OpenMapFlow: A Library for Rapid Map Creation with Machine Learning and Remote Sensing Data

Ivan Zvonkov[1], Gabriel Tseng[2], Catherine Nakalembe[1], Hannah Kerner[3]

[1] University of Maryland, College Park
[2] McGill University and Mila – Quebec AI Institute
[3] Arizona State University
izvonkov@umd.edu, gabriel.tseng@mail.mcgill.ca, cnakalem@umd.edu, hkerner@asu.edu

Abstract

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. We hope this library will stimulate novel research in areas such as domain shift, unsupervised learning, and societally-relevant applications and lessen the barrier to adopting research methods for real-world tasks.

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 (Pradhan et al. 2018; Rolnick et al. 2022; Rasp, Pritchard, and Gentine 2018), food security (Wang, Azzari, and Lobell 2019; Kerner et al. 2020; Tseng et al. 2020), disasters (Bonafilia et al. 2020), and poverty (Xie et al. 2016). 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 (Sumbul et al. 2019), segmentation (Bonafilia et al. 2020), object detection (Christie et al. 2018) and diverse application areas such as agriculture (Tseng et al. 2021), disasters (Bonafilia et al. 2020), and land cover (Sumbul et al. 2019; Alemohammad and Booth 2020). 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 (Tuia, Persello, and Bruzzone 2016; Song et al. 2019), data fusion (Babaeian et al. 2021), 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 (Rolnick et al. 2022)).

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). The input samples used for generating the dense predictions are geographically contiguous and 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 (Tseng et al. 2021). 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 paper, we introduce OpenMapFlow (https://github.com/nasaharvest/openmapflow), an open-source python library for rapid map creation with ML and

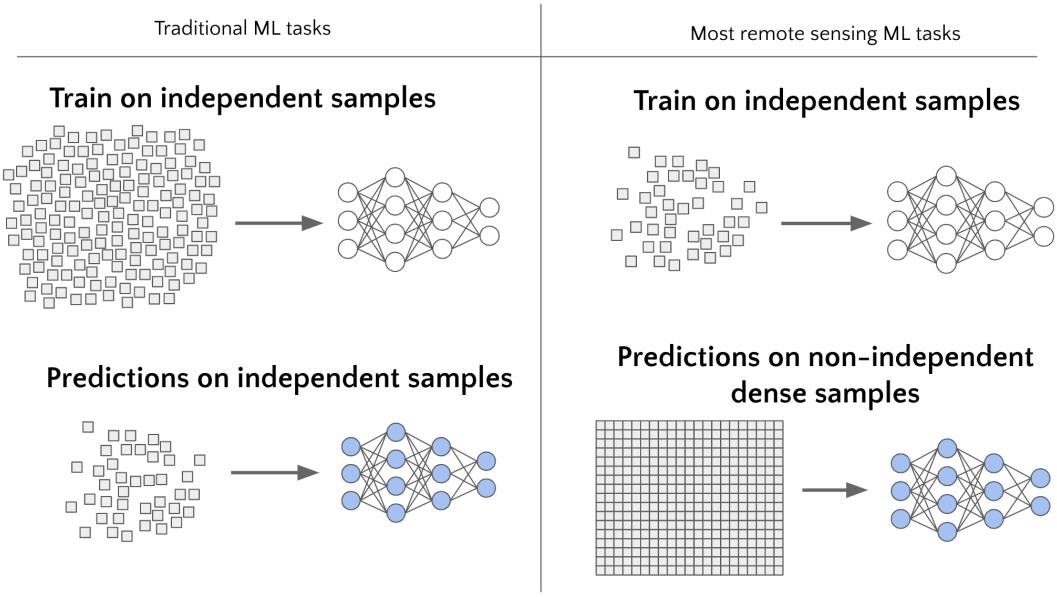

Figure 1: Sparse vs dense predictions

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 (https://nasaharvest.org/). 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 OpenMapFlow 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.

