# Small is Beautiful: Distributed Orchestration of Spatial Deep Learning Workloads

Daniel Rammer, Kevin Bruhwiler, Paahuni Khandelwal,
Samuel Armstrong, Shrideep Pallickara, Sangmi Lee Pallickara
Colorado State University
Fort Collins, Colorado
rammerd@rams.colostate.edu,Kevin.Bruhwiler@rams.colostate.edu,paahuni@colostate.edu,
Sam.Armstrong@rams.colostate.edu,Shrideep.Pallickara@colostate.edu,Sangmi.Pallickara@colostate.edu

## Abstract

Several domains such as agriculture, urban sustainability, and meteorology entail processing satellite imagery for modeling and decision-making. In this study, we describe our novel methodology to train deep learning models over collections of satellite imagery. Deep learning models are computationally and resource expensive. As dataset sizes increase, there is a corresponding increase in the CPU, GPU, disk, and network I/O requirements to train models. Our methodology exploits spatial characteristics inherent in satellite data to partition, disperse, and orchestrate model training workloads. Rather than train a single, all-encompassing model we facilitate producing an ensemble of models - each tuned to a particular spatial extent. We support query-based retrieval of targeted portions of satellite imagery including those that satisfy properties relating to cloud occlusion, We validate the suitability of our methodology by supporting deep learning models for multiple spatial analyses. Our approach is agnostic of the underlying deep learning library. Our extensive empirical benchmark demonstrates the suitability of our methodology to not just preserve accuracy, but reduce completion times by 13.9x while reducing data movement costs by 4 orders of magnitude and ensuring frugal resource utilization.

## Keywords

Deep Learning, Spatial Data, Workload Orchestration

## 1 Introduction

Advances in remote sensing and increases in the number of satellites have contributed to a proliferation of satellite data. In particular, there have been sustained improvements in the precision, resolution, and frequency at which these remote sensing observations have been performed. These data offer opportunities to inform analyses in several domains such as agriculture, land-use/land-change analyses, urban systems, and forest fires.

Measurements reported by satellites are expressed as a collection of optical imagery comprising multiple bands - one for each interval of wavelengths for which surface reflectance is measured using onboard instrumentation. Satellite data are geo-encoded with each pixel representing measurements at a particular spatial extent. The spatial extent represented by a pixel is the *resolution* - this is 30m x 30m for LANDSAT and 250-1000m for MODIS. The revisit interval (or temporal resolution) for remote sensing observations performed by these satellites are also different.

Satellite data collections are voluminous and outpace the capacities of a single machine. In particular, these collections entail distributed and concurrent data processing. The workloads we consider are deep learning workloads. Satellite data processing using deep learning has similarities with computer vision applications, but the nature of satellite data introduces several challenges and opportunities. Deep learning also offers opportunities to extract non-linear patterns from satellite data and inform analyses.

The crux of this paper is to support effective data management and model training over large satellite data collections. To this end, we explore the orchestration of tasks in a distributed, cluster-based environment. Our notion of effectiveness encompasses concurrency to facilitate timely completion of tasks, utilization of available resources including coprocessors and scaling to ensure high throughput. Another consideration is effectiveness in extracting patterns from the datasets; the measures we consider here include the error of regression models and representation loss for larger, composite images from GANs.

### 1.1 Challenges

Effective processing and orchestration of deep learning workloads over large satellite data collections introduces several challenges. In particular, these challenges include:

(1) Volumes: Satellite data are voluminous and exceed hard disk capacities available on a single disk.
(2) Variety: Satellite data are made available in different formats (GeoTIFF, HDF) and at different resolutions. Furthermore, the number of bands available varies depending on the satellite. For example, there are 13 available bands in Sentinel while MODIS supports 36 bands.
(3) Variability: There is variability in the availability of data. These may be due to occlusions caused by atmospheric phenomena (clouds) or on-board instrumentation errors or post-processing issues.
(4) Multiple bands. Satellite data processing encapsulates multiple bands that may be sensed at different resolutions.
(5) Complexity of images is different in different portions of the image. Extracting patterns from the data is subject to not just regional variations but also temporal/seasonal variations.

### 1.2 Research Questions

The overarching theme of this study is the effective storage, processing, and learning from large satellite data collections. Specific research questions that we explore include:

**RQ-1:** *How can we amortize data management and processing workloads?* These have implications for CPU, memory, and I/O footprints. For example, training a particular deep network in a distributed setting may trigger network I/O.

| Satellite | Format | Spatial Reference System | Scan Frequency | Coverage | Pixel Resolution | Band Count |
|---|---|---|---|---|---|---|
| MODIS | hdf | Sinusoidal | 1-2 Days | Global | 200m | 7 |
| | | | | | 500m | 7 |
| NAIP | zip / jp2 | Transverse Mercator | 2 Years | United States | 1m | 4 |
| NLCD | img | Albers Equal-Area Conic | 3 Years | United States | 30m | 1 |
| Sentinel-2 | SAFE | Transverse Mercator | 5 Days | Global | 10m | 6 |
| | | | | | 20m | 4 |
| | | | | | 60m | 3 |

**Table 1:** Satellite imagery dataset definitions. Our system currently supports MODIS, NAIP, NLCD, and Sentinel-2 imagery; enabling support for reconciliation between diverse formats, spatial reference systems, pixel resolutions, band types, and dimensionality.

**RQ-2:** *How can we extract patterns from the data?* The patterns we consider cross-cut spatiotemporal scopes.
**RQ-3:** *How can we scale with increases not just in data volumes, but also increases in the diversity and availability of data?*

## 1.3 Approach Summary

We posit that different characteristics of the observational space are accentuated by different features at different spatial scopes. As such different regions can benefit from customized processing and refinements. This entails partitioning images and extracting patterns from the smaller spatial extents that can then be aggregated and combined. This approach is amenable to distribution and scaling.

Our methodology partitions satellite data across all bands based on their spatial extents for analysis, reconstruction, and refinement. Spatial extents have geocodes associated with them. These geocodes are computed deterministically and the precision of these geocodes correspond to geographical extents. We use geocodes to collocate images from proximate geographical extents on the same machine. A single machine is responsible for data from multiple spatial extents. The size of the spatial extents is informed by the memory footprints and computational overheads.

Satellite optical imagery is partitioned as they are ingested. The partitioning scheme is based on spatial extents and is consistent across all available bands. Our underlying substrate is based on a DHT (distributed hash table) scheme to ensure decentralized dispersion and management of partitioned images.

We organize metadata and ancillary information to support queries. As images are partitioned during the ingestion process and staged within the DHT we extract metadata associated with them. The metadata for each partitioned image includes those that are inherited from the original image and those that are computed from the optical imagery. Metadata inherited from the original image include satellite information, available bands, per-band resolutions, and chronological information. Metadata that is recalibrated include spatial extents for which the data is available. These metadata are complemented with additional metadata relating to occlusion (cloud cover, unavailability) that we compute. We allow users to identify useful datasets for model training while constructing predicates based on particular satellites, cloud cover, bands, spatial extents, etc.

