Apache Spark Accelerated Deep Learning Inference for Large Scale Satellite Image Analytics

Dalton Lunga, Jonathan Gerrand, Lexie Yang, Christopher Layton, and Robert Stewart

***Abstract*—The shear volumes of data generated from earth observation and remote sensing technologies continue to make major impact; leaping key geospatial applications into the dual data and compute-intensive era. As a consequence, this rapid advancement poses new computational and data processing challenges. We implement a novel remote sensing data flow (RESFlow) for advancing machine learning to compute with massive amounts of remotely sensed imagery. The core contribution is partitioning massive amounts of data into homogeneous distributions for fitting simple models. RESFlow takes advantage of Apache Spark and the availability of modern computing hardware to harness the acceleration of deep learning inference on expansive remote sensing imagery. The framework incorporates a strategy to optimize resource utilization across multiple executors assigned to a single worker. We showcase its deployment in both computationally and data-intensive workloads for pixel-level labeling tasks. The pipeline invokes deep learning inference at three stages; during deep feature extraction, deep metric mapping, and deep semantic segmentation. The tasks impose compute-intensive and GPU resource sharing challenges motivating for a parallelized pipeline for all execution steps. To address the problem of hardware resource contention, our containerized workflow further incorporates a novel GPU checkout routine and the ticketing system across multiple workers. The workflow is demonstrated with NVIDIA DGX accelerated platforms and offers appreciable compute speed-ups for deep learning inference on pixel labeling workloads; processing 21 028 TB of imagery data and delivering output maps at area rate of 5.245 sq.km/s, amounting to 453 168 sq.km/day—reducing a 28 day workload to 21 h.**

***Index Terms*—Big data applications, high performance computing, image classification, inference mechanisms, machine learning, supervised learning.**

I. Introduction

EARTH observation and remote-sensing are both fields that have undergone a renaissance recently, making major impacts in key geospatial applications including land cover mapping, infrastructure mapping, damage assessment, and population distribution studies [1]–[4]. Multiple factors are contributing to this change, including significant improvements and rapid deployment of satellite technologies that are enabling the acquisition of vast volumes of high-resolution imagery at high velocity rates. As such, remote sensing applications have leaped into a data and compute-intensive era presenting challenges and opportunities for new advanced machine learning and computer vision workflows. Examples of such applications include supporting accurate population distribution estimates, possibilities to study sustainability outcomes at scale [5], and identifying urban environments over large contexts using abundant satellite imagery and breakthroughs in deep learning based image classification [6].

To achieve greater impact with machine learning on data and compute intense workloads, new approaches are required for efficient utilization of high-performance computing resources and to produce efficacious results for end-users. These approaches need to consider both the computational aspects of their target applications, as well as the challenges inherent with analyzing remote sensing data in a generalizable manner. Specifically, we consider the following three problem areas, which need to be addressed in order to advance the current state of the art, namely:

1) data-intensive challenges;
2) labor-intensive challenges;
3) compute-intensive challenges.

For clarity, we briefly discuss each of these challenges.

On the data-intensive challenge: The shear volumes of remote sensing imagery are increasingly becoming heterogeneous and challenging the efficacy of current machine learning workflows [7]. With imagery data acquired as signals from varying system configurations and environmental conditions, efforts to analyze such diversity at scale are immediately thwarted by a lack of workflows whose results are adequately generalizable both spatially and temporally within the data.

On the labor-intensive challenge: Current techniques from machine learning, especially deep learning, continue to demonstrate the near-human performance. However, such methods are heavily dependent on large annotated datasets. While there are growing efforts to build open research benchmark data to address this issue [8], when relevant amounts of high-quality labeled data are not available, open source data driven models tend to achieve poor generalization capability. The endeavor to obtain high-quality training data can then be tedious, especially for pixel labeling, and is characterized by multiple attributes that include; availability of domain experts, stratification of data into diverse representative samples, mitigation of human sampling bias, and accurate labeling of data samples.

On the compute-intensive challenge: Pixel labeling algorithms pose a compute-intensive workload even under normal

Manuscript received June 20, 2019; revised November 9, 2019; accepted December 4, 2019. Date of publication January 2, 2020; date of current version February 12, 2020. This study was supported by the National Security Sciences Directorate, Oak Ridge National Laboratory. *(Corresponding author: Dalton Lunga.)*

The authors are with the National Security Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN 37830 USA (e-mail: lungadd@ornl.gov; gerrand.jonathan@gmail.com; yangh@ornl.gov; laytoncc@ornl.gov; stewartrn@ornl.gov).

Digital Object Identifier 10.1109/JSTARS.2019.2959707

usage within the natural image domain. When considered in the context of petabytes of remote sensing data that needs to be processed, however, with single image files often reaching tens of gigabytes in size, the efficient use of high-performance computing resources needs to be considered. Here, solutions existing within a distributed environment, spanning several nodes either for feature extractor training or model inferencing [7], need to be considered.

This article seeks to address the abovementioned challenges from a generic perspective amenable for use in other large scale object mapping applications using remote sensing imagery. As such, we propose a novel and efficient single pipeline, herein referred to as RESFlow, with multiple deep neural networks for multipass distributed image data analysis performed in an embarrassingly parallel fashion.

RESFlow seeks to stratify imagery into homogeneous distributions from which levels of diversity are contained to inform the reuse of models for inference tasks on imagery partitions with similar characteristics but originating from mixed geographies. By seeking automated mapping into these homogeneous partitions, our goal is to create data buckets from which to sample representative images with similar characteristics, mitigating the need to stratify large and diverse training data. Furthermore, to address computational aspect of the problem, we formulate RESflow as constituting a set of subtasks with interdependence but which are each executable in an embarrassingly parallel fashion across bucket partitions. As such, no communication of compute results takes place between partitions. Each partition is computed upon independently with little communication only encountered on the last stage to reconstruct inference results for a given image scene. However, within partitions, subtasks have dependence on each other, which imposes an order of precedence on their execution, creating a task scheduling problem and resource contention that we handle via a novel GPU ticketing system.

The technical contributions of this article are as follows.

1) We present an unprecedented homogeneous partitioning of massive amounts of imagery data based on its semantic and spectral characteristics. We leverage this partitioned space to enable efficient indexing 10 s of models and 1000 s of image patches for distributed pixel labeling.
2) We take advantage of Apache Spark to provide, for a single large image scene, fast parallel inference functionality wherein an area pixel labeling rate of 5.245 sq.km/s, amounting to 453 168 sq.km/day is achieved—reducing a 28 day workload to 21 h.
3) We present a containerized workflow for Apache Spark operations coordinated with GPUs for deep learning inference best practices, e.g., efficient GPU usage and ticketing across multiple workers, for large deep learning workloads deployed on GPU clusters.

Although presented for a pixel labeling task on satellite imagery, the workflow can easily be deployed to domains that exhibit the same problematic data characteristics as described previously. Examples include biomedical and climate image based applications.

The remainder of this article is arranged as follows. Section II reviews the components of several satellite image analytics workflows, including deep neural networks for semantic segmentation and distributed computing frameworks. Using insights gained from this review, Section III discusses the proposed high-performance computing-based remote sensing imagery analytic workflow. To illustrate the workflow, benchmarking compute efficiency statistics on varying workloads are presented in Section III-J. Section IV large scale pixel-level segmentation results for building extraction. Finally, Section V concludes this article.

II. Satellite Image Analytics With Deep Neural Networks and Distributed Computing at Scale

Given the prevalent nature of high-resolution remote sensing instruments, it is now conceivable to pursue computer vision methods for large scale object segmentation. Very-high-resolution remote sensing imagery, which now supports ground spatial resolutions of less than 50 cm, is enabling new capability to exploit subtle and yet expressive spatial features for fitting highly complex objective functions for structured predictions with computer vision and machine learning methods. Deep convolutional neural networks have become the dominant machine learning technique for visual recognition, achieving state-of-the-art results on a number of problems that seek dense semantic labeling of image pixels. Early attempts on this problem include work in [9] where an atrous method to expand the support of filters and reduce the down-sampling for input feature maps to achieve dense labeling was used. In [10], an efficient and precise biomedical image segmentation convolutional neural network (U-net) was proposed. Improving on the architectural design to reconstruct the original input resolution, Badrinarayanan *et al.* [11] proposed a semantic pixel-wise segmentation method using a fully convolutional neural network (Seg-Net), which uses decoder-deconvolutional layers to map the low-resolution encoder feature maps to the full input resolution feature maps. The use of deep convolutional neural networks extends to other applications including big data mining for search and retrieval tasks. Designed to seek expressive spatial and visual content representational features, the deep hashing framework created by Li *et al.* demonstrated capability for large-scale image retrieval in [12]. In another image retrieval task, pretrained networks were used in order to extract intermediate image representation as input for metric and hash-code learning [13].