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 (Wagstaff 2012). 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 (Kumar et al. 2019). 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 (Hoiem, Chodpathumwan, and Dai 2012; Bolya et al. 2020). 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 (Shankar et al. 2017). 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 (Gomes, Queiroz, and Ferreira 2020), such as TorchGeo (Stewart et al. 2021), Radiant Earth MLHub (Alemohammad 2019), and CropHarvest (Tseng et al. 2021). 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 (Gorelick et al. 2017), SEPAL (FAO 2020), and Descartes Labs (Beneke et al. 2017) 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 (Gorelick et al. 2017) 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 (FAO 2020) 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 (Beneke et al. 2017) 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.

Reproducibility is another major limitation of prior work involving ML for remote sensing datasets (Gomes, Queiroz, and Ferreira 2020). The preprocessing steps applied to generate the remote sensing data inputs associated with the dataset labels are often not clearly described. This makes it especially difficult to use a trained off-the-shelf model to make predictions in a new region since there may be a significant distribution shift between the training and inference data. OpenMapFlow addresses this by providing the data processing pipeline for developing training and inference data in which the parameters for dataset construction are fully described.

Methods

OpenMapFlow aims to facilitate ML research with remote sensing datasets by providing an accessible and extensible workflow that allows rapid iteration.

Accessible Design 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 2) 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.

Rapid Iteration Ability to evaluate results and iterate quickly is critical to consistent progress in the practice of data science (Rudin 2019). 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.

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 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

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`

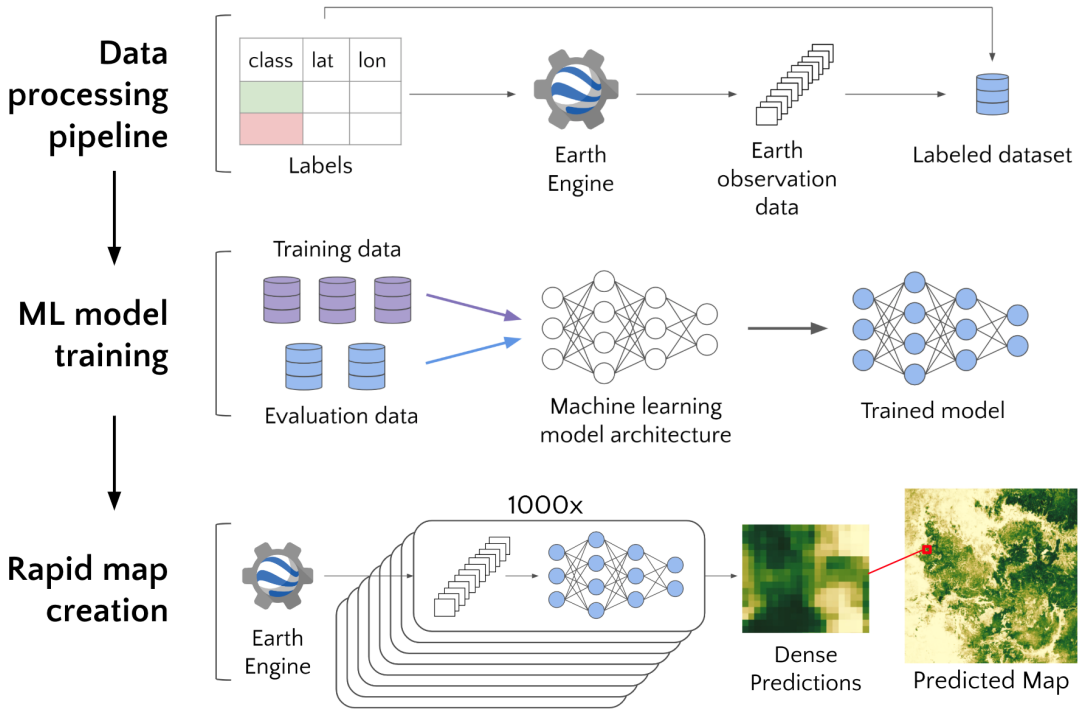

Figure 2: OpenMapFlow workflow

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 (Tseng et al. 2021) where each timestep contains multispectral 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.

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 (Paszke et al. 2019) 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 (Rußwurm and Körner 2020) implemented in PyTorch from the Time Series AI library (TSAI) (Oguiza 2022).

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.

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 1: Dense Predictions Time and Cost

- 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.

