaluate the ML method on any data outside the original work. The first step of creating a labeled dataset involves generating labels. This implies that the labeling procedure must be clearly documented including input data used, sampling strategy, label definitions, and labeling platform used. A vague labeling procedure leaves the adopter making assumptions which can

Stage in Hierarchy of Needs	Checklist
1. Getting remote sensing data	☐ Remote sensing data format described ☐ Remote sensing data processing described ☐ Code for fetching data accessible ☐ Code for processing data accessible
2. Creating a remote sensing dataset	☐ Dataset with splits accessible ☐ Labeling procedure documented ☐ Code for creating new datasets accessible
3. Creating a model	☐ Model clearly described ☐ Model code available ☐ Example model weights available
4. Training and evaluation	☐ Training and evaluation methodology described ☐ Training and evaluation code accessible ☐ Cost of training quantified
5. Making map	☐ Dense predictions made for at least one region ☐ Code for making dense predictions accessible ☐ Cost of dense predictions quantified ☐ Potential use cases described
Overall	☐ Documentation centralized ☐ Code centralized ☐ Method is actively used
Usability Rating	Number of checked boxes / 20

Table 4.3: Questions for assessing usability of ML for remote sensing

result in significantly different results than the original work. A common practice in the remote sensing data modality involves using a higher quality dataset for labeling [77] which differs from the ML input dataset. In this case, the label is associated with a location (e.g., coordinate, polygon) and time and not the actual data sample being labeled. The georeferenced label is then matched to the appropriate input data to create an ML-ready dataset. This additional step is not necessary for text and image data modalities where the input data is labeled directly, but is critical for creating new labeled datasets in the remote sensing data modality. Thereby the code

used for associating labels with the appropriate input data in order to create an ML-ready dataset should be made available. In an ideal case a datasheet [31] is made available in conjuction with the dataset, which contains information about how the dataset was created and how it should be used. However focusing specifically on usability for the remote sensing data modality, we can narrow down the key requirements into three checklist items: 1) dataset and splits have been made available, 2) the labeling procedure is documented, and 3) the code for creating new datasets is accessible.

Stage 3 (creating a model) includes checklist items for describing and making the code accessible for the model architecture, and making example model weights available. Inaccessible model code means that from-scratch implementation is necessary for adoption. Absence of example model weights becomes a serious concern when the model has a large parameter size, as it implies that fine-tuning is not possible and a new model must be trained from scratch for usage which may require significant compute cost. Stage 4 (training and evaluation) naturally follows stage 3 and includes checklist items for describing and making the code accessible for training and evaluation. Similar to an absence of model code, absence of training and evaluation code puts the burden of implementation on the end-user and makes reproducibility more difficult. A checklist item for quantifying the cost of training is also added and has become increasingly relevant as larger models which take a long time and a large amount of compute to train become more popular. To check off this item the time to train and hardware used should be reported.

Stage 5 (making map), contains four checklist items to ensure the trained model can be used for downstream tasks. The first item is demonstrating model usage through dense geospatial predictions over at least one region forming a predicted map. Error modes stemming from varying sensor resolutions and artifacts are often not revealed during i.i.d. sample evaluation and thereby

performing dense predictions is important to ensure their absence. In addition, a predicted map can be directly used for a downstream use case. Accessible code for generating a predicted map with the ML method is vital to ensure predictions can be made for study regions beyond the original work. The large scale of predictions needed for generating a predicted map make compute and storage costs an important consideration when deciding whether to adopt an ML method. For example if a novel deep learning based model is shown to perform better than a random forest classifier, but the cost of making millions of predictions with the novel model is several factors more expensive than a random forest classifier, it will likely not be adopted by many practitioners. Therefore a quantification of the cost of compute for dense predictions is important to measure and report. The last item follows the spirit of Model Cards [32], but rather than a description of potential use cases of the trained model, our Usability Cards requires a description of potential use cases of the generated predicted maps.

Finally three checklist items are devoted to whether all documentation and code is available in one location and if the method is actively being used for some real-world task (which is a strong signal of utility). These factors impact usability but are not associated with a particular stage. The total usability rating is the sum of all boxes checked with a high score indicating a low cost of implementation, and thus increased likelihood of end-user adoption.