In general, to perform complex tasks at the level of humans, deep learning methods heavily depend upon the availability of enormous amounts of high-quality annotated data. Despite the fact that remote sensing instruments are acquiring data in substantial volumes and the robust computing power needed to efficiently process it is available; such massive datasets are not simple to annotate. The process of gathering labeled training data is mired by inconsistencies, poor selection of representative samples, and the annotation is often prohibitively expensive. It is labor-intensive requiring a huge number of worker hours, making it challenging to train a single high-performing deep network model for use on wide area geographical coverage.

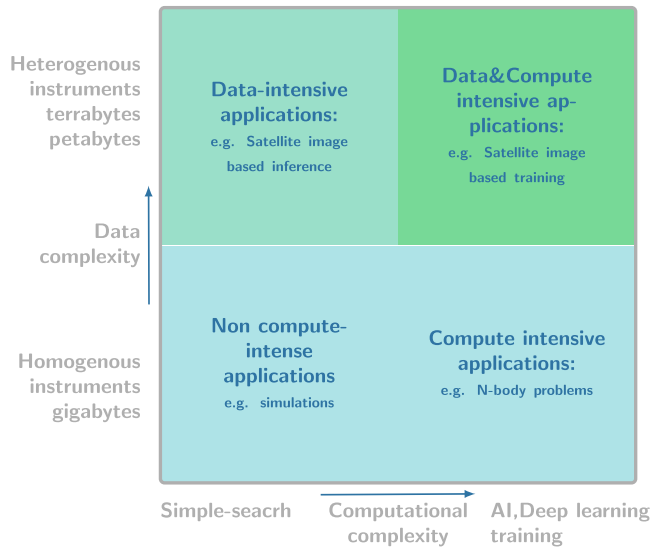


Fig. 1. Data, compute, and labor intensity paradigm in remote sensing applications. (Note: labor is illustrated with color intensity *limegreen* denotes intensity in labor demand.)

We, therefore, feel it is appropriate to seek automated workflows, which support representative training data selections, e.g., avoid underrepresentation, enable localized model for capturing homogeneous data distributions, or fit models on diverse yet equally sampled image characteristics.

Moreover, with a growing demand for geospatial applications to deliver imagery products on data that scales over 10 s of Terrabytes (TB) of high-resolution outputs per given geographic region, current applications are gradually becoming immense in terms of both data and compute requirements. To be specific, typical workloads for semantic labeling often entail processing imagery acquired across an average country of land-area size 783 562 sq.km. The accompanying imagery coverage would span $3 \times 783\,562$ sq.km to account for scene overlapping and lack of cloud free data. Compare this volume of data against the largest known computer vision dataset ImageNet [14]. ImageNet has a total of 14 197 122 images, each at 224×224 pixels thus 50 176 pixels per image, totaling 712 354 793 472 pixels. When sampled at ground-sampling distance of 50 cm, a mosaicking of all imageNet totals 356 177 sq.km, slightly less the size of Montana, USA. In contrast, for an average country size, at 50 cm ground sampling distance each RGB image scene spans about $40\,000 \times 35\,000$ pixels and carries $\approx$13 GB of data for a total of 3000 scenes (covering equivalent of $3 \times 783\,562$ sq.km ($7 \times$ ImageNet) land-area and totalling $\approx$39 TB of data). Using current serial processing pipelines a single image scene takes 35 min to process a pixel-labeling task on a single computing node with one 16 GB GPU card. Considering the demands to process multiple country scale products, it is imperative that object segmentation and semantic labeling tasks are deployed through parallel and distributed inferencing pipelines to reduce such computational intensity. With this motivation, we identify advanced remote sensing dataflows (including *RESFlow*) to be located in the top two quadrants of Fig. 1 and continue to develop core computational modules that can match the demands of such applications.

Over the past decade, Hadoop has emerged as an early experimental testbed for several big data applications due to its excellent large-scale data-handling capability, high fault tolerance, reliability, and low cost of operation [15]. Hadoop provides distributed data storage and analysis solutions, which previously have been exploited for implementing large scale mean-shift-based image segmentation algorithms [16]. In [17], an optimization effort on the Hadoop file storage system was studied to elicit better performance for large scale computing with image data. The authors of [18] studied a Hadoop and MapReduce [19] based implementation of the parallel K-means algorithm to reduce the computational time taken for executing parallel data clustering on a large number of satellite images. Pursuing content mining on digital images, the authors in [20] introduced an approach for large-scale scene retrieval on massive image databases.

While MapReduce enables large-scale distributed computing for imagery when used in conjunction with Hadoop, a limitation is seen in its heavy usage of disk input–output (I/O) operations and network resources to store intermediate steps during processing. Deterred by this computational cost, Huang *et al.* [21] studied Apache Spark [22] to take advantage of its resilient distributed datasets (RDDs) [23]. Spark has been shown to accelerate several other remote sensing imagery workloads. For example, in applications where transregional remote sensing images are key, the frequent data I/O requirements for mosaicking were shown to benefit from a parallel algorithm implemented with Spark [22], [24]. The work of Sun *et al.* [25] also demonstrates this performance increase, wherein the authors implement an iterative singular value decomposition algorithm to process massive amounts of remote sensing data. In concurrence, high-performance computing environments are enabling targeted computing with extremely large earth observation data and the sharing of data in parallel across hundreds of nodes [26], [27]. Taking advantage of the high processing power, large memory capacity, and Infiniband (IB) enabled interconnects between nodes in Summit, Kurth *et al.* [28] proposed an exascale ready workflow and software stack for extracting signals for extreme weather patterns using variants of deep neural networks. The work scaled up to 27 360 V100 GPUs and sustained throughput of 325.8 PF/s and a parallel efficiency of 90.7% in single precision. We see these impressive results as an indication Spark's ability to enable the processing of remote sensing data at scale and demonstrate this as part of the RESFlow framework presented within the remainder of this article.

III. Proposed RESFlow Framework

The RESFlow architecture seeks to present itself as an intelligent big data engine where end-to-end inference tasks are efficiently executed while exploiting the geometry of the data and being agnostic to sensor variations as well as geographic constraints. To this end, it is formulated to contain several integrated computational stages and algorithms; providing a common data pipeline, which is shared across multiple inference tasks and geospatial applications. At the core of RESFlow is the concept of data distribution partitioning, which is performed via efficient geometric based clustering and metric learning.

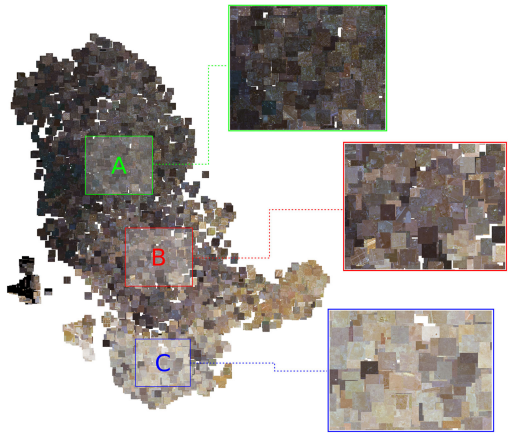


Fig. 2. Visual representation of a continuum embedding space for an image scene. Groupings A, B, and C illustrate sets of image patches that inform homogeneous data partitions in the workflow.

Both techniques play key roles in the overall mapping of data to a partitioned image space indexing—a strategy often lacking in traditional learning workflows. Here, we believe that data partitioning is key to mitigating bias in training data collection and labeling (even though not in scope of the current study). Fig. 2 illustrates the partitioning of an embedding representation for image patches extracted from a single large satellite image scene. From this result, the concept of data partitioning can be observed as the grouping and extraction of homogeneous image spaces for further exploitation during inference of the large image scene.

As noted previously, computer vision and machine learning algorithms are proving superior in providing automated means to describe the distinctive nature of objects in remotely sensed image data [1], [5], [6], [29]. However, the deployment of such algorithms remains a significant challenge when considered on large geographic areas covered by hundreds of thousands of images [29].

As is tradition with data/compute intensive applications, success depends upon scarce and expensive hardware resources. It is, therefore, not surprising that the use of hybrid CPU/GPU technology stacks is emerging as the means to address such deployment challenges. For example, deep learning functionality for analysis can be developed as user defined functions (UDFs) and used within Apache Spark clusters for inference deployment on GPU and CPU servers in a resourceful manner. Motivated by such potential, we combine salient features from the deep learning frameworks (e.g., TensorFlow and PyTorch) and big-data capabilities from Apache Spark, to implement accelerated and parallelized inference modules for use on both CPU and GPU servers.