We support several data wrangling operations that are aligned with the needs of satellite data processing. The data wrangling operations include support for reconciling data formats (GeoTIFF, HDF). We also support creation of composite images from contiguous spatial regions and also from contiguous temporal segments from the same spatial extent. The latter scheme is useful for cases where we need to construct images that are not occluded or transform satellite images into tensors (multidimensional arrays) based on the prescribed dimensionality. These tensors are amenable for use as inputs in different learning algorithms. Additionally, we support the model creation and refinement process through support for several operations, including transfer learning.

Queries and data wrangling are used to identify relevant data and facilitate custom datasets for model training. This includes support for creating composite images from data based on temporal proximity to reduce occlusion/sparsity issues. We use the spatial partitioning schemes to inform orchestration of training workloads. In particular, our collocation properties ensure data locality for a given spatial extent. Furthermore, when training different models for the same spatial extent we support pinning the corresponding tensor in memory for different types of analyses. This reduces duplicate preprocessing and disk I/O overheads. We also identify spatial extents that are likely to benefit from transfer learning based on the extracted metadata. This significantly reduces model training times. We demonstrate that our methodology is amenable to containerization using frameworks such as Docker and Kubernetes, allowing it to integrate easily into modern cloud computing clusters. Additionally, we observe substantial performance benefits provided by containerization.

## 1.4 Paper Contributions

In this study we have designed a framework to process satellite optical imagery at scale. In particular, our contributions include:

(1) Effective reconciliation of differences in satellite data formats, resolutions across bands, and sparsity/occlusion across different spatiotemporal scopes.
(2) Effective extraction of patterns in spatiotemporal phenomena and scalable orchestration of workloads (storage and processing) over large satellite data collections.
(3) Effective utilization of resources, leveraging co-processors and frugal network I/O while minimizing duplicate work.
(4) A deep learning library agnostic framework to process large satellite data collections; we interoperate with several popular libraries (PyTorch, TensorFlow) and outperform their own native distributed implementations.

## 1.5 Paper Organization

The remainder of this paper is organized as follows. Section 2 describes our methodology, including data staging and partitioning, querying schemes, workload orchestration, sample analyses, and transfer learning. Section 3 covers systems benchmarks, including the experimental setup, an analysis between model training methodologies, and the performance of our varied analyses. Other work related to satellite data partitioning, deep learning orchestration, and some applications of satellite data is addressed in Section 4.

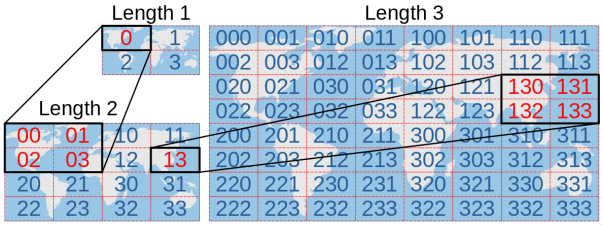



**Figure 1:** An example of partitioning the globe using quad-tiles. Each additional character produces 4 subregions.

Finally, Section 5 summarizes the conclusions and future research directions.

## 2 Methodology

Our methodology to orchestrate satellite data processing while ensuring effective resource utilizations includes a carefully calibrated mix of:

(1) Data partitioning and staging within a distributed cluster
(2) Support for queries to identify/explore data of interest and data wrangling operations at diverse spatiotemporal scopes
(3) Decentralized orchestration of model training workloads
(4) Profiling effectiveness of our methodology with support for different analytic operations

Table 1 provides a compilation of datasets that we currently support. Our methodology leverages design patterns to abstract and simplify integration of satellite data collections. Each implementation of the abstraction is customized primarily to handle parsing image formats and metadata retrieval for a particular collection. All image manipulation code is agnostic of image format, number and type of bands, pixel resolution, and spatial reference system. This facilitates seamless support for additional image datasets.

### 2.1 Partitioning and Staging Within a Cluster [RQ-1 / 3]

Partitioning and staging data is a precursor to model construction and pattern extraction across spatial scopes. Inefficiencies in partitioning and staging entail data movements. The costs associated with data movement are expensive as they incur disk I/O (during reads and writes) and network I/O to complete the transfers. To be effective these must be decentralized, deterministic, and performed without centralized coordination. Given the model building is iterative, the impact of these inefficiencies are amplified.

A consideration when we deal with hyperspectral imagery is their size, which may induce resource pressure. Hyperspectral imagery (comprising multiple bands) fed into models result in tensors being created and exchanged between different layers. The size of these tensors and the memory pressure they induce is correlated with the size of the input.

Our methodology to facilitate partitioning and staging data within the cluster involves two key steps that are aligned with the overall goal to facilitate model creation: (1) preprocessing the images to reconcile formats, spatial reference systems, and any alignment issues that these involve to address boundary conditions, and (2) ensuring that the deterministic, decentralized, and colocation properties are preserved during dispersion.

Rather than build an all-encompassing model we build an ensemble of deep learning models each tuned to the particular spatial extent. Partitioning must be aligned with expected model training workloads; in particular, it must ensure that we can preserve data locality (all data from a given spatial extent must be colocated) while at the same time ensuring that storage and processing workloads are balanced.

The partitioning phase encompasses steps to ensure discovery, load balancing, timeliness, and throughput of operations. These operations include pre-processing of images to reconcile projections, formats, etc. We harness two distinct spatial partitioning schemes to support the desired objectives: geohashes and quad-tiles. Both these algorithms rely on hierarchically partitioning the globe into smaller spatial extents. We use the overarching term geocode to refer to spatial extents represented using either geohash or quad-tiles. The geocodes represented by geohashes and quad-tiles are represented as 1-dimensional strings. The greater the length (or precision) of the string, the smaller the spatial extent represented by the geocode. The choice of the algorithm is mostly influenced by the input dataset and types of analyses. The spatial reference system within the imagery also plays a role in this choice: geohashes work with <latitude, longitude> bounding boxes while quad-tiles leverage Mercator projections. Another consideration is the granularity of the hierarchical splits as the length of the geocode increases. Geohashes produce 32 subregions at each level, while quad-tiles result in 4 subregions. An example of the quad-tile algorithm is provided in Figure 1. We leverage the appropriate geocoding algorithm to split images along spatial boundaries.

We use a multi-token distributed hash table (DHT) to disseminate geocode-based image splits throughout the cluster. We support dispersion based on any substring length of the geocode. For example, processing length k ensures that all images beginning with the same k characters are collocated. Alternatively, using geocode length k-1 disperses images so that data from spatially proximate subregions are collocated. The geocodes are passed through a cryptographic hashing function (SHA-1) and each DHT node is responsible for a contiguous portion of the cryptographic hash space. The dispersion properties of the cryptographic hash function ensure that the load is uniformly dispersed over the cluster. In particular, this allows for fine-grained and dynamic load balancing of storage workloads within the system. This, in turn, ensures that subsequent pre-processing and model training workloads are dispersed and dynamically balanced as well.