Experimental Results

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. (2020), which represents a traditional inference approach. These regions are a subset 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 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.

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 corre-

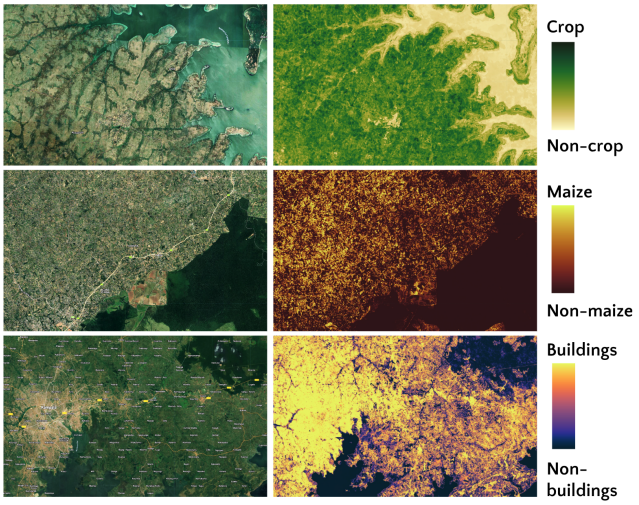

Figure 3: Predicted cropland map for region near Abagatchi, Togo (top), predicted crop type (maize) map for region in Kakamega County, Kenya (middle), predicted buildings map for Kampala, Uganda and surrounding region (bottom).

sponding datasets, evaluation metrics, and predicted geospatial maps. We describe each task below.

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 (Nakalembe et al. 2021). 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. (2020) and an additional 34,270 globally-distributed training labels obtained from the Geowiki dataset (See 2017).

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 (Nakalembe et al. 2021). For this task, we used 2,080 maize/non-maize labels from the CropHarvest dataset (Tseng et al. 2021) in Kenya (1,229 train, 438 val, 413 test).

Building Detection in Uganda Mapping buildings is useful for a variety of applications including population mapping, humanitarian response efforts, and urban planning (Sirko et al. 2021). For this task, we sampled 8,117 Uganda specific positive (building) labels from the Google Open Buildings dataset (Sirko et al. 2021), 13,993 negative (non-building) labels from the Geowiki dataset (See 2017) and 1,455 negative (non-building) labels from the Uganda Protected Planet dataset (UNEP-WCMC 2022) (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 and make dense predictions. Predicted maps were generated for each task's evaluation set region. Figure 3 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 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.

Discussion

Limitations The combination of remote sensing bands used in the CropHarvest dataset (Sentinel-1, Sentinel-2, ERA5, SRTM) (Tseng et al. 2021) 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 inputs and does not support datasets with spatial dimensions. While the temporal dimension contains the most relevant information for some remote sensing tasks (particularly for agriculture applications), the spatial dimension is also useful for many tasks. Spatial inputs (e.g., images or image patches) would enable the use of a wider array of deep learning models that use spatial convolutions. Another limitation is that OpenMapFlow is currently limited to binary labels and thus can only be used for binary classification tasks. 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

Task	Model Architecture	Batch size	Accuracy	F1-Score
Cropland classification	RNN (Rumelhart, Hinton, and Williams 1985)	64	0.76	0.70
Cropland classification	RNN	256	0.80	0.75
Cropland classification	LSTM (Hochreiter and Schmidhuber 1997)	64	0.73	0.67
Cropland classification	LSTM	256	0.74	0.65
Cropland classification	MLP (Wang, Yan, and Oates 2017)	64	0.78	0.69
Cropland classification	MLP	256	0.74	0.70
Cropland classification	ResNet (Wang, Yan, and Oates 2017)	64	0.80	0.73
Cropland classification	ResNet	256	0.79	0.74
Cropland classification	ResCNN (Zou et al. 2019)	64	0.77	0.69
Cropland classification	ResCNN	256	0.71	0.69
Cropland classification	Transformer (Rußwurm and Körner 2020)	64	0.76	0.69
Cropland classification	Transformer	256	0.79	0.71
Cropland classification	gMLP (Liu et al. 2021)	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 2: Test set metrics for cropland classification, building detection, and crop type classification example tasks.

the data limitations by including other data sources, allowing spatial data as input, and extending label types to include multi-class classification and regression. Due to the project's open-source nature, community contributions to address these limitations in future work are also possible.