The central means to achieving such capabilities is the idea that both remotely sensed image data, and deep learning models, can be mapped to and paired within local regions in which the extreme diversity induced by sensor characteristics and scene content is constrained. As depicted in Fig. 3, this procedure is initially facilitated using a learned functional mapping, which partitions high-dimensional data embeddings into several buckets of similar semantic and spectral content and stored within an image gallery. The bucket partitions then provide a basis to train and update associated indexable models, stored within a model gallery, which can be tailored for specific inference tasks as defined by the application domain.

We briefly describe the *RESFlow* architecture several of its modular components in the following sections.

A. Clustering and Embedding

The first step to enable remote sensing imagery partitioning is utilizing the clustering and embedding module (CEM) in *RESFlow*. The responsibilities of the CEM are two-fold. First, as the initial step within the coalescing of imagery to appropriate partitions of the image gallery, the module maps each input image as a datapoint using a learned feature extractor to an intermediate representation in which other datapoints characterized by similar acquisition conditions, spectral and semantic content share a close proximity. This intermediate mapping, or network embedding, is important as it provides a basis for an appropriate metric space to be learned in a data-driven manner. Second, during *RESFlow's* initialization, the CEM is used to assign labels via clustering as a means to assist learning of the metric space projection function. Here, multiple clustering algorithms can be considered for usage and the framework is importantly unconstrained by any specific approach. Within our experimentation, we initially evaluate several popular clustering algorithms including centroid and hierarchical-based approaches using a euclidean metric space. Based on its favourable performance in minimizing intercluser variance, we select agglomerative clustering with a Ward linkage criterion for use within the framework.

B. Image-Bucket Assignment

After clustering the partitioned images, we seek to construct buckets that uniquely represent those clustered images. The unique binary representation generated for each image plays a significant role in the following two ways: 1) They provide compact binary bitstrings that preserve semantically similar content for the partitioned image chips, and 2) The binary bitstrings provide an efficient image-model indexing mechanism during large scale inference. This dual benefit is achieved by learning a hashing metric space whose properties include the following:

1) generating a unique hash-map associated with each distinct image chip;
2) providing a compact representation that is smaller than the original input dimensions;
3) a metric space from which the distance property for binary bitstrings can be used to relate image scenes whose geographies and image characteristics are also similar.

The resulting binary bitstrings represent a desirable format with which to both efficiently index pretrained models and the respective buckets; proving an intuitive gathering space for localized training data.

The assignment of images to buckets is achieved by first computing the centroid binary bitstring for each bucket. Each bucket centroid offers a unique binary bitstring that is reused to identify each bucket model. Following this assignment, the centroid gets reused within a hamming space to identify which

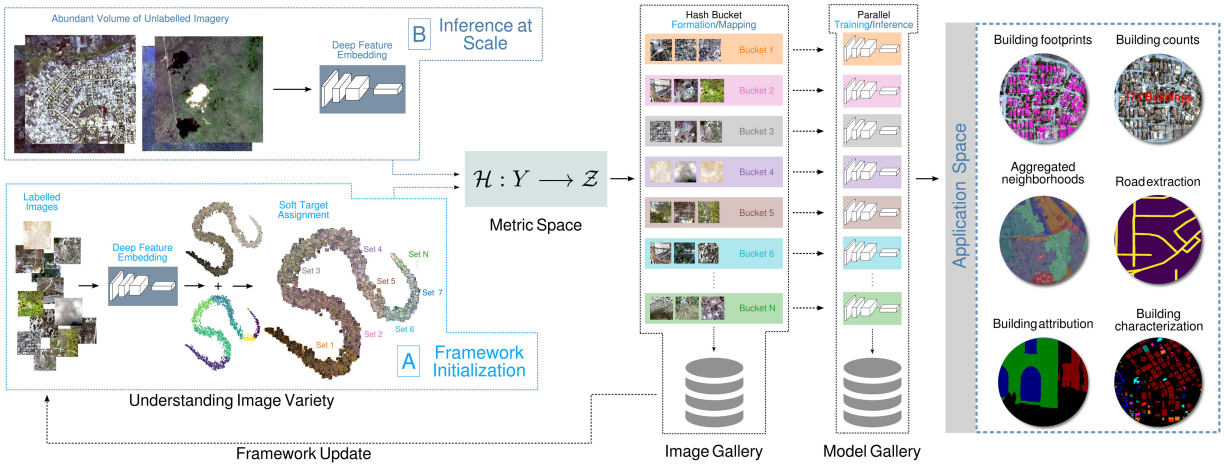


Fig. 3. Overview of the RESFlow framework.

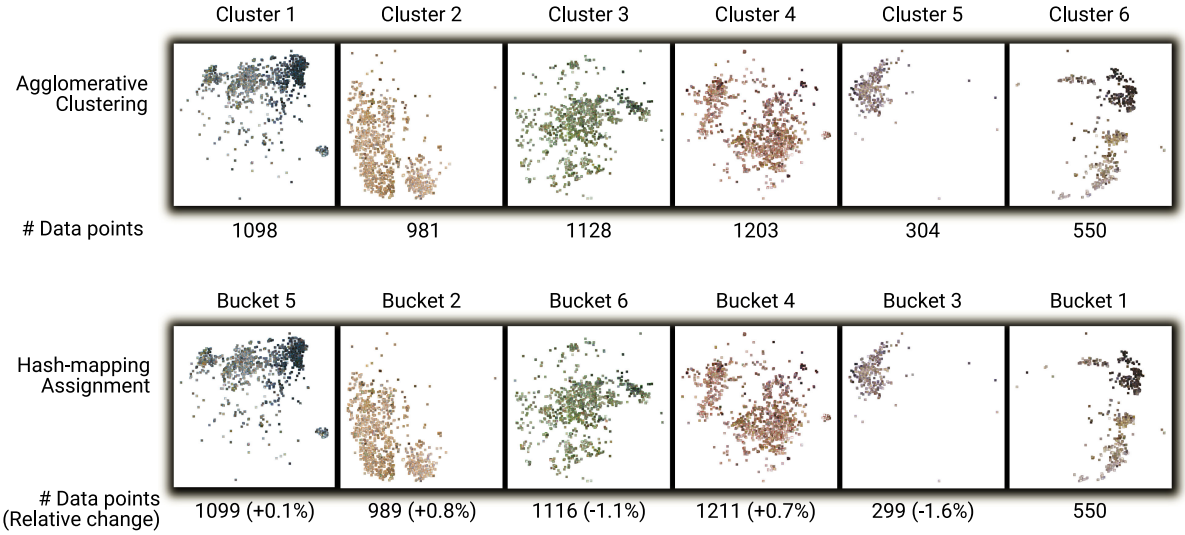


Fig. 4. Comparison between buckets initialized via agglomerative clustering and hash-mapping. Each cluster on the top row is color-coded to denote image patches belonging to one cluster. The bottom row shows a reconstruction of clusters mapped into hash-buckets. Hash-buckets are reconstructed with a validation mAP score of 98.3% with fewer image patches on the bottom row noted to have been placed in different buckets than their cluster of origin.

bucket each image chip be assigned to. Fig. 4 illustrates the main concept upon which *RESFlow* operates with hash-mapped buckets created via clustering. Here initial clustering provides the key soft-labels needed to learn the semantic structure of the large satellite imagery archive. As shown, the reconstruction of the similar content structure, without emphasizing cluster-bucket correspondence, is carried out by the hash-mapping function. Furthermore, the distinct buckets become completely indexable within the image gallery shown in Fig. 3.

To illustrate the hash-mapping module, we observe the ability of a convolutional network based model to reconstruct the CEM generated embedding space while varying the number of initial clusters. Using agglomerative clustering, we select clusters to generate the soft-labels based upon a metric of the smallest variance per cluster obtained over several different cluster counts. The hash-mapping network is evaluated using the mean Average Precision (mAP) metric, which assesses the average value of the maximum precision for different recall levels while reconstructing the structure of initial clusters. Using a color-coding scheme, Fig. 4 shows the relative changes between the agglomerative-based clusters and the convolutional neural network based hash-mapped buckets.

C. Image Gallery

Following their creation via the metric space projection function, the binary bitstrings generated from input imagery are partitioned by similarity relative to the distinct buckets shown in Fig. 4. These partitions form a powerful abstraction, independently characterizing homogeneous image acquisition characteristics and spectral content, from which both training and inference data can be sourced for a related model gallery network. Further associated with each binary bitstring entry within this gallery is additional meta-data detailing attributes

such as geo-information, acquisition conditions, storage location, and image subcoordinates. Combined with the efficient binary representation of each datapoint for rapid image search capability, the inclusion of these attributes enable rich insights to be gained through exploratory inspection of the gallery content. These insights could not only benefit performance evaluation of models but also help explain complex data characteristics and their bias.