The proposed Usability Card is defined from a hierarchy of needs perspective, in which each stage depends on the usability of the prior stages. Omissions from the early stages have compounding consequences for usability on the later stages. For example if the remote sensing data format and remote sensing data processing are never formally described (stage 1), it becomes very difficult to create a remote sensing dataset (stage 2), and thereby to create a model architecture (stage 3), and thereby to train and evaluate a model (stage 4), and thereby to use the model for

map creation (stage 5). Therefore for a holistic assessment of usability we encourage the use of the entire card rather than just the usability rating. We hope Usability Cards will aid remote sensing scientists and other practitioners in assessing novel ML methods. We also hope that Usability Cards will aid ML researchers in assessing anticipated impact of existing work and developing methods that have a high chance of end-user adoption.

4.4 Case study

To demonstrate the use of the Usability Card, we use it to conduct a case study of the crop-mask system [78] discussed in Chapter 3. The crop-mask system uses the OpenMapFlow library to provide an end-to-end workflow for generating regional cropland maps, which are critical datasets for agriculture and food security monitoring [38]. To assess the usability of this method we score it using the proposed Usability Card in Table 4.4.

The crop-mask system scores full points on stage 1: getting remote sensing data, since the remote sensing data and processing is described [11] and the code for obtaining and processing remote sensing data is available through the OpenMapFlow library. Two out of three points are obtained in stage 2, because the latest version of the dataset used has not yet been released. However, the labeling procedure is documented in the wiki [78]. Since crop-mask uses the OpenMapFlow library, the code for creating new datasets is accessible (Section 3.1.2). Two out of three points are obtained for stage 3. While the model is clearly described [5] and the model code is available inside the crop-mask repository [78], example model weights are not publicly available. For stage 4, the training and evaluation methodology is described in the crop-mask wiki and the code for training and evaluation is accessible [78], however the cost of training is not

Stage	Checklist and Reference
1. Getting remote sensing data	☑ Remote sensing data format described ☑ Remote sensing data processing described ☑ Code for fetching data accessible ☑ Code for processing data accessible
2. Creating a remote sensing dataset	☐ Dataset with splits accessible ☑ Labeling procedure documented ☑ Code for creating new datasets accessible
3. Creating a model	☑ Model clearly described ☑ Model code available ☐ Example model weights available
4. Training and evaluation	☑ Training and evaluation methodology described ☑ Training and evaluation code accessible ☐ Cost of training quantified
5. Making map	☑ Dense predictions made for at least one region ☑ Code for making dense predictions accessible ☑ Cost of dense predictions quantified ☐ Potential use cases described
Overall	☐ Documentation centralized ☑ Code centralized ☑ Method is actively used
Usability Rating	15 / 20

Table 4.4: Assessment of the usability of the crop-mask project (github.com/nasaharvest/crop-mask)

explicitly quantified. The crop-mask system has been used to make dense predictions for many regions (as shown in Figure 3.3), which satisfies the first checklist item in stage 5. Next since crop-mask uses the OpenMapFlow library, the code for making dense predictions is accessible (Section 3.1.4) and the cost of dense predictions is quantified (Table 3.1). However, potential use cases of cropland maps produced by crop-mask are not explicitly described in the documentation. The different documentation for the crop-mask system is spread through out related papers and its wiki page which means no point is awarded for centralized documentation. However, the code

for the crop-mask system is centralized and the system is currently used by the NASA Harvest consortium. Summing the awarded checks, we obtain a usability rating of 15/20 indicating that this system can be adopted with relatively low implementation cost while also showing where future work is needed for improved usability.

Chapter 5: Conclusion

To reduce the barrier of adopting machine learning methods for remote sensing data, we presented OpenMapFlow, a library for rapid geospatial map creation with ML and remote sensing data. OpenMapFlow was created to help make dense prediction an integral component of ML research using remote sensing data, thereby making them more usable. We demonstrated how OpenMapFlow enables rapid start-up, experimentation, iteration, and deployment for ML projects using experiments for three real-world tasks (cropland, crop type, and building classification). We showed that OpenMapFlow greatly reduces the time required to generate dense prediction maps using trained models compared to traditional approaches. OpenMapFlow is open-source (https://github.com/nasaharvest/openmapflow) and accessible as a Python package on the Python Package Index (PyPI). OpenMapFlow is currently used to train and deploy state-of-the-art cropland models as part of NASA Harvest's crop-mask system. These models have been used to generate high-resolution, multi-year cropland maps which are being used by stakeholders for socially-impactful downstream applications including cultivated area estimation, crop yield forecasting, and crop conditions assessments.