During partitioning, as each hyperspectral image comprising multiple bands is being ingested, we extract metadata from the images and organize them so that they are amenable to query evaluations. Our dispersion scheme allows us to load balance storage, query evaluation, and model training workloads. Once the spatial granularity is finalized, the geocode and the DHT node responsible can be calculated deterministically and without coordination. Furthermore, since all data for a particular geocode are stored on the same DHT node our methodology ensures colocation of data from spatial regions across different temporal ranges on the same machine. A consequence is that our methodology supports targeted processing of fine-grained spatiotemporal extents. Model training in distributed environments often entail substantial data movements as data are pulled from multiple nodes. Since we ensure colocation when training models for a particular spatiotemporal extent, we substantially alleviate the adverse impact of network I/O during data transfers.

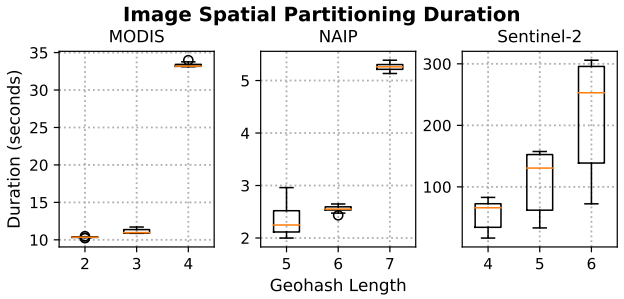



**Figure 2:** Duration to partition and distribute satellite imagery at different geohash precisions. The number of bands and their accompanying resolutions are a significant impact on performance.

**Microbenchmark:** In Figure 2 we performed a series of experiments profiling raw-image processing, including spatial partitioning and distribution. We chose 3 unique geohash lengths for each dataset based on image pixel resolutions – with all these datasets there is a point at which image resolutions become too small and the utility diminished. The first observation is the relative difference in processing time between datasets. This may be attributed to the difference in the number of bands and unique resolutions within each image. Next, we see a difference in variance among processing times between datasets. We noticed a strong correlation between file sizes and processing time. MODIS and NAIP images are typically uniform across observations. Sentinel-2's higher bands and varying resolution introduce variances. Consequently, decompression, parsing, and data transfer speeds are higher for Sentinel-2.

**Microbenchmark:** To highlight the effectiveness of our DHT we staged a satellite data collection and report on the distribution of image geocode splits. In this experiment, our base dataset comprises all available Sentinel-2 images for the Continental United States during July 2019, roughly 6,400 images. We staged this collection by partitioning the data using geohashes of length 5, producing bounding regions approximately 5km x 5km. Table 2 profiles the effectiveness of our distribution scheme using a variety of metrics. We present dataset size, or the aggregate size of all images at a specific node, and image count, which is the total number of images at a specific node. As can be seen, the standard deviation are quite small for each metric indicating balanced data distribution.

## 2.2 Support for Queries [RQ-1 / 2]

To support effective query evaluation as partitioned imagery is ingested within our system we: (1) extract and organize metadata, (2) support a rich set of queries, and (3) leverage structural properties of the DHT and data structures that we use to organize the metadata to effective support query evaluations. In particular, our query evaluations are effective due to search space reductions, distributed evaluations, and fast traversal of data structures during evaluations.

*2.2.1 Metadata Extraction* As partitioned images are ingested, we extract and organize metadata to support fast query evaluations. Image attributes are typically drawn from source metadata, which includes spatial and temporal ranges, dataset identifiers, etc. We supplement these attributes with additional metadata relevant for building spatial deep learning models. In particular, we compute two additional attributes as satellite imagery are ingested into the system: pixel coverage and cloud coverage. Pixel coverage is the percentage of pixels within the image (across bands) that contain valid information. Source images may misalign with geocode spatial bounds. As a result, spatially partitioned images may contain partial information with padded 'fill' value pixels. This may be seen in Figure 3, which shows four separate NAIP images with less than 100% pixel coverage percentages combined into a full image. Cloud coverage refers to the percentage of pixels that are identified as clouds within an image. Clouds result in occlusions, and their opacity preclude analyses for the areas that they impact. Therefore, for each pixel we use a likelihood estimate to identify the probability of cloud coverage. We make aggregate information available for filtering as a percentage of cloud coverage within an image. This is particularly useful for applications requiring spatial imagery that may be subject to extremes, such as vegetation covers, urban areas, deserts, etc.

| Metric | Mean | Standard Deviation |
|---|---|---|
| Dataset Size | 139.5GB | 1.5GB |
| Image Count | 255k | 2.8k |

**Table 2:** 50 machine cluster distribution metrics performed on 6,400 raw Sentinel-2 images partitioned at geohash length 5. Our DHT provides load-balanced distribution evaluated under dataset sizes and image counts.

*2.2.2 Query Support* Fitting deep learning models to satellite imagery benefit from inclusion/exclusion of certain portions of the dataspace. We support expressive queries aligned with the characteristics of the data to support model construction over the underlying dataspace. This becomes increasingly important when considering the iterative nature of the modelling process which may involve diverse temporal queries for a particular spatiotemporal scope. Challenges in efficiently filtering large datasets are exacerbated in distributed environments where approaches based on distributed indexing and cooperative query evaluation have been studied. Distributed data indices are expensive in terms of both memory and computation. Additionally, solutions where the indices and the underlying data are resident on disparate nodes introduces overheads stemming from indirections and network transfers. Cooperative query evaluation introduces processing inefficiencies with nodes participating in queries even when they do not have relevant data.

Our query evaluations are distributed and avoid wasteful processing. First, any node within our DHT can serve as the conduit for query evaluations. Second, at a given node the subset of DHT nodes that must be contacted is deterministic and computed without having to coordinate/gossip with other nodes in the system. As a result, we reduce search space during query evaluations. These search space reductions ensure that only nodes holding relevant content are involved in query evaluation and avoid wasteful processing at other nodes within the system.

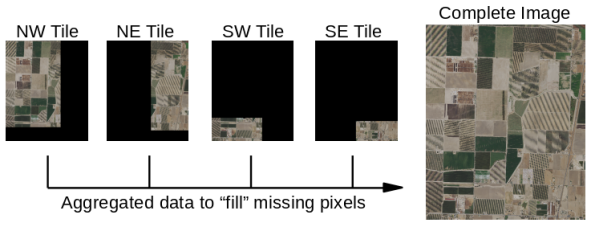
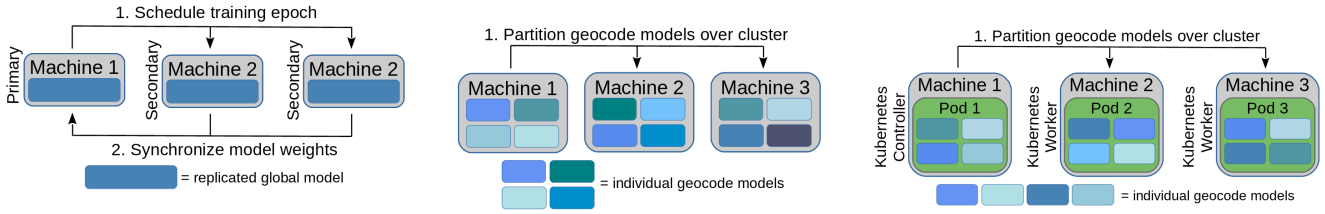



**Figure 3:** An example of producing a complete image for a spatiotemporal scope where multiple image partitions exist with incomplete pixel coverage.