D. Model Gallery

Once populated, the image gallery presents a collection of homogeneous partitions. As previously motivated, this collection, with constrained diversity, is a desirable property when sampling for both training and inference data with a given deep learning algorithm. The model gallery will provide a direct mapping between a bucket partition within the image gallery and a paired and trained network model that is either fine-tuned during training, or applied during inference. Following this procedure can prove as a highly efficient alternative to standard model training and inference practices, as the localized constraints placed on the new data which each gallery model sees mitigates the need for continual retraining.

E. Accelerated Inference

To this point, each of the discussed modules are seen to play a role in overcoming the extreme variance characteristic within imagery with the core concept of partitioning remote sensing imagery. However, the issue of analyzing this data at sufficient scale to meet the demands of global-size applications remains a prohibitive concern. In addressing this problem, it is further observed that each of the modules within *RESFlow* utilize elements of deep learning to perform different tasks, presenting a computationally heavy workload that requires the same GPU hardware resources to be reused in a single inference run. However, *RESFlow*'s building blocks and their functionality are amenable to massive parallelization across the partitioned image space. In recognition of this condition, we exploit Apache Spark as a fabric for distributing and coordinating the framework's computations at scale. Learning from libraries such as TensorflowOnSpark [30] and Tensorframes [31], we leverage Spark's big data capability to ingest large quantities of input data and process these using deep learning frameworks complimentary to Spark in a fault-tolerant and highly parallel manner. Fig. 5 shows *RESFlow* tile inference and reconstruction illustration. The tile partitioning strategy injects a key property that allows for spatially noncontiguous image tiles to be processed by a single bucket model—enabling consistent inference over wide geographic conditions.

F. Application Space

By design, the modules presented thus far within the *RESFlow* framework have been agnostic to any specific application or use case within the realm of remote-sensing imagery. In this manner, one could think of the model gallery partitioned to fulfill multiple tasks such as object detection, neighborhood and settlement mapping, or temporal change detection, which are only a subset of potential applications. Based on this premise, within the application space multiple copies of the model gallery are formed, each containing trained models, which are uniformly purposed to perform a given task.

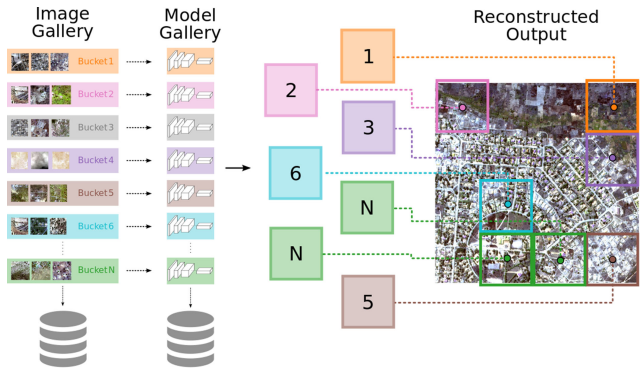


Fig. 5. *RESFlow* tile inference and reconstruction illustration. Colors denote inference deployment of different gallery models as assigned by the image-bucket module. The tile partitioning strategy injects a key property that allows for spatially noncontiguous image tiles to be processed by a single bucket model—enabling consistent inference over wide geographic conditions.

G. Image Analytics Via Parallel Computing

1) Satellite Imagery RDDs: Spark currently does not support extended image file types such as. tiff or. dicom to be serialized into byte arrays encapsulated within its RDD objects.[1] As a consequence of this limitation, a design choice was required between either converting the collected sensor-based imagery used within *RESFlow* into supported formats (such as 8 b jpg), or to instead use RDD objects to store path-based references to the location of the imagery stored within network storage. The former approach would result in a loss of data precision (32–8 b for each scene), while the latter would force the image data to be read twice from disk during workflow execution (once for hashing and embedding, and a second time for per-bucket inference). With precedence being placed on precision to enable higher levels of accuracy during training, we choose the latter option and instantiate an RDD to contain the paths of the scene data to be analyzed in *RESFlow*.

2) Spark-Based UDFs: We implement several UDFs within Spark in order to realize *RESFlow's* operation. Represented in Fig. 6, these functions encapsulate tasks such as getting the extent of a given scene tile, or performing inference across a partition-based bucket. We utilise external libraries such as Tensorflow [32], Pytorch [33], and GDAL [34] to assist in performing these operations, which act upon one or more rows records within a given RDD partition.

3) GPU Allocation Heuristic: An important limitation in using Spark for *RESFlow*'s implementation is its lack of support for GPU-based resources. Here, first-class status is given to cluster-based resources such as CPU cores and RAM; allowing a fixed quantity of these resources to be allocated to instantiate a spark-executor. On the other hand, Spark contains no

[1][Online]. Available: https://issues.apache.org/jira/browse/SPARK-21866

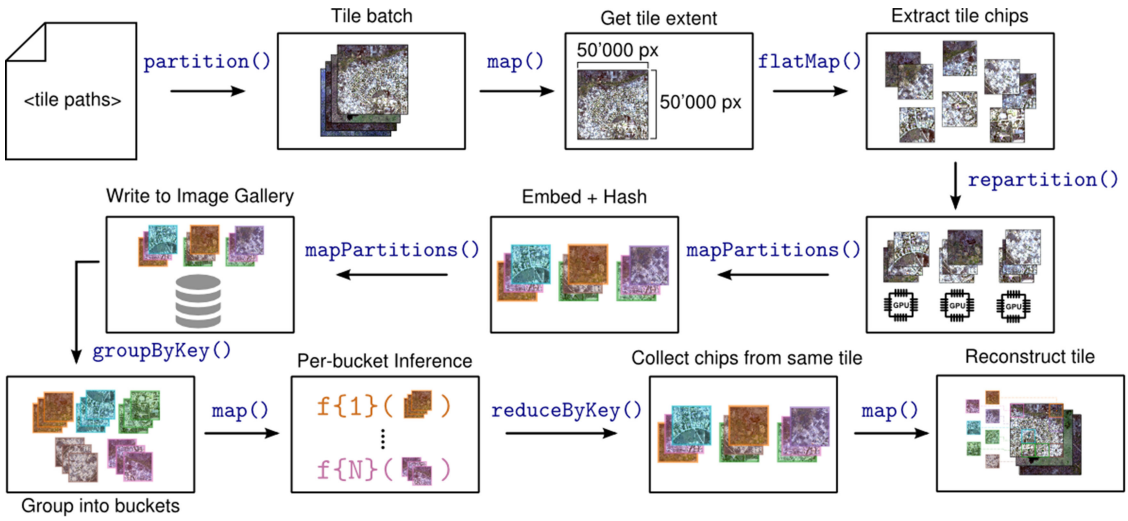


Fig. 6. *RESFlow* parallel inferencing flow: a high-level representation of the Spark-based transformations and actions implementation.

functionality to enable GPUs to be exclusively associated with a given executor in the same manner. When considered in the light of how Spark runs tasks that belong to the same processing stage, maintaining independence, and concurrency, this lack of GPU resource allocation can become problematic. For example, during the embedding and hashing step of the workflow each executor is required to process the data within its associated partitions via GPU computation. With no robust mechanism to reserve a GPU for a given executor, this step may result in several executors utilizing the same GPU—causing the stage to fail as the limited amount of GPU RAM is quickly exhausted by the concurrent executor tasks.

As a workaround to this problem, we implemented a simplistic `GPU checkout` routine that allows for the soft-assignment of GPU resources to take place between the executors on a given worker node. We accomplish this by creating a ticketing folder within the executors' shared work-space, into which we place one or more "tickets" per physical GPU within the node. Using this ticket-system metaphor, when an executor requires the use of a GPU, it scans the ticketing folder for available instances—removing a ticket if one is available, or returning a ticket if the GPU resource is no longer required.