To better understand the low adoption rate of novel ML methods for remote sensing data we compared the remote sensing data modality to other common data modalities in the ML field using a hierarchy of needs framework. We then proposed ML for Remote Sensing Usability Cards

to assess usability of any ML paper or system using remote sensing data. The crop-mask system used for creating high resolution cropland maps was assessed to provide a detailed example. We hope our ML for Remote Sensing Usability Cards allow both ML researchers and remote sensing scientists to easily assess the usability of ML methods and make appropriate changes to ensure continued progress and innovation. Altogether we hope the ML for Remote Sensing Usability Cards in conjunction with the OpenMapFlow library increase the adoption of novel ML research for real-world tasks and impact.

Bibliography

[1] Ritesh Pradhan, Ramazan S. Aygun, Manil Maskey, Rahul Ramachandran, and Daniel J. Cecil. Tropical cyclone intensity estimation using a deep convolutional neural network. *IEEE Transactions on Image Processing*, 27(2):692–702, 2018. doi: 10.1109/TIP.2017.2766358.

[2] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

[3] Stephan Rasp, Michael S Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39): 9684–9689, 2018.

[4] Sherrie Wang, George Azzari, and David B. Lobell. Crop type mapping without field-level labels: Random forest transfer and unsupervised clustering techniques. *Remote Sensing of Environment*, 222:303–317, 2019. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2018.12.026. URL https://www.sciencedirect.com/science/article/pii/S0034425718305790.

[5] Hannah Kerner, Gabriel Tseng, Inbal Becker-Reshef, Catherine Nakalembe, Brian Barker, Blake Munshell, Madhava Paliyam, and Mehdi Hosseini. Rapid response crop maps in data sparse regions. In *Proceedings of the ACM SIGKDD Conference on Data Mining and Knowledge Discovery Workshops*, 2020.

[6] Gabriel Tseng, Hannah R Kerner, Catherine L Nakalembe, and Inbal Becker-Reshef. Annual and in-season mapping of cropland at field scale with sparse labels. In *Proceedings of the Thirty-fourth Conference on Neural Information Processing Systems Workshops*, 2020. URL https://www.climatechange.ai/papers/neurips2020/29.

[7] Derrick Bonafilia, Beth Tellman, Tyler Anderson, and Erica Issenberg. Sen1floods11: A georeferenced dataset to train and test deep learning flood algorithms for sentinel-1. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[8] Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. Transfer learning from deep features for remote sensing and poverty mapping. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[9] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pages 5901–5904. IEEE, 2019.

[10] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018.

[11] Gabriel Tseng, Ivan Zvonkov, Catherine Lilian Nakalembe, and Hannah Kerner. Cropharvest: A global dataset for crop-type classification. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=JtjzUXPEaCu`.

[12] Hamed Alemohammad and Kevin Booth. Landcovernet: A global benchmark land cover classification training dataset. In *Proceedings of the Thirty-fourth Conference on Neural Information Processing Systems Workshops*, 2020.

[13] Devis Tuia, Claudio Persello, and Lorenzo Bruzzone. Domain adaptation for the classification of remote sensing data: An overview of recent advances. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):41–57, 2016.

[14] Shaoyue Song, Hongkai Yu, Zhenjiang Miao, Qiang Zhang, Yuewei Lin, and Song Wang. Domain adaptation for convolutional neural networks-based remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 16(8):1324–1328, 2019.

[15] Ebrahim Babaeian, Sidike Paheding, Nahian Siddique, Vijay K Devabhaktuni, and Markus Tuller. Estimation of root zone soil moisture from ground and remotely sensed soil information with multisensor data fusion and automated machine learning. *Remote Sensing of Environment*, 260:112434, 2021.

[16] Mitchell B. Lyons, Chris M. Roelfsema, Emma V. Kennedy, Eva M. Kovacs, Rodney Borrego-Acevedo, Kathryn Markey, Meredith Roe, Doddy M. Yuwono, Daniel L. Harris, Stuart R. Phinn, Gregory P. Asner, Jiwei Li, David E. Knapp, Nicholas S. Fabina, Kirk Larsen, Dimosthenis Traganos, and Nicholas J. Murray. Mapping the world's coral reefs using a global multiscale earth observation framework. *Remote Sensing in Ecology and Conservation*, 6(4):557–568, 2020. doi: https://doi.org/10.1002/rse2.157. URL `https://zslpublications.onlinelibrary.wiley.com/doi/abs/10.1002/rse2.157`.

