ARGUS: Rapid Wildfire Tracking using Satellite Data Collections

Saptashwa Mitra, Paahuni Khandelwal, Shrideep Pallickara, Sangmi Lee Pallickara
Department of Computer Science, Colorado State University, Fort Collins, USA
{sapmitra,paahuni,shrideep,sangmi}@colostate.edu

***Abstract*—Interactive visual analytics over distributed systems housing voluminous datasets is hindered by three main factors - disk and network I/O, and data processing overhead. Requests over geospatial data are prone to erratic query load and hotspots due to users' simultaneous interest over a small sub-domain of the overall data space at a time. Interactive analytics in a distributed setting is further hindered in cases of voluminous datasets with large/high-dimensional data objects, such as multi-spectral satellite imagery. The size of the data objects prohibits efficient caching mechanisms that could significantly reduce response latencies. Additionally, extracting information from these large data objects incurs significant data processing overheads and they often entail resource-intensive computational methods.**

Here, we present our framework, ARGUS, that extracts low-dimensional representation (embeddings) of high-dimensional satellite images during ingestion and houses them in the cache for use in model-driven analysis relating to wildfire detection. These embeddings are versatile and are used to perform model-based extraction of analytical information for a set of different scenarios, to reduce the high computational costs that are involved with typical transformations over high-dimensional datasets. The models for each such analytical process are trained in a distributed manner in a connected, multi-task learning fashion, along with the encoder network that generates the original embeddings.

***Index Terms*—embedding, multi-task learning, science-guided machine learning, visual analytics**

I. INTRODUCTION

Over the past decade, there has been a significant increase in the number of wildfire incidents across northern Europe and North America fueled by higher temperatures and drought. Heatwaves with prolonged dry conditions in western Europe resulted in increased wildfire activity and intensity; the ensuing wildfires burned 1.2 million acres over 4 summer months in 2022, which is more than double the average over the period 2006 – 2021 [1]. In Northern America, wildfires burned through 363,917 acres in California, USA alone [2], and over 3.9 million acres in Canada [3] in 2022. Forest fires have adverse effects like smogs from dense smokes on the urban fringes of forests leading to thousands of premature deaths [4].

Wildfires often occur in remote regions, where surveillance infrastructure is not always available, making remote sensing a viable option. To monitor and track wildfires, planet-scale geospatial observations such as satellite images have been widely used [5], [6]. These datasets offer critical capabilities to identify the locations of wildfires, discover underlying patterns and track their progress. However, identifying and retrieving highly relevant subsets of satellite images from voluminous satellite imagery in real-time is an extremely challenging task. Atmospheric conditions at proximate locations of an ongoing wildfire are often covered with smoke and temperature-based fire masking operations often create ambiguity in detecting the precise extent of the wildfire coverage [7].

Most satellite observations have associated identifiers such as timestamps and geospatial coordinates. Remotely sensed satellite images have low pass frequency (temporal resolution), which makes active fire detection challenging. Typically, the key indicator of the relevance between a particular phenomenon (e.g., wildfire) and observations, such as scientific labels or annotations, are not available in the low-level data product. It is quite common to see a small portion of highly relevant datasets labeled by the subject matter experts [8], [9] prior to being archived and hosted for broader dissemination. Since manual labeling is extremely time-consuming, this solution is not well-suited for scenarios such as navigating recent data collection or real-time monitoring. Similarly, it is hard to extend the existing set of labels for different use cases. Despite their low temporal frequency, periodically observed satellite imagery are often the most feasible as well as economical and less time-consuming solution for the analysis of phenomena such as wildfire, since they often occur in remote regions, than conventional methods of such as aerial images [10].

There have been several recent approaches to extracting patterns using machine learning to indicate the presence or absence of phenomena in observations [11]–[13]. However, since training and executing models over large amounts of data entails substantial computational requirements as well as increased network and disk I/O, ad-hoc integration of this approach into traditional data retrieval systems poses significant overheads in the latency for the operations and computing resources needed to complete them.

In this study, we present our framework, ARGUS, that allows rapid retrieval of satellite observations that are closely related to ongoing wildfire activities from unlabeled satellite imagery. ARGUS indexes satellite imagery using novel low-dimensional latent representations maintained in the memory of a distributed cluster, where the underlying datastore is continuously evolving with new observations continuously ingested. With our approach, a storage system is able to evaluate the user's spatiotemporal query with phenomena-specific keywords such as "has a wildfire", "wildfire affected area", or "level of severity of a wildfire" of "wildfire affected area" to retrieve more accurately relevant satellite tiles.

A. Challenges

Enabling rapid multi-faceted query analytics over high-dimensional voluminous datasets over a distributed system poses several challenges:
1) It is important to ensure that supporting frameworks or data structures needed for efficient retrieval of a related set of analytics over