Systems specifications: The computing environments for all experiments consist of various Spark cluster configurations on GPU optimized platforms. All Spark clusters are configured to take advantage of the NVIDIA DGX Systems as they deliver an integrated hardware and software solution thats been optimized to deliver faster time-to-solution with latest GPU resources. *Nvidia-DGX1 systems*: We first consider an environment with 3xNvidia-DGX1 machines each capable of a total of 80 threads from 40 cores via two Xeon CPU E5-2698 processors operating at 2.20 GHz with 512 G DDR4 of system RAM. Each machine has eight 16 GB Volta GPUs achieving a maximum graphics clock of 1530 MHz and a maximum memory clock speed of 877 MHz with a 300 W power cap on each GPU. The all SSD-based local storage has 500 G for the OS with an additional 7 TB on Raid 0 for data storage. Each machine is connected to the network file system (NFS) storage via a single 10 GB Ethernet network connection. In total, we have three such machines that are interconnected via 4 $\times$ 100 GB EDR IB ports (2 GPUs per IB connection). *Nvidia-DGX2 systems*: For the second environment, we consider cluster nodes setup on NVIDIA DGX2 systems, which each combine 16 GPUs fully interconnected via NVLink. The first node, a Nvidia-DGX2, can run 96 threads from 48 cores via two Xeon Platinum 8168 processors operating at 2.7 GHz with 1.5 TB of the DDR4 system RAM. The machine has sixteen 32 GB Volta 350 W GPUs with max clocks of 1597/958 (Graphics/Memory). Local storage consist of a 1 TB NVME Raid 1 boot partition and a 30 TB NVME Raid0 data partitoin offering maximum transfer speeds of approx 20 GB/s. The machine has 8 $\times$ 100 GB EDR IB Ports (2 GPUs per IB port). The second node, Nvidia-DGX2-H, similarly has 96 threads with 48 cores, however, with two Xeon Platinum 8174 operating at 3.1 GHz with 1.5 TB RAM DDR4 2666. The machine has sixteen 32 GB OCed Volta GPUs running at 450 W maximum power consumption with max clocks of 1702/1107 (Graphics/Memory), local storage consists of a 1 TB NVME Raid 1 boot partition, and a 30 TB NVME Raid0 data partition offering maximum transfer speeds of approx 20 GB/s. The machine equally has 8 $\times$ 100 GB EDR IB Ports (2 GPUs per IB port).

Experimental evaluations are conducted by first setting up a *single-Nvidia-DGX1 Spark cluster* as follows: a configuration of one master node and eight workers. The cluster is instantiated from singularity containers. Each worker is allocated 30 GB memory and 9 cores and is responsible for launching a single executor. At execution the master node runs the driver process whose allocated memory is 10 GB. Extending this configuration to a *three-node Nvidia-DGX1 Spark cluster,* we instantiate a singularity container with single master node and eight workers on one Nvidia-DGX1 machine and add two more singularity instances on two more additional machines each equipped to support eight workers with allocation of 30 GB memory and 9 cores per worker. Two more clusters, *single-node Nvidia-DGX2-H Spark cluster,* and *two-node Nvidia-DGX2 Spark cluster*, are setup in a similar manner, however, each worker is allocated 80 GB of memory and 6 cores per worker.

TABLE I
SOURCES AND NUMBER OF LABELLED DATA FOR TRAINING, VALIDATION, AND TESTING EVALUATION

Country/State	Data Split	
	Train samples	Validation/Testing samples
Ethiopia	2714	302
South Sudan	892	99
Yemen	4023	447
Zambia	1125	125
Alabama	0	156
Arizona	0	269
Puerto Rico	0	300
New Mexico	0	61

H. Case Study: Building Footprint Mapping

Country scale building footprint mapping fits the scope of both a data and compute-intensive application. We envision current deep learning algorithms for solving this case study benefiting from a hybrid combination of the in-memory computing capability of Apache Spark and high-performance computing hardware platforms.

Our experimental dataset consists of (0.3–0.7)-m resolution satellite imagery acquired by Digital Globe constellations, i.e., WorldView-2, and WorldView-3. These constellations provide high-resolution imagery that is suitable for pixel level semantic segmentation objects. As summarized in Table I, training and validation image data (using a 90%:10% split) is collected from several countries Ethiopia, South Sudan, Zambia. Each sample is a 500 × 500 pixel RGB image with an associated label mask. Importantly, we further explore out-of-country/out-of-sample testing regions from other different geographical areas. These areas consist of New Mexico of the United States, Puerto Rico, Alabama, and Arizona, and are used to demonstrate both the deployment and generalization performance of *RESFlow*. These testing sets also represent a more realistic use-case of the framework once placed in production where rapid inferencing is much needed. Table I summarizes the distribution of testing samples that are used from each of the out-of-sample locations.

I. Workflow Initialization

To initialize the *RESFlow* ensemble with all available training samples, we selected an optimal count of six buckets for the image and model galleries based-off the average intracluster variance measured over a range of cluster numbers used for the CEM. For each of these buckets, we trained four different convolutional neural networks for the building mapping task where training and validation data are from its corresponding image gallery bucket. We picked these CNNs, which each contain encoder and decoder paths, namely ResNet50-FCN, [35], U-net [10], Seg-Net [36], and DeepLab [37] without multiple feature fusion, based on various sizes of model (number of parameters to train) as well as their superior performance on semantic segmentation tasks. The number of parameters for these models are listed in Table II. This also showcases the flexibility of the modularized application space in *RESFlow* where researchers can easily set up the preferred algorithms to test the model gallery. In training, we utilize standard binary cross entropy as a cost function to guide model learning. During testing (performing inferencing) on the out-of-sample data, each of these trained models becomes responsible for independently performing building extraction on testing samples, which are assigned to its membership via the same process of image gallery mapping. When reporting results, we refer to this combined quorum of models as *RESFlow*. All of the CNNs were concurrently trained from scratch, without using any pretrained models. Hyper parameters are held constant through all experiments so that we limit number of variables in the experimental runs. The final selected hyper parameter values used are *learning rate* of *0.002*, *batch size* of *6*, and *Adam* as the optimizer. These values are reused across all the four architectures.

TABLE II
NUMBER OF PARAMETERS AND TRAINING TIME FOR USED CNNS

	Model parameters	time per sample(sec)
ResNet50-FCN	23,541,769	0.032
U-Net	13,395,329	0.068
Seg-Net	29,443,073	0.076
DeepLab	42,542,280	0.111

Speed decreases with model complexity.

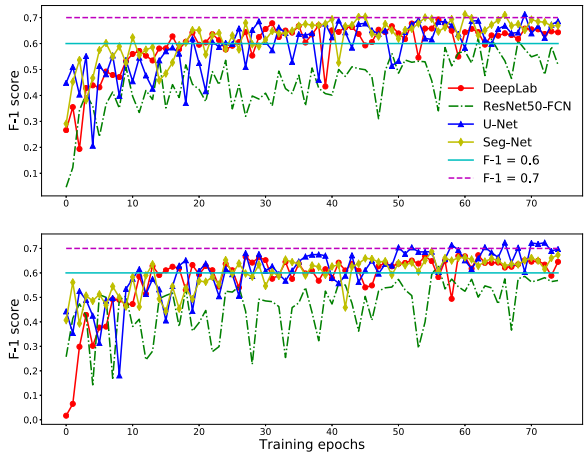


Fig. 7. F1 scores on validation set for the tested CNNs. The models were trained on a regular DGX-1 machine (top) and on a SPARK GPU cluster (bottom).

J. Performance and Computational Efficiency

Among the main contributions of this article, improving workload computational efficiency is vital in processing large volumes of data. RESFlow seeks to achieve this by combining algorithmic innovations as well as accelerated deep learning computing capable system architectures for remote sensing data analytics. More specific, its capability to partition data in the metric space offers desirable properties for large scale accelerated training and inferencing tasks. We observe and assess the system behavior under varying workloads and different computing environments to identify computational tradeoffs between constrained resources and need to process large volumes of imagery. The deep learning fully convolutional network of choice is the U-Net architecture [10], selected for its fast training convergence across the 60% F1-score on validation of 300 samples, as shown in Fig. 7.