**(a) Distributed architecture.** **(b) Partitioned Architecture.** **(c) Containerized Architecture.**

**Figure 4:** The three training architectures used. (a) is a typical master model setup, provided by most libraries, (b) and (c) are variations on our methodology.

At each node comprising our DHT, we use B+-Trees for efficient range-based queries over data features, including images timestamps, cloud coverage, and pixel coverage. We also employ Radix Trie's for fast (and recursive) geocode prefix matching, meaning we can quickly identify regions starting with a given character string. Finally, we perform query evaluations in parallel on each cluster node; results from the nodes where the queries are evaluated are streamed to ensure that data retrieval is not a bottleneck.

We support queries involving predicates that span the following characteristics.

- **Chronological range based on open or closed bounds:** Filter images by temporal bounds, either end may be open.
- **Spatial range:** Identify all images within the given bounding spatial bounding box. This query may be recursive (i.e. all subregions) or not (i.e. only the queried region).
- **Cloud coverage:** Retrieve imagery where the occlusion due to clouds is above the specified threshold.
- **Pixel coverage:** Source images may not provide data for the spatial extent represented by a geocode. Pixel coverage refers to percentage of image representing valid data.
- **Dataset:** Given that the system supports a variety of data sources, users may specify the satellite imagery of interest: MODIS, NAIP, NLCD, or Sentinel-2.

*2.2.3 Complementing Dataspace Queries* We also support data wrangling operations that complement our dataspace queries. Specifically, they remedy situations where simple filtering operations are not powerful enough to identify and retrieve complex data subsets. For example, co-processing of multiple image sources where each has different spatiotemporal granularity's and pixel resolutions. We logically partition these operations into three subsets, namely bolstering filtering criteria, reconciling image extents, and aligning data from diverse spatiotemporal scopes.

The first capability we provide is to combine temporally- proximate data for the same spatial region to construct a synthetic image that satisfies pixel or cloud coverage properties. Since each source image relates to a specific spatial bound, splitting a collection of source imagery by geocodes may result in incomplete images for the same spatiotemporal scope. We are able to efficiently identify collections of images that share a spatiotemporal scope and for which the pixel/cloud coverage satisfies the queries. By combining portions from different hyperspectral images we are able to dynamically splice together an image with the desired coverage. This construct is depicted in Figure 3, where four partial images from the same spatiotemporal extent that are aggregated to produce a complete image. These temporal reconciliations of the dataspace ensure a complete dataset and are useful for effective deep learning over satellite imagery.

Another capability we support is tensor dimensionality reconciliation across bands that comprise the hyperspectral image. As seen in Table 1, images from different bands for the same scan may be at different resolutions. In some cases, we may also attempt to perform analyses on data from bands from different satellites. In each case, the tensors (multidimensional arrays) that serve as inputs during processing must be at a desired dimensionality. To solve this issue, we provide image up-sampling and down-sampling functionality which amalgamates data with contrasting resolutions.

## 2.3 Orchestration of Workloads [RQ-1 / 2 / 3]

Effective workload orchestration and concurrent execution are necessary to harness resources within a commodity cluster. Orchestration of distributed spatial deep learning tasks introduces a number of challenges. First, deep learning operations depend on every layer of the resource hierarchy (i.e. CPU, GPU, RAM, disk, and network) and varying workflow stages have different resource requirements. Second, the model building process is iterative, involving synchronization barriers as model weights are aggregated after each epoch. Given the speed differential across the memory and I/O hierarchy, network I/O costs start to dominate completion times. Finally, right-sizing resource allocations is difficult. Under-provisioning leaves system resources unused, while over-provisioning may further prolong completion times due to excessive disk head movements, CPU context switches, memory pressure, and I/O interference.

Our orchestration (see Figure 4) based on decentralized DHT leveraging traits aligned with the characteristics of the spatial workloads. Specifically, we train a single model for each individual geocode within the system, producing an ensemble of smaller models rather than a large, all-encompassing model. This allows us to reconcile spatial heterogeneity by having smaller regions tune themselves to particular spatial extents; our performance benchmarks (see empirical benchmarks in section 3.4). It also permits us to use the DHT's balanced distribution of geocodes as a guide for orchestrating workloads. In addition to balanced evaluation, it ensures training jobs are scheduled with data locality, avoiding unnecessary network transfers and coordination-related overheads.

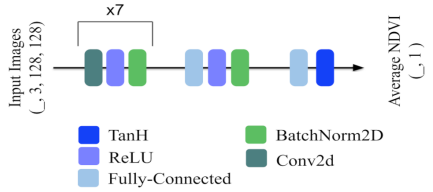



**Figure 5:** The architecture for the NDVI model. It conforms to a typical convolutional structure and the output is a single value.

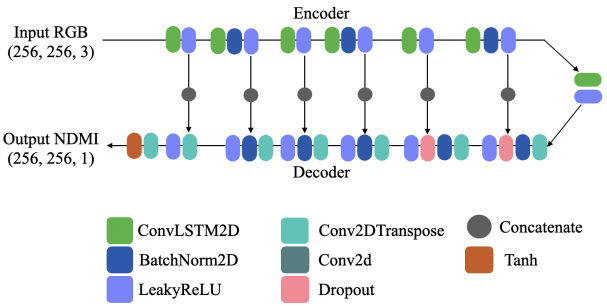



**Figure 6:** The architecture of model used for generating NDMI maps.

We also demonstrate that our workload orchestration scheme is amenable to containerized environments. We containerize the process of training an individual model using Docker [1] and schedule the execution of model training using Kubernetes [2]. We show that Kubernetes is able to take advantage of our partitioning scheme to train multiple data-local models on each machine simultaneously, further optimizing resource use.

## 2.4 Leveraging our Methodology in Different Analyses [RQ-2]

As part of this study, we also profile the suitability of our methodology for diverse deep learning networks. The networks were chosen based on the types of spatial analyses that are typically performed, and also the representativeness of the structural elements such as depth of the network, layers, regularization schemes, and the types of network. We profile our methodology's impact on training time, resource consumption, and statistical error of models with varying sizes and requirements. These real-world use cases are: (1) Calculation of metrics across bands in hyperspectral imagery, (2) Image translations with preservation of structural characteristics such as roads, buildings, etc. using skip connections. (3) Cloud removal from satellite imagery - a relatively new image-to-image translation using spatial attention in a generative adversarial network.