Research Topics Motivated by Dense Predictions As discussed in the introduction and illustrated in Figure 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 understudied 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 (2021); Sun et al. (2021)), robustness and adaptation to multiple types of distribution shift (e.g., Guillory et al. (2021); Taori et al. (2020); Raghunathan et al. (2020)), 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 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.

Conclusion

We presented OpenMapFlow, a library for rapid geospatial map creation with machine learning and remote sensing data. OpenMapFlow was created to help make dense prediction an integral component of ML research using remote sensing data. We demonstrated how OpenMapFlow enables rapid start-up, experimentation, iteration, and deployment for machine learning projects using remote sensing datasets 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 has already been used to deploy state-of-the-art deep learning models to generate high-resolution, multi-year cropland and crop type maps in multiple countries, including Rwanda, Malawi, Namibia, and Tanzania. 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.

Acknowledgements

This research was supported by NASA Harvest (Award No. 80NSSC17K0625), SwissRe Foundation (Award No. 302916-00001), NASA GSFC/USAID (Award 3302915-00001), and NASA SERVIR (Award No. 80NSSC20K0264).

References

Alemohammad, H. 2019. Radiant MLHub: A repository for machine learning ready geospatial training data. In *AGU Fall Meeting Abstracts*.

Alemohammad, H.; and Booth, K. 2020. LandCoverNet: A global benchmark land cover classification training dataset. In *Proceedings of the Thirty-fourth Conference on Neural Information Processing Systems Workshops*.

Babaeian, E.; Paheding, S.; Siddique, N.; Devabhaktuni, V. K.; and Tuller, M. 2021. 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.

Beneke, C. M.; Skillman, S.; Warren, M. S.; Kelton, T.; Brumby, S. P.; Chartrand, R.; and Mathis, M. 2017. A Platform for Scalable Satellite and Geospatial Data Analysis. In *AGU Fall Meeting Abstracts*.

Bolya, D.; Foley, S.; Hays, J.; and Hoffman, J. 2020. TIDE: A General Toolbox for Identifying Object Detection Errors. In *ECCV*.

Bonafilia, D.; Tellman, B.; Anderson, T.; and Issenberg, E. 2020. 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*.

Christie, G.; Fendley, N.; Wilson, J.; and Mukherjee, R. 2018. Functional map of the world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6172–6180.

Deng, W.; and Zheng, L. 2021. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15069–15078.

FAO. 2020. SEPAL Repository. https://github.com/openforis/sepal/. Accessed: 2022-07-03.

Gomes, V. C.; Queiroz, G. R.; and Ferreira, K. R. 2020. An Overview of Platforms for Big Earth Observation Data Management and Analysis. *Remote Sensing*, 12(8): 1253.

Gorelick, N.; Hancher, M.; Dixon, M.; Ilyushchenko, S.; Thau, D.; and Moore, R. 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 202: 18–27.

Guillory, D.; Shankar, V.; Ebrahimi, S.; Darrell, T.; and Schmidt, L. 2021. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1134–1144.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hoiem, D.; Chodpathumwan, Y.; and Dai, Q. 2012. Diagnosing error in object detectors. In *Proceedings of the European Conference on Computer Vision*, 340–353. Springer.

Kerner, H.; Tseng, G.; Becker-Reshef, I.; Nakalembe, C.; Barker, B.; Munshell, B.; Paliyam, M.; and Hosseini, M. 2020. Rapid Response Crop Maps in Data Sparse Regions. In *Proceedings of the ACM SIGKDD Conference on Data Mining and Knowledge Discovery Workshops*.

Kumar, R. S. S.; Brien, D. O.; Albert, K.; Viljöen, S.; and Snover, J. 2019. Failure modes in machine learning systems. *arXiv preprint arXiv:1911.11034*.

Liu, H.; Dai, Z.; So, D.; and Le, Q. V. 2021. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34: 9204–9215.