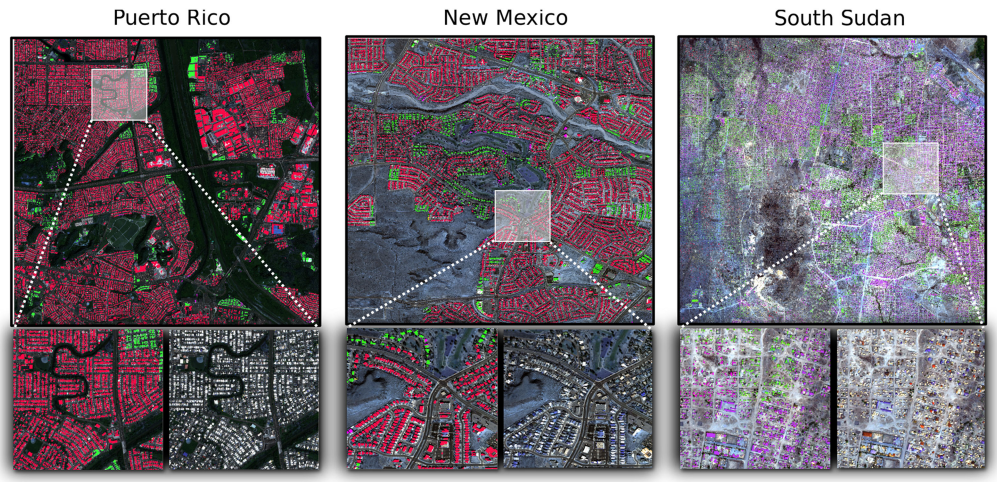


Fig. 8. Example building footprint mapping results for the *RESFlow* models on held-out New Mexico, Puerto Rico, and South Sudan data. Owing to the varying image characteristics and geographies image tiles clipped from same image scene are processed in parallel using different bucket models. Note: varying mask colours denote different bucket models for a total of six models from the model gallery.

TABLE III
COMPUTE EFFICIENCY (IN MINS) ON VARYING WORKLOADS ON ONE-NODE NVIDIA-DGX1 SPARK CLUSTER

	Files size(GB)					
	15.03	47.78	96.88	130.07	163.49	197.13
	Area (sq.km)					
Workers	313.97	1002.15	2035.66	2733.05	3433.78	4141.16
1GPU	3.81m	7.92m	13.93m	17.43m*	21.32m*	23.81m*
3GPUs	3.51m	5.35m	8.07m	11.22m	12.52m	12.77m
6GPUs	3.39m	4.38m	6.52m*	8.17m*	9.17m*	11.09m*
8GPUs	3.02m*	4.52m*	6.77m*	8.92m*	12.21m*	12.12m*

*Indicates a result for which soft GPU-executor assignment is needed to prevent run failure.

TABLE IV
RESULTS FOR HELD-OUT TESTING DATA

Region	Model	IoU	F1-score
Alabama	Mono	0.64	0.78
	RESFlow	0.63	0.77
Arizona	Mono	0.79	0.88
	RESFlow	0.76	0.86
Puerto Rico	Mono	0.54	0.70
	RESFlow	0.52	0.69
New Mexico	Mono	0.72	0.84
	RESFlow	0.62	0.77

To provide an appropriate comparison of the performance, we additionally train a single U-Net model, again using all available training samples with the identical fully convolutional network architecture. We refer to this monolithic network as the *Mono* model, and measure its performance across the entire out-of-sample test set. As metrics, we utilize the Intersection over Union (IoU) in addition to F1 scores in order to follow the accepted community standard [38] and report the performance results in Table IV. To quantitatively assert the viability of our proposed framework, Table IV presents the performance of both the Mono and *RESFlow* models on the held-out set of test data. Here, *RESFlow* is seen to perform very similarly to the Mono model for two of the three test regions. This is considerable, as each model from the *RESFlow* Image Gallery sees considerably less data compared to its Mono model counterpart during training, and yet is able to generalize to a similar degree. This is not true for the New Mexico region, however, with a large deficit in the performance being observed. Furthermore, visual maps to illustrate the performance of *RESFlow* ensemble of models over large geographic extents is shown in Fig. 8.

Tables III and VI illustrate the computational performance of deploying RESFlow on an Apache Spark cluster equipped with deep learning modules based on PyTorch and Tensorflow. The baseline is recorded to average 35 min for pixel-level semantic segmentation inferencing time for a single image scene of size 40 000 $\times$ 35 000 pixels and data volume of 11 GB processed on a single Nvidia DGX1 V100 GPU. Fig. 9 shows the speedup as a function of both the number of GPUs and data sizes in GigaBytes (computed from number of image scenes). The speedup is calculated as the ratio of the baseline execution time to the normalized compute time for given number of GPUs. For all different size workloads, we observe a tremendous speed up ranging from $9\times$ to over $400\times$ across all Spark cluster environments. Overall, Tables III–V and Fig. 9 demonstrate the speedup factors of the various cluster configurations.

We report on the strong and weak scaling of the segmentation inference module. The embedding and hashing modules are embarrassingly parallel with operations performed at image patch level to completion and always benefiting from additional numbers of GPU workers. To establish the strong scaling aspect of the workflow, Fig. 9 shows the speed-up performance over six workloads of data (between 1 and 12 image scenes) while varying the number of spark-based workers. The inference module is parallelizable while the merging of image patches to reconstruct the large scene is performed in a serial fashion; perhaps introducing an upper limit on the compute speed-up. An immediate observation is that compute speedup does not assume a linear increase with additional GPUs—parallelization efficiency decreases for each workload as the amount of GPU workers are increased. The presence of the increasing and

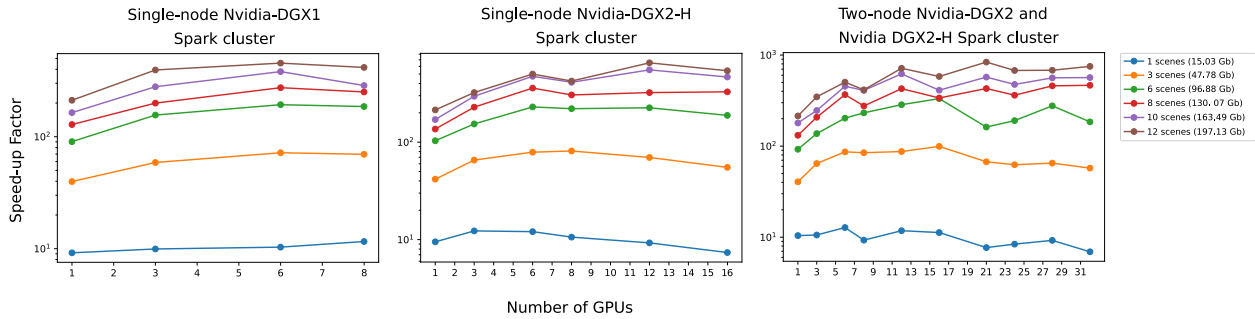


Fig. 9. Log-scale speed up of varying workloads when processed with *RESflow* parallel inference. Using serial processing single GPU baseline that averages inference speed of 35 min per 40 000 × 35 000 pixel image scene.

TABLE V
COMPUTE EFFICIENCY (IN MINS) ON VARYING WORKLOADS ON ONE-NODE NVIDIA-DGX2-H SPARK CLUSTER

	Files size(GB)					
	15.03	47.78	96.88	130.07	163.49	197.13
	Area (sq.km)					
Workers	313.97	1002.15	2035.66	2733.05	3433.78	4141.16
1GPU	3.69m	7.55m	12.19m	16.43m	20.50m	23.53m
3GPUs	2.87m	4.81m	8.19m	9.79m	11.82m	15.6m
6GPUs	2.9m	3.99m	5.48m	6.24m	7.38m	10.08m
8GPUs	3.3m	3.88m	5.71m	7.34m	8.5m	11.89m
12GPUs	3.78m	4.52m	5.59m*	6.94m*	6.34m*	7.73m*
16GPUs	4.75m*	5.71m*	6.69m*	6.82m*	7.50m*	9.32m*

*Indicates a result for which soft GPU-executor assignment is needed to prevent run failure.

TABLE VI
COMPUTE EFFICIENCY (IN MINS) ON VARYING WORKLOADS ON TWO-NODE NVIDIA-DGX2-H SPARK CLUSTER

	Files size(GB)					
	15.03	47.78	96.88	130.07	163.49	197.13
	Area (sq.km)					
Workers	313.97	1002.15	2035.66	2733.05	3433.78	4141.16
1GPU	3.36m	7.76m	13.64m	17.04m	19.49m	23.49m
3GPUs	3.31m	4.89m	9.18m	10.79m	14.22m	14.51m
6GPUs	2.74m	3.64m	6.21m	6.09m	7.7m	10.01m
8GPUs	3.77m	3.72m	5.43m	8.11m	8.55m	12.14m
12GPUs	2.97m	3.61m	4.42m	5.24m	5.63m	7.04m
16GPUs	3.11m	3.17m	3.79m	6.63m	8.51m	8.66m
21GPUs	4.55m	4.68m	7.78m	5.22m	6.11m	6.02m
24GPUs	4.17m	5.04m	6.61m	6.18m	7.36m	7.44m
28GPUs	3.79m	4.84m	4.55m	4.88m	6.23m	7.40m
32GPUs	5.06min	5.49m	6.82m	4.81m	6.18m	6.71m