[17] M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. A. Turubanova, A. Tyukavina, D. Thau, S. V. Stehman, S. J. Goetz, T. R. Loveland, A. Kommareddy, A. Egorov, L. Chini, C. O. Justice, and J. R. G. Townshend. High-resolution global maps of 21st-century forest

cover change. *Science*, 342(6160):850–853, 2013. doi: 10.1126/science.1244693. URL https://www.science.org/doi/abs/10.1126/science.1244693.

[18] Christopher F. Brown, Steven P. Brumby, Brookie Guzder-Williams, Tanya Birch, Samantha Brooks Hyde, Joseph Mazzariello, Wanda Czerwinski, Valerie J. Pasquarella, Robert Haertel, Simon Ilyushchenko, Kurt Schwehr, Mikaela Weisse, Fred Stolle, Craig Hanson, Oliver Guinan, Rebecca Moore, and Alexander M. Tait. Dynamic world, near real-time global 10 m land use land cover mapping. *Scientific Data*, 9(1):251, Jun 2022. ISSN 2052-4463. doi: 10.1038/s41597-022-01307-4.

[19] Kiri L. Wagstaff. Machine learning that matters. In *Proceedings of the 29th International Conference on Machine Learning*, page 1851–1856, 2012.

[20] Ram Shankar Siva Kumar, David O Brien, Kendra Albert, Salomé Viljöen, and Jeffrey Snover. Failure modes in machine learning systems. *arXiv preprint arXiv:1911.11034*, 2019.

[21] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *Proceedings of the European Conference on Computer Vision*, pages 340–353. Springer, 2012.

[22] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *ECCV*, 2020.

[23] Shreya Shankar, Yoni Halpern, Eric Breck, James Atwood, Jimbo Wilson, and D. Sculley. No classification without representation: Assessing geodiversity issues in open data sets for the developing world. In *Thirty-first Conference on Neural Information Processing Systems Workshops*, 2017.

[24] Vitor CF Gomes, Gilberto R Queiroz, and Karine R Ferreira. An overview of platforms for big earth observation data management and analysis. *Remote Sensing*, 12(8):1253, 2020.

[25] Adam J Stewart, Caleb Robinson, Isaac A Corley, Anthony Ortiz, Juan M Lavista Ferres, and Arindam Banerjee. Torchgeo: deep learning with geospatial data. *arXiv preprint arXiv:2111.08872*, 2021.

[26] Hamed Alemohammad. Radiant mlhub: A repository for machine learning ready geospatial training data. In *AGU Fall Meeting Abstracts*, 2019.

[27] Noel Gorelick, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 202:18–27, 2017.

[28] FAO. Sepal repository. https://github.com/openforis/sepal/, 2020. Accessed: 2022-07-03.

[29] Carly Marie Beneke, Samuel Skillman, Michael S Warren, Tim Kelton, Steven Patrick Brumby, Rick Chartrand, and Mark Mathis. A platform for scalable satellite and geospatial data analysis. In *AGU Fall Meeting Abstracts*, 2017.

[30] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, Mar 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.18.

[31] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, Dec 2021.

[32] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 220–229, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361255. doi: 10.1145/3287560.3287596. URL https://doi.org/10.1145/3287560.3287596.

[33] Alexander Lavin, Ciarán M. Gilligan-Lee, Alessya Visnjic, Siddha Ganju, Dava Newman, Atilim Gunes Baydin, Sujoy Ganguly, Danny B. Lange, Ajay Sharma, Stephan Zheng, Eric P. Xing, Adam Gibson, James Parr, Chris Mattmann, and Yarin Gal. Technology readiness levels for machine learning systems. *ArXiv*, abs/2101.03989, 2021.

[34] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[36] Marc Rußwurm and Marco Körner. Self-attention for raw optical satellite time series classification. *ISPRS journal of photogrammetry and remote sensing*, 169:421–435, 2020.

[37] Ignacio Oguiza. tsai - a state-of-the-art deep learning library for time series and sequential data. https://github.com/timeseriesAI/tsai, 2022. URL https://github.com/timeseriesAI/tsai. Accessed: 2022-05-10.

[38] Catherine Nakalembe, Christina Justice, Hannah Kerner, Christopher Justice, and I Becker-Reshef. Sowing seeds of food security in africa. *Eos*, 102, 2021.