Nakalembe, C.; Justice, C.; Kerner, H.; Justice, C.; and Becker-Reshef, I. 2021. Sowing seeds of food security in africa. *Eos*, 102.

Oguiza, I. 2022. tsai - A state-of-the-art deep learning library for time series and sequential data. https://github.com/timeseriesAI/tsai. Accessed: 2022-05-10.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Pradhan, R.; Aygun, R. S.; Maskey, M.; Ramachandran, R.; and Cecil, D. J. 2018. Tropical Cyclone Intensity Estimation Using a Deep Convolutional Neural Network. *IEEE Transactions on Image Processing*, 27(2): 692–702.

Raghunathan, A.; Xie, S. M.; Yang, F.; Duchi, J. C.; and Liang, P. 2020. Understanding and Mitigating the Tradeoff between Robustness and Accuracy. In *Proceedings of the 37th International Conference on Machine Learning*.

Rasp, S.; Pritchard, M. S.; and Gentine, P. 2018. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39): 9684–9689.

Rolnick, D.; Donti, P. L.; Kaack, L. H.; Kochanski, K.; Lacoste, A.; Sankaran, K.; Ross, A. S.; Milojevic-Dupont, N.; Jaques, N.; Waldman-Brown, A.; et al. 2022. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2): 1–96.

Rudin, C. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5): 206–215.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

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

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

Shankar, S.; Halpern, Y.; Breck, E.; Atwood, J.; Wilson, J.; and Sculley, D. 2017. 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*.

Sirko, W.; Kashubin, S.; Ritter, M.; Annkah, A.; Bouchareb, Y. S. E.; Dauphin, Y.; Keysers, D.; Neumann, M.; Cisse,

M.; and Quinn, J. 2021. Continental-scale building detection from high resolution satellite imagery. *arXiv preprint arXiv:2107.12283*.

Song, S.; Yu, H.; Miao, Z.; Zhang, Q.; Lin, Y.; and Wang, S. 2019. Domain adaptation for convolutional neural networks-based remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 16(8): 1324–1328.

Stewart, A. J.; Robinson, C.; Corley, I. A.; Ortiz, A.; Ferres, J. M. L.; and Banerjee, A. 2021. TorchGeo: deep learning with geospatial data. *arXiv preprint arXiv:2111.08872*.

Sumbul, G.; Charfuelan, M.; Demir, B.; and Markl, V. 2019. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, 5901–5904. IEEE.

Sun, X.; Hou, Y.; Deng, W.; Li, H.; and Zheng, L. 2021. Ranking Models in Unlabeled New Environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11761–11771.

Taori, R.; Dave, A.; Shankar, V.; Carlini, N.; Recht, B.; and Schmidt, L. 2020. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33: 18583–18599.

Tseng, G.; Kerner, H. R.; Nakalembe, C. L.; and Becker-Reshef, I. 2020. 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*.

Tseng, G.; Zvonkov, I.; Nakalembe, C. L.; and Kerner, H. 2021. CropHarvest: A global dataset for crop-type classification. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Tuia, D.; Persello, C.; and Bruzzone, L. 2016. Domain adaptation for the classification of remote sensing data: An overview of recent advances. *IEEE Geoscience and Remote Sensing Magazine*, 4(2): 41–57.

UNEP-WCMC. 2022. Protected Area Profile for Uganda from the World Database on Protected Areas, August 2022. www.protectedplanet.net. Accessed: 2022-08-01.

Wagstaff, K. L. 2012. Machine Learning That Matters. In *Proceedings of the 29th International Conference on Machine Learning*, 1851–1856.

Wang, S.; Azzari, G.; and Lobell, D. B. 2019. Crop type mapping without field-level labels: Random forest transfer and unsupervised clustering techniques. *Remote Sensing of Environment*, 222: 303–317.

Wang, Z.; Yan, W.; and Oates, T. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, 1578–1585. IEEE.

Xie, M.; Jean, N.; Burke, M.; Lobell, D.; and Ermon, S. 2016. Transfer learning from deep features for remote sensing and poverty mapping. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Zou, X.; Wang, Z.; Li, Q.; and Sheng, W. 2019. Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification. *Neurocomputing*, 367: 39–45.