decreasing trends on the performance curves could be explained by highlighting two aspects. The first is due to algorithmic implementation of our workflow. As aforementioned the most important contribution of our workflow is its ability to partition or tile large imagery and enable computing at scale. During inference execution on tiled partitions, worker processors do not communicate with one another, however on completion, we perform a `reduceByKey`[2] operation within Spark to group all tiles of the same image scene ID for merging into the large extent output mask. The merging operation for output mask reconstruction is performed by a single worker. As illustrated from Fig. 9, the performance bottleneck is more pronounced for workers executing on 6 GPUs or more. In addition, inferencing of fewer image scene appears to incur the largest compromise on speed up. The second aspect significantly influencing the observed performance bottleneck is attributed to increased I/O read activities. The computational efficiency introduced by RESFlow benefits tremendously from the lazy evaluation of Spark RDD data transformation. Throughout the pipeline, image data are only read from disk at inference time for each corresponding image tile. As a result, an increase in workers executing on more GPUs concurrently, thereby adding to the I/O read count, considerably compounds and causes a decrease in the compute performance. Tables III–VI further illustrate computational efficiency across a range imagery data sizes (or as measured in land area square kilometers). To establish weak scaling, we increase the workload on each GPU processor and also observe an improved weak scaling aspect of the workflow. Each row shows the compute time obtained for different area sizes for a fixed number of GPUs. The results indicate a desirable scaling factor, i.e., computational efficiency appears to increase as the land area to map (calculated for number of image scenes) correspondingly increases. For example, in Fig. 9, for 12 GPU-workers, the compute speed for an area of 313.97 sq.km (or a single image coverage with data size of 15.03 GB) is a $6.91\times$ improvement over the baseline of 35 min to process a single image scene. However, for an increased workload or land area of 4141.16 sq.km (or 12 image scenes with data size of 197.13 GB), the speed-up reaches $750\times$ over the baseline. This performance increase appears to be due to fact that overall communication and system bottlenecks are more dependent on the number of GPUs than on the land area.

We also evaluate the overall inference performance for varying input image size to the U-Net network architecture. Fig. 10 provides results for RGB image input of size $500 \times 500 \times 3$, $800 \times 800 \times 3$, $1000 \times 1000 \times 3$, and $1500 \times 1500 \times 3$. Inference is done in batches of size 12, 8, 5, and 2 image scenes, respectively. Compute efficiency is observed to vary with network completing inference at rate 1220 per seconds (or throughput of 3.7 GB/s) for image tiles of size $500 \times 500 \times 3$, which has an equivalent area rate of 23.45 sq.km/s. Given the smaller size in input image, I/O reads are increased putting a burden to the NFS file system. By considering much larger input images not only do we reduce the amount of I/O reads per image scene but we also increase both the throughput and the equivalent land area mapped. For example, with $1000 \times 1000 \times 3$ input images the network reaches a peak inference throughput of 6.59 GB/s

[2]While a typical `reduce` operation within Spark collects all data into a single data point, the `reduceByKey` collects all data into N distinct data points, where each datapoint is associated with a unique key from the data.

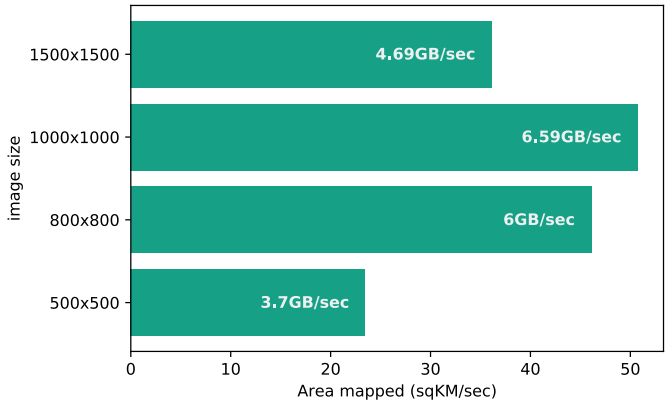


Fig. 10. Illustration of total area mapped per second for different image input sizes on a Nvidia-DGX2-H Spark cluster.

(processing a total of 564 images/s) and equivalence of mapping 50.78 sq.km/s. At this rate, we posit that the GPU compute time dominates the I/O read. However, when for much larger input image sizes, we note a throughput degradation to level of 4.69 GB/s (processing a total of 178 images/s) for $1500 \times 1500 \times 3$ tiles. Even though larger tiles increase the GPU utilization time they also require allocation of larger memory footprint for the inference results.

IV. Large Scale Experiments

A. Training

We first evaluate several aspects of the training phase in the SPARK-enabled GPU clusters. Taking advantage of the partitioned image gallery, there is no interaction between buckets, turning the model training into an embarrassingly parallel task. As shown in Table II, the training speed scales with the complexity of the model (i.e., number of trainable parameters). Also, note that the total time needed to achieve the model convergence is also linear to the number of training samples (total training time = number of training samples $\times$ seconds/ sample). In Fig. 7, we demonstrate the similar model training performance achieved by a regular DGX-1 and a SPARK-enabled DGX-1. Because of different network initialization conditions (as we trained the models from scratch), the learning curves are not identical under these two computing environments. Nevertheless, with the fixed training hyper parameters, we can see the best F1 scores obtained from both computing platforms for these four models are similar: Seg-Net and U-Net both deliver F1 scores slightly above 0.7, DeepLab has F1 scores close to 0.7, and ResNet50-FCN achieves F1 scores close to 0.65. In addition, the training improvements become trivial after $\sim$25 epochs for both computing environments except for ResNet50-FCN, which requires longer training epochs ($\sim$60 epochs) to get optimized parameters.

B. Inference

We here present the large scale inferencing results produced by the *RESFlow* model gallery on three states/countries: Puerto Rico, New Mexico of the United States, and South Sudan. Fig. 8 visualizes output building semantic segmentation maps across varying geographies that encompass the abovementioned three test sites. Finally, having evaluated the different components for *RESFlow,* e.g., impact of varying number of workers for different number of image to process in a single batch (see Fig. 9), establishing the throughput bounds as a function of input image size (see Fig. 10), we assess the scalability and applicability of deploying the workflow as depicted in Section II on **14** TB of imagery data covering the State of New Mexico. We select the Two-node Nvidia-DGX2 Spark cluster to execute the task with 28 GPUs and batch size of 12 for image scenes. The pipeline execution entails computing three deep learning tasks for each image tile: deep feature extraction stage, a deep hashing stage, and a deep semantic segmentation inference. The main goal for the large scale deployment was to assess the performance of our pipeline when deployed for production task. Therefore, we account for both read and write (I/O) bottlenecks in addition to the compute time. Table VII presents the throughput for this workload.

TABLE VII
RESFlow Deployment Performance on Two-Node Nvidia-DGX2 Spark Cluster

	Per second	Per day
Area mapped (sq.km)	5.245	453,168
Number of images	80	6,912,000
Total image data (GB)	0.243	21,028

This end-to-end inferencing workflow demonstrates large-scale processing of vast amounts of satellite imagery. Averaging a throughput of 0.243 GB/s (or 5.245 sq.km/s) inference of the entire set of 1440 image scene completed in 21 h—A tremendous achievement over a previous serial based inference workflow that would have taken over 28 days to complete.

V. Conclusion and Future Work

With a novel remote sensing data partitioning concept, this article presented a parallel inferencing workflow based on accelerated AI deployment hardware. Demonstrated on the pixel labeling challenge, we extended herein, Apache Spark based satellite imagery RDDs for processing with deep learning at scale and demonstrated compute efficiency across TB of high-resolution data covering land area equivalent of 787 300 sq.km. We achieve unprecedented pixel labeling area rates of 5.245 sq.km/s, amounting to 453 168 sq.km/day (or a daily capacity processing of 21 028 TB) demonstrating a reduction of a 28 day workload to 21 h. We further make an observation and identify workflow bottlenecks by studying compute speedup performance metrics as the amount of GPU workers are increased. In order to leverage Apache Spark to support deep feature learning and accommodate the problem of model generalization, the remote sensing data partitioning, the central concept in *RESFlow*, is realized from a combined set of four modules. *RESFlow* ignites optimism about a number of other relevant, and perhaps such new research directions, including enabling faster search for

retrieval of images with similar content, combining human geography and machine learning, e.g., establishing limiting bounds and the performance of models learned from finer abstract and deconflated geographical spaces, establishing robustness and stability of learning models for objects that are rare in some geographies and common in others. These directions involve studying varying heterogeneous levels of imagery data and their impact on training and inference tasks, and can potentially benefit from sensor and geographic agnostic workflows as compared to spatially constrained approaches. *Other future directions:* understanding the staging of geospatial workloads and deploying containerized workflows on supercomputing platforms presents a future research direction for our work. In addition, the work can be extended by exploring methodologies to enable better architectural design of computers specific to geospatial processing to benefit more applications.