*2.4.1 Average NDVI Prediction* Our first use-case uses non-linear regression to predict the average Normalized Difference Vegetation Index (NDVI) from the RGB bands of satellite images. NDVI is a metric quantifying the amount of vegetation in remote-sensing imagery, and is commonly used to estimate the ecosystem health [3] or the emissivity [4] (the ability to emit thermal radiation, essential in climate modeling) for different spatial extents. Additionally, NDVI is subject to high spatial heterogeneity (neighboring regions are likely to have similar NDVI, regions far apart are not). Consequently, NDVI-based regression is both a highly relevant potential use-case for many applications and well-suited to our methodology. The non-linear regression is done with a neural network consisting of several cells of convolutional layers and activation functions followed by two fully-connected layers (see Figure 5). The model is intended to be representative of a common image analysis network and has not been specifically optimized for NDVI prediction.

*2.4.2 Image-to-Image Translational Models* We also trained an image-to-image translation model to generate Normalized Difference Moisture Index (NDMI) maps using reflectance values of R, G, and B bands at 10m spatial resolution. NDMI is calculated using the refracted radiation in short-wave infrared (SWIR) and near-infrared

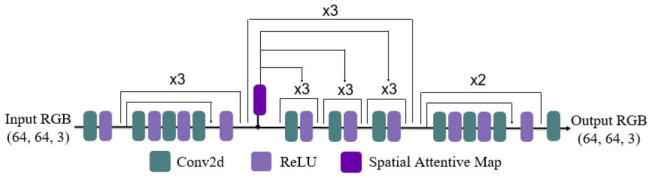



**Figure 7:** The architecture for the SpAGAN generator.

(NIR) bands, and can be used to measure the water-stress level of crops.

Calculating NDMI can be tricky for satellite sensors such as NAIP, as they don't capture the SWIR / NIR reflectance values that are critical in calculating the amount of water present in the plants. To overcome this issue, we have trained a convolutional neural network (CNN) model to estimate NDMI maps using RGB bands captured by the majority of the satellite sensors. The U-net architecture of the model is described in Figure 6. This image-to-image translation model is a representative use case since such models are often used to translate images from black-and-white to colored, aerial photos into cartographic maps [5] and so on. The model comprises blocks of 2D convolutional, batch normalization, and LeakyReLU layers. Each block sequentially extracts high-level features and reduces the dimensionality of the inputs. The decoder part of the network consists of a Conv2DTranspose layer to connect the input feature space with the output feature space at each block level. This helps in preserving the underlying land and road structures.

*2.4.3 Generating Cloud Free Images with a Spatial Attention GAN* Finally, we use a spatial attention generative adversarial network (SpAGAN) to remove cloud occlusions from the satellite imagery. Cloud coverage in satellite imagery can make collecting information for NDVI, NDMI, and other geospatial metrics difficult as clouds can obscure large portions of satellite imagery. About 67% of earth is typically covered by clouds at any given moment [6]. By removing cloud coverage and accurately generating the landscape underneath we are able to improve the quality of these geospatial metrics. The architecture of the SpAGAN generator can be seen in Figure 7. The generator comprises of three standard residual blocks, three spatial attentive blocks, two more standard residual blocks, and a final convolutional layer. The SpAGAN discriminator comprises of six convolutional blocks as illustrated in Figure 8.

By querying for satellite images with zero cloud coverage and pairing them with the most temporally proximate image with cloud coverage, we are able to create cloudy-cloudless pairings for our models. After training our SpAGAN models we are able to compare the quality of their generated cloud-free images using the peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM). PSNR and SSIM are both metrics commonly used in measuring the

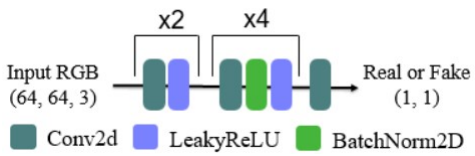



**Figure 8:** The architecture for the SpAGAN discriminator.

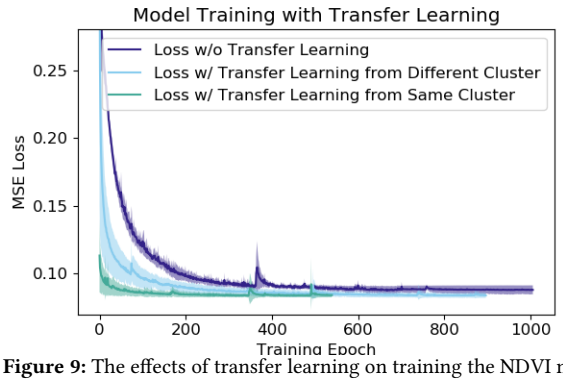



**Figure 9:** The effects of transfer learning on training the NDVI model.

quality of an image after compression, but in our case they are good measures of the quality of generated cloud-free images. PSNR is the ratio between the maximum possible power of a signal (255 for pixels) and the power of corrupting noise (mean squared error) between two images. SSIM measures the similarity in luminance, contrast, and structure between two images using a sliding window.

## 2.5 Transfer Learning [RQ-1 / 3]

Transfer learning is a broadly applied paradigm in deep learning. It involves taking a model trained on one dataset and applying it to a related task, and it serves to produce more accurate models in a shorter time, with significantly reduced resource requirements. We posit that regions with similar textural qualities will benefit from transfer learning.

Our solution applies the gray level cooccurrence matrix (GLCM) algorithm to identify similar regions based on geocode bounds. GLCM processes gray-scaled image pixels to produce a matrix capturing neighboring pixel relationships. It is a widely accepted practice to quantify textural qualities of an image. Subsequent analyses are often facilitated by calculating a variety of features on the GLCM; these include dissimilarity, homogeneity, contract, energy, ASM, and correlation.

Given textural features for each geohash we quantify spatial region similarity using the k-means clustering algorithm. GLCM feature matrices are quite large, in our application each feature produces 3 unique 255 x 255 byte structures. To improve the accuracy of k-means, which typically struggles with large-dimensionality data, we compute averages over the main matrix axes. The result is a vector of 18 values, each of which captures qualities of a single GLCM feature. Finally, we dissected the elbow in the cluster silhouette score plot (using multiple distance functions like cosine, Manhattan, and euclidean) to find the ideal cluster count at 12.

# 3 System Benchmarks

We profile our methodology under a variety of models (described in Section 2.4) and architectures (as illustrated in Figure 4). Specifically

| Model | Distributed | Partitioned | Containerized |
|---|---|---|---|
| NDVI Prediction | 6.8 hrs | 59.4 mins | 41.7 mins |
| NDMI Generation | 2.0 hrs | 1.6 hrs | - |
| SpAGAN | 17.35hrs | 1.3 hrs | - |

**Table 3:** Duration to complete model training comparing architectures. Our partitioned approach reduces training times by up to 13.9x. Containerizing the partitions reduces by an additional 29.8%.