[39] Linda See. A global reference database of crowdsourced cropland data collected using the Geo-Wiki platform. https://doi.org/10.1594/PANGAEA.873912, 2017. URL https://doi.org/10.1594/PANGAEA.873912. Accessed: 2022-06-01.

[40] Wojciech Sirko, Sergii Kashubin, Marvin Ritter, Abigail Annkah, Yasser Salah Eddine Bouchareb, Yann Dauphin, Daniel Keysers, Maxim Neumann, Moustapha Cisse, and John Quinn. Continental-scale building detection from high resolution satellite imagery. *arXiv preprint arXiv:2107.12283*, 2021.

[41] UNEP-WCMC. Protected area profile for uganda from the world database on protected areas, august 2022. www.protectedplanet.net, 2022. Accessed: 2022-08-01.

[42] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[44] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[45] Xiaowu Zou, Zidong Wang, Qi Li, and Weiguo Sheng. Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification. *Neurocomputing*, 367:39–45, Nov 2019. ISSN 0925-2312. doi: 10.1016/j.neucom.2019.08.023.

[46] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021.

[47] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. 08 2017. doi: 10.1109/JSTARS.2019.2918242.

[48] Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15069–15078, 2021.

[49] Xiaoxiao Sun, Yunzhong Hou, Weijian Deng, Hongdong Li, and Liang Zheng. Ranking models in unlabeled new environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11761–11771, 2021.

[50] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1134–1144, 2021.

[51] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

[52] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA, 2015. MIT Press.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[56] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[57] Meta AI. Papers with code - datasets. https://paperswithcode.com/datasets, 2022. Accessed: 2023-01-05.

[58] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[60] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[61] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[62] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

[63] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

[64] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

[65] Charles Toth and Grzegorz Jóźków. Remote sensing platforms and sensors: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:22–36, 2016. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2015.10.004. URL https://www.sciencedirect.com/science/article/pii/S0924271615002270. Theme issue 'State-of-the-art in photogrammetry, remote sensing and spatial information science'.

[66] Amazon. Amazon sagemaker data labeling pricing. https://aws.amazon.com/sagemaker/data-labeling/pricing/, 2023. Accessed: 2023-03-31.

[67] Development Seed. Data annotation team. https://developmentseed.org/expertise/data-team/, 2023. Accessed: 2023-03-31.

[68] Gencer Sumbul, Marcela Charfuelan, Beg"um Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. *CoRR*, abs/1902.06148, 2019.

[69] Vivien Sainte Fare Garnot and Loic Landrieu. Panoptic segmentation of satellite image time series with convolutional temporal attention networks. *ICCV*, 2021.

[70] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa, et al. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.

[71] Shashi Pal Singh, Ajai Kumar, Hemant Darbari, Lenali Singh, Anshika Rastogi, and Shikha Jain. Machine translation using deep learning: An overview. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pages 162–167, 2017. doi: 10.1109/COMPTELIX.2017.8003957.

[72] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*, 2017.

[73] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[74] Bradley J Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L Kline. Machine learning for medical imaging. *Radiographics*, 37(2):505–515, 2017.

[75] Priyanga Muruganantham, Santoso Wibowo, Srimannarayana Grandhi, Nahidul Hoque Samrat, and Nahina Islam. A systematic literature review on crop yield prediction with deep learning and remote sensing. *Remote Sensing*, 14(9):1990, 2022.

[76] Louisa J.M. Jansen and Antonio Di Gregorio. Parametric land cover and land-use classifications as tools for environmental change detection. *Agriculture, Ecosystems Environment*, 91(1):89–100, 2002. ISSN 0167-8809. doi: https://doi.org/10.1016/S0167-8809(01)00243-2. URL https://www.sciencedirect.com/science/article/pii/S0167880901002432.

[77] Pontus Olofsson, Giles M. Foody, Martin Herold, Stephen V. Stehman, Curtis E. Woodcock, and Michael A. Wulder. Good practices for estimating area and assessing accuracy of land change. *Remote Sensing of Environment*, 148:42–57, 2014. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2014.02.015. URL https://www.sciencedirect.com/science/article/pii/S0034425714000704.

[78] Ivan Zvonkov. Crop-mask github repository. https://github.com/nasaharvest/crop-mask, 2023. Accessed: 2023-03-31.