ACKNOWLEDGMENT

Additionally, we would like to acknowledge that this manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

REFERENCES

[1] H. L. Yang, J. Yuan, D. Lunga, M. Laverdiere, A. N. Rose, and B. L. Bhaduri, "Building extraction at scale using convolutional neural network: Mapping of the united states," *IEEE J. Select. Topics, Appl. Earth, Observations, Sensing*, vol. 11, no. 8, pp. 2600–2614, Aug. 2018.

[2] L. Gueguen and R. Hamid, "Large-scale damage detection using satellite imagery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1321–1328.

[3] R. Hamid, S. O'Hara, and M. Tabb, "Global-scale object detection using satellite imagery," in *Proc. Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 40, no. 3, pp. 107–113, Jun. 2014.

[4] B. Bhaduri, E. Bright, P. Coleman, and M. L. Urban, "Landscan usa: A high-resolution geospatial and temporal modeling approach for population distribution and dynamics," *GeoJournal*, vol. 69, no. 1, pp. 103–117, Jun. 2007.

[5] B. Oshri *et al.*, "Infrastructure quality assessment in africa using Satellite imagery and deep learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, vol. 18, pp. 616–625.

[6] A. Albert, J. Kaur, and M. Gonzalez, "Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1357–1366.

[7] J. Sun *et al.*, "An efficient and scalable framework for processing remotely sensed big data in cloud computing environments," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4294–4308, Jul. 2019.

[8] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, "Bigearthnet: A large-scale benchmark archive for remote sensing image understanding," 2019, *arXiv:1902.06148*.

[9] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, May 7–9, 2015, 2015. [Online]. Available: http://arxiv.org/abs/1412.7062

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[12] Y. Li, Y. Zhang, X. Huang, H. Zhu, and J. Ma, "Large-scale remote sensing image retrieval by deep hashing neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 950–965, Feb. 2018.

[13] S. Roy, E. Sangineto, B. Demir, and N. Sebe, "Deep metric and hash-code learning for content-based retrieval of remote sensing images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2018, pp. 4539–4542.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[15] "Apache hadoop." 2019. [Online]. Available: http://hadoop.apache.org/docs/rl.2.1/hdfs_design.html

[16] F. C. Jie Li and L. Zhu, "Remote sensing image segmentation based on hadoop cloud platform," *Proc. SPIE*, vol. 10620, 2018, Art. no. 106200S. [Online]. Available: https://doi.org/10.1117/12.2283032

[17] R. Rajak, D. Raveendran, M. C. Bh, and S. S. Medasani, "High resolution satellite image processing using hadoop framework," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets*, Nov. 2015, pp. 16–21.

[18] Z. Lv, Y. Hu, H. Zhong, J. Wu, B. Li, and H. Zhao, "Parallel k-means clustering of remote sensing images based on mapreduce," in *Web Information Systems and Mining*, F. L. Wang, Z. Gong, X. Luo, and J. Lei, Eds. Berlin, Germany: Springer, 2010, pp. 162–170.

[19] J. Dean and S. Ghemawat, "Mapreduce: A flexible data processing tool," *Commun. ACM*, vol. 53, pp. 72–77, 2010.

[20] H. Shi, G. Hu, Jianfang Cao, M. Wang, and Y. Tian, "A new approach for large-scale scene image retrieval based on improved parallel k-means algorithm in mapreduce environment," *Math. Problems Eng.*, vol. 2016, 2016, Art. no. 3593975.

[21] W. Huang, L. Meng, D. Zhang, and W. Zhang, "In-memory parallel processing of massive remotely sensed data using an apache spark on hadoop yarn model," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 1, pp. 3–19, Jan. 2017.

[22] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, p. 10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863113

[23] M. Zaharia *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in in *Proc. 9th USENIX Symposium on Networked Systems Design and Implementation*, 2012, pp. 15–28.

[24] W. Jing, S. Huo, Q. Miao, and X. Chen, "A model of parallel mosaicking for massive remote sensing images based on spark," *IEEE Access*, vol. 5, pp. 18229–18237, 2017.

[25] Z. Sun, F. Chen, M. Chi, and Y. Zhu, "A spark-based big data platform for massive remote sensing data processing," in *Proc. Int. Conf. Data Sci.*, Aug. 2015, pp. 120–126.

[26] R. Pittman, H. Guan, X. Shen, S.-H. Lim, and R. M. Patton, "Exploring flexible communications for streamlining DNN ensemble training pipelines," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, 2018, pp. 64:1–64:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291742

[27] A. Mathuriya *et al.*, "Cosmoflow: Using deep learning to learn the universe at scale," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, 2018, pp. 65:1–65:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291743

[28] T. Kurth *et al.*, "Exascale deep learning for climate analytics," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, 2018, pp. 51:1–51:12.

[29] T. Prakash and A. C. Kak, "Active learning for designing detectors for infrequently occurring objects in wide-area satellite imagery," *Comput. Vis. Image Understanding*, vol. 170, pp. 92–108, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314218300390

[30] L. Yang, J. Shi, B. Chern, and A. Feng, "Open sourcing tensorflowonspark: Distributed deep learning on big-data clusters," 2017. [Online]: Available: https://developer.yahoo.com/blogs/157196317141/, Accessed: Jun. 2019.

[31] T. Hunter, "Tensorframes on google's tensorflow and apache spark," Bay Area Spark Meetup, 2016. [Online]: Available: https://databricks.com/session/tensorframes-deep-learning-with-tensorflow-on-apache-spark, Accessed on: Jun. 2019.

[32] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

[33] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *Proc Conf. Neural Info. Process. Syst. Workshop*, 2017, pp. 1–4.
[34] GDAL/OGR contributors, *GDAL/OGR Geospatial Data Abstraction Software Library*, Open Source Geospatial Foundation, 2018. [Online]. Available: http://gdal.org
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.
[36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[37] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
[38] I. Demir *et al.*, "DeepGlobe 2018: A challenge to parse the earth through satellite images," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 182–186.

Dalton Lunga received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2012.

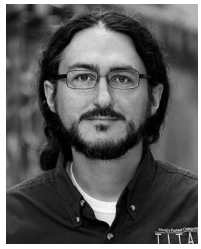
He is currently a Geospatial and Machine Learning Scientist within the Geographic Data Sciences Group, Oak Ridge National Laboratory. His technical background includes image processing, statistical machine learning, remote sensing, and geospatial data analysis. He currently conducts research and development in machine learning techniques and advanced workflows for handling large volumes of geospatial data. Prior to ORNL, he was a Machine Learning Research Scientist with the Council for Scientific and Industrial Research, South Africa on a variety of projects. His research interests are in domain adaptation,manifold learning, and unsupervised representation learning using deep learning approaches for geospatial imagery analytics.

Jonathan Gerrand received the B.Sc and M.Sc. degrees in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa, in 2015 and 2017, respectively. He is currently working toward the Ph.D. degree with the School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa.

He was an Intern with the National Emerging Technologies Division, Oak Ridge National Laboratory. His research focuses on attention-based modeling of domain-specific imagery.

Lexie Yang received the Ph.D. degree in civil engineering and Statistics Certificate in applied statistics from Purdue University, West Lafayette, IN, USA, in 2014.

She is currently a Research Scientist with the National Emerging Technologies Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. Her research focus on high-performance machine learning approaches for large scale data analysis, transfer learning and data fusion for multimodality/multisource remote sensing images, and automate feature learning with deep learning methods.



Christopher Layton received the B.A. degree in psychology from the University of Memphis, in 1998. He is currently a Linux Systems Engineer with the Compute and Data Environment for Science (CADES), Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, USA. He has over two decades of experience supporting Linux environments in both scientific and commercial arenas. His current work is in support of the scientific effort with ORNL. This work includes overseeing all aspects of a lab wide private cloud, HPC, and implementing and supporting unique systems such as the NVIDIA DGX1/2.

Robert Stewart received in Ph.D. degree in geography from the University of Tennessee, in 2014. He leads the Geographic Data Science Group, Oak Ridge National Laboratory, Oak Ridge, TN, USA. He is a Joint Faculty Assistant Professor in Geography with the University of Tennessee, Knoxville, TN, USA. His work spans a number of analytical domains including machine learning, spatio-temporal analytics, uncertainty quantification, geostatistics, data mining, and visualization. Areas of application range equally wide including population dynamics, socioeconomic analytics, urban dynamics, transportation, energy-water nexus, risk assessment, and decision support.