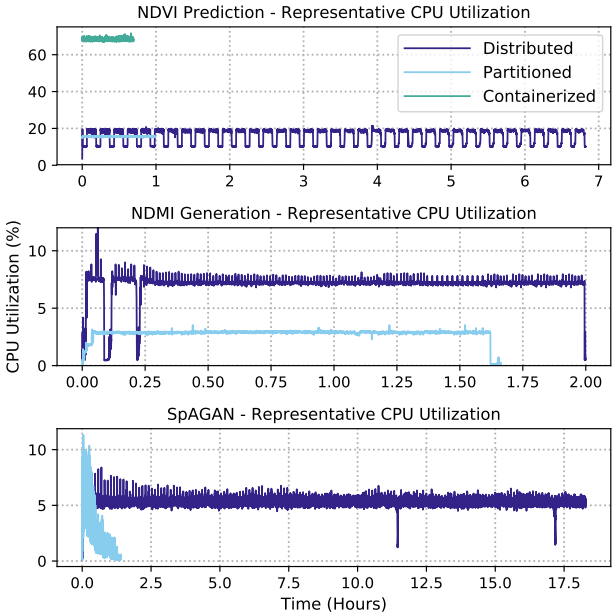



**Figure 10:** Average representative CPU utilization across cluster hosts during model training. Our partitioned architecture consistently reduces utilization compared with distribution. The containerized environment introduces additional CPU overhead.

we profile model training, reporting overall duration and resource utilization metrics, and model performance by:

(1) Training a single distributed model across the cluster using standard methods
(2) Training individual models for each geocode in the dataset, with models on each machine being trained sequentially
(3) Using a containerization system to schedule training of individual models across the cluster

## 3.1 Experimental Setup

Experiments were performed on a cluster of 50 machines (Xeon E5-2620, 64 GB Memory), each with a single Quadro P2200 GPU (1280 cores, 5GB of memory). Images captured from Sentinel-2 were used for each experiment, with varying spatiotemporal ranges.

## 3.2 Model Training Duration

We present the duration of model training in Table 3. Across models, our partitioned algorithm is faster than the traditional distributed approach. **Explicitly, there is a 6.9x, 1.25x, and 13.3x reduction in training durations for the NDVI prediction, NDMI generation, and SpAGAN models respectively**. The difference may be attributed to the large overhead required in distributing the training and syncing the model across all nodes in the cluster. We also note that **containerizing the partitioned training further reduces training times by 29.8% over a non-containerized partitioned workload and almost 10x over the distributed approach**. This is because Kubernetes leverages composite cluster resource utilization to more effectively schedule training tasks.

*3.2.1 Transfer Learning* In Figure 9 we observe that transfer learning has a significant effect on NDVI model training. We partitioned geohash regions into 12 clusters using the approach described in Section 2.5. Transfer learning from a random cluster (naive transfer

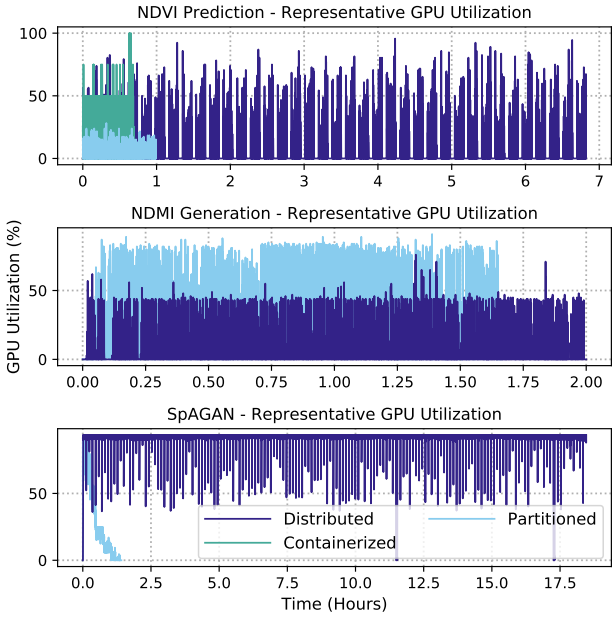



**Figure 11:** Average representative GPU utilization across cluster hosts during model training. Utilization varies among model definitions and training architectures. Broadly, the complexity of the model has significant impact on GPU use.

learning) converge twice as fast as without transfer learning, as well as reaching a lower loss. **Transfer learning from the same cluster resulted in approximately 4 times faster convergence and even lower loss.**

## 3.3 Training Resource Utilization

*3.3.1 CPU Utilization* We profile CPU utilization in Figure 10. The partitioned architecture consistently uses less CPU than distributed. Alternatively, containerizing the environment required considerably more CPU because it is reading datasets for multiple models simultaneously. For each model, we see a strong cyclic relationship on CPU utilization for the distributed architecture. This is due to periods of centralized model weight synchronization.

*3.3.2 GPU Utilization* Figure 11 compares GPU use while training. There is a large variance in utilization across model types. Generally this is correlated with the complexity of the model, where complexity increases from NDVI prediction to NDMI generation to SpAGAN. Within models, containerization makes the best use of the GPU by training multiple model simultaneously. Overall, the distributed architecture requires more use than our partitioned approach. This is because the distributed approach exhibits the same periodicity as CPU utilization. Again, this is a construct to model weight synchronization, but results in less efficient use.

*3.3.3 Network I/O* Total and average network I/O are presented in Table 5. Total cluster values may be extrapolated by the number of hosts. Maintaining lower network I/O is important because as clusters scale out, data transfers may become a performance bottleneck. Fortunately, our **partitioned training is significantly lower than distributed training for every model; consistently resulting in a 5 order of magnitude reduction in total network I/O**. Alternatively, containerized training requires each machine to communicate with the Kubernetes Controller, generating a modest amount of traffic. This decrease is established by the colocation of training tasks and data. Our partitioned approach does not require any data movements beyond initializing training tasks. In comparison, distributed training suffers from centralized management and synchronization of model weights.

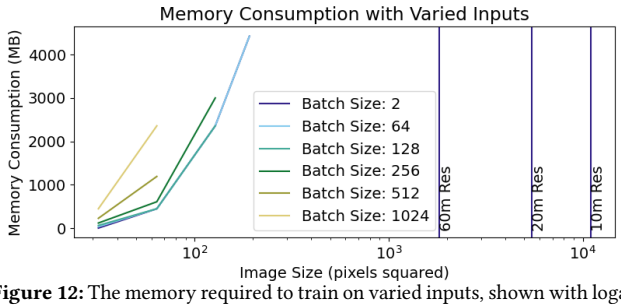



**Figure 12:** The memory required to train on varied inputs, shown with logarithmic scale. Undivided images are 100 times larger than the largest images that can fit in memory, making training on undivided images impossible.

*3.3.4 Memory Scalability* GPU memory is often a source of contention when training deep learning models. Within each training epoch training performance increases with the batch size (ie. number of concurrently processed images). However, batch sizes are restricted by the number of images that can be stored on the GPU simultaneously.

Figure 12 shows the limits of training models on undivided images, with lines showing the size of undivided Sentinel-2 images for their native resolutions. Multiple image and batch sizes are tested. Even with a batch size of two, which is generally inadvisable, the lowest resolution Sentinel-2 images are more than 10x too large to fit into memory during training and would require a GPU with 50-500GB of memory or distributing the model across many GPUs.

Alternatively, the raw images could be down-sampled by a factor of 10-100x. Doing so to this degree requires discarding a large amount of information. Our methodology reconciles the trade-off between a large number of expensive GPUs and reduced image quality by dividing the images into smaller pieces.

## 3.4 Model Performance

The accuracy of models is presented in Table 4. Figure 13 also presents sample result images of the NDMI prediction and SpAGAN models. These visualize image characteristics that are difficult to quantify, for example using our partitioned approach with the SpAGAN model is able to more accurately capture details of the surrounding landscape, like rivers and mountain ridges.

| Model | Loss | Distributed | Partitioned |
|---|---|---|---|
| NDVI Prediction | MSE | 0.121 | 0.083 |
| NDMI Generation | MSE | 0.00134 | 0.000253 |
| | MS-SSIM | 0.81 | 0.93 |
| SpAGAN | PSNR | 30.93409 | 31.559914 |
| | SSIM | 0.90471 | 0.92147 |

**Table 4:** Accuracy of models using distributed and partitioned architectures. Training models using our partitioned approach is statistically similar, or in some cases outperform, models trained with the distributed approach.

| Model | Average Network I/O | | | Total Network I/O | | |
|---|---|---|---|---|---|---|
| | Distributed | Partitioned | Containerized | Distributed | Partitioned | Containerized |
| NDVI Prediction | 460.1KB/s | 1.7KB/s | 5.6KB/s | 113.0GB | 6.1MB | 14.0MB |
| NDMI Generation | 102.4MB/s | 4KB/s | - | 737.0GB | 23.8MB | - |
| SpAGAN | 100.9MB/s | 87.5KB/s | - | 6.48TB | 430MB | - |

**Table 5:** Network I/O incurred during model training. Average network I/O is reduced by up to 4 orders of magnitude and total network I/O by over 30,000x when comparing distributed training to our partitioned approach. Containerized environments increase both metrics by 3x compared to our partitions.

**Our partitioned architecture performed better, on average, than the distributed approach on the NDVI prediction model**. Essentially, some regions had a very low loss and others had a very high loss. Comparatively, training with the distributed architecture results in a moderately higher loss for the single model, caused by its inability to specialize for certain regions. It should be noted that containerized training is functionally identical to partitioned training, it is merely scheduled differently. Therefore, accuracy measures are directly transferable.

NDMI prediction accuracy is reported using both the MSE and MS-SIM metrics. Using MSE, we can measure how much the predicted pixel value differs from the target pixel value, reflecting the water content of that spatial region. To compare the quality of the generated map, we have used multi-scale SSIM metrics that apply a low-pass filter followed by iterative down-sampling of the image, comparing the luminance, contrast, and structure at multiple scales. This helps in capturing the quality of the image at different resolutions. We see that the **our partitioned solution results in higher accuracy when compared to the distributed approach.**

We see the average PSNR and SSIM for our partitioned training with the SpAGAN model was slightly higher than the distributed training, demonstrating that **the partitioned models generated images more similar to the ground truth image than the distributed model**. Additionally, in Figure 13 we see the partitioned models more precisely capture landscape details, such as rivers and mountain ridges, than the distributed model. This is because our partitioned models are able to specialize on a particular region in contrast to the distributed model.

## 4 Related Work

Distributed deep learning practices aim to train a single model on multiple machines, where the increased parallelism reduces training times [7]. The two competing approaches are data parallelism [8] and model parallelism [9]. Data parallelism trains a copy of the entire model on each machine and synchronizes model weights after each epoch. This is the primary technique used in Tensorflow [10] and PyTorch [11] distributed packages. It introduces significant network overheads, which may transform network I/O into a bottleneck for large models. The Ring AllReduce algorithm [12] combats this by promising optimal bandwidth usage as long as network buffers are large enough. Alternatively, model parallelism distributed parts of the model across the cluster. The necessary training consensus among hosts mean this solution often suffers from concurrency-related training issues [13]. Consequently, it's only applied on very large models [14]. Our system is an alternative to data and model parallelism. Rather than partitioning the data or model, we partition the problem. This approach is similar to modular reinforcement learning [15–17] where a complicated task is decomposed into isolated sub-tasks.

Partitioning satellite imagery for efficient analysis has been proposed in a variety of domains. Recent efforts in Geographic Object-based Image Analysis (GEOBIA) [18, 19] accent the importance of resolutions in rectifying inclusion of multi-source imagery. Similarly, extraction of image features for farmland [20], building [21], and irrigation [22] show accuracy variances depending on spatial bound definitions. Georganos et. al. [23] process non-uniform spatial partitions to improve land-cover / land-use maps. Drawing from these previous works, our system facilitates partitioning satellite imagery; rectifying formats, spatial reference systems, and resolutions. Additionally, we provide verbose metadata filtering, such as image cloud coverage, which may be applied in any the aforementioned domains.

The normalized difference moisture index (NDMI) is used to quantify moisture, often absorbed by plants, and is essential in many agricultural remote-sensing applications. The issue is that hyperspectral imagery typically observed NDMI as lower resolutions than other correlated bands. Yang et. al. [24] leverages the green and NIR bands to overcome low resolution (20m) SWIR bands and generate pan-sharped water maps at 10m resolution. Similarly, Du et. al. [25] applied the ATWT method to combine high-resolution bands with interpolated SWIR bands using wavelet transformations. Rokni et. al. [26] applies PCA over a multi-temporal NDMI index to achieve comparable results. Our approach achieves similar results by employing neural network that learns the relationships between correlated bands (ex. RGB and NIR/SWIR) over spatial regions.

A number of modern methods of cloud removal are based on conditional GANs. These include multispectral conditional GANs (MCGAN) [27], Pix2Pix GANs [5], and spatiotemporal GANs (STGAN) [28]. Because of the success of these multi-band models, other geospatial data are now being incorporated into cloud removal GANs. One example of this is the Simulation-Fusion GAN [29] which uses synthetic-aperture radar (SAR), combined with satellite

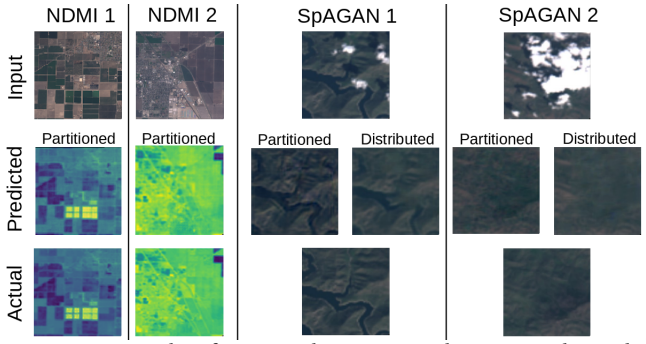



**Figure 13:** Examples of NDMI and SpAGAN predictions visualizing the accuracy of our partition-based training approach.

imagery, to produce cloud-free images. Although these models perform well, in practice they require large amounts of data, long training times, and therefore must be distributed. Our system focuses on combating these issues to improve the utility of the aforementioned techniques.

## 5 Conclusions and Future Work

In this study, we presented our methodology to build effective deep learning models over satellite image collections. Our empirical benchmarks demonstrate the suitability of our methodology in ensuring significantly faster completion times without compromising on accuracy, while using resources frugally.

**RQ-1:** Partitioning hyperspectral satellite imagery allow us to apportion not just the storage load, but subsequent processing workloads as well. Leveraging geocodes allows us to deterministically and hierarchically partition the imagery. Collocating all data from a given extent on the same machine precludes data movements during model training.

**RQ-2:** Reconciling spatial reference systems, encoding formats, and image types allows modelers to reconcile heterogeneity across satellite data collections. Targeted retrievals of data based on query predicates specified over spatial, temporal, and other attributes is faster because of search space reductions and data structures used for query evaluations. Creation of synthetic imagery, identification of occlusions due to clouds and available data allows us to construct effective training datasets. Rather than build an all-encompassing model, building an ensemble of models, each for a particular spatial extent, allows each constituent model to tune itself to feature space characteristics at that scope. Given that this has the added benefit of reduced resource utilizations and overall completion times we expect greater experimentation as well.

**RQ-3:** Our methodology is particularly amenable to horizontal scaling. This is made possible by a mix of partitioning imagery, leveraging geocodes to do so hierarchically and deterministically, and multi-token DHTs for decentralized and dynamic load balancing. Reducing data movements during model training is key to ensuring that network I/O is not amplified during training. By ensuring data collocation during model construction we substantially reduce data movements. Furthermore, our methodology of training models for spatial extents ensures that there is no network I/O due to synchronization barriers over large collections of machines. Cumulatively, this allows our methodology to scale with increases in DVs and the number of available machines. As our benchmarks demonstrate, not only is our methodology agnostic of the deep learning libraries, we are able to leverage these libraries to outperform their own native distributed implementations. Finally, our methodology demonstrates excellent performance in containerized environments as well.

As part of future work, we will be exploring issues relating to object tracking across spatiotemporal scopes. The objects here also represent phenomena where features derived from hyperspectral imagery take on values within a specified range.

## Acknowledgments

This research was supported by the National Science Foundation [OAC-1931363, ACI-1553685], the National Institute of Food & Agriculture [COL0-FACT-2019], and a Cochran Family Professorship.

## References

[1] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014.
[2] Kelsey Hightower, Brendan Burns, and Joe Beda. *Kubernetes: Up and Running Dive into the Future of Infrastructure.* O'Reilly Media, Inc., 1st edition, 2017.
[3] DM Stoms and WW Hargrove. Potential ndvi as a baseline for monitoring ecosystem functioning. *International Journal of Remote Sensing*, 21(2):401–407.
[4] Juan C Jiménez-Muñoz, José A Sobrino, Alan Gillespie, Donald Sabol, and William T Gustafson. Improved land surface emissivities over agricultural areas using aster ndvi. *Remote Sensing of Environment*, 103(4):474–487, 2006.
[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, 2016.
[6] Jesse Allen, Kevin Ward, Adam Voiland, et al. Cloudy earth.
[7] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.
[8] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2017.
[9] Jeffrey Dean, Greg Corrado, Rajat Monga, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.
[10] Martín Abadi, Paul Barham, Jianmin Chen, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.
[11] Adam Paszke, Sam Gross, Soumith Chintala, et al. Automatic differentiation in pytorch. 2017.
[12] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.
[13] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond data and model parallelism for deep neural networks. *arXiv preprint arXiv:1807.05358*, 2018.
[14] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2019.
[15] D. Jacob, D. Polani, and C. L. Nehaniv. Legs that can walk: embodiment-based modular reinforcement learning applied. In *2005 International Symposium on Computational Intelligence in Robotics and Automation*, pages 365–372, 2005.
[16] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175, 2017.
[17] Sooraj Bhat, Charles L Isbell, and Michael Mateas. On the difficulty of modular reinforcement learning for real-world partial programming. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 318. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
[18] Stefan Lang, Geoffrey J Hay, et al. Geobia achievements and spatial opportunities in the era of big earth observation data. *ISPRS International Journal of Geo-Information*, 8(11):474, 2019.
[19] Nicholus Mboga, Stefanos Georganos, et al. Fully convolutional networks and geographic object-based image analysis for the classification of vhr imagery. *Remote Sensing*, 11(5):597, 2019.
[20] Lu Xu, Dongping Ming, Wen Zhou, Hanqing Bao, Yangyang Chen, and Xiao Ling. Farmland extraction from high spatial resolution remote sensing images based on stratified scale pre-estimation. *Remote Sensing*, 11(2):108, 2019.
[21] Gunho Sohn and Ian Dowman. Data fusion of high-resolution satellite imagery and lidar data for automatic building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(1):43–63, 2007.
[22] Hassan Bazzi, Nicolas Baghdadi, Dino Ienco, et al. Mapping irrigated areas using sentinel-1 time series in catalonia, spain. *Remote Sensing*, 11(15):1836, 2019.
[23] Stefanos Georganos, Tais Grippa, Moritz Lennert, et al. Scale matters: Spatially partitioned unsupervised segmentation parameter optimization for large and heterogeneous satellite images. *Remote Sensing*, 10(9):1440, 2018.
[24] Xiucheng Yang, Shanshan Zhao, Xuebin Qin, Na Zhao, and Ligang Liang. Mapping of urban surface water bodies from sentinel-2 msi imagery at 10 m resolution via ndwi-based image sharpening. *Remote Sensing*, 9(6):596, 2017.
[25] Yun Du, Yihang Zhang, Feng Ling, Qunming Wang, Wenbo Li, and Xiaodong Li. Water bodies' mapping from sentinel-2 imagery with modified normalized difference water index at 10-m spatial resolution produced by sharpening the swir band. *Remote Sensing*, 8(4):354, 2016.
[26] Komeil Rokni, Anuar Ahmad, Ali Selamat, and Sharifeh Hazini. Water feature extraction and change detection using multitemporal landsat imagery. *Remote sensing*, 6(5):4173–4189, 2014.
[27] Kenji Enomoto, Ken Sakurada, Weimin Wang, et al. Filmy cloud removal on satellite imagery with multispectral conditional generative adversarial nets. *CoRR*.
[28] Vishnu Sarukkai, Anirudh Jain, Burak Uzkent, and Stefano Ermon. Cloud removal in satellite images using spatiotemporal generative networks. 2019.
[29] Jianhao Gao, Qiangqiang Yuan, Jie Li, Hai Zhang, and Xin Su. Cloud removal with fusion of high resolution optical and sar images using generative adversarial networks. *Remote Sensing*, 12:191, 01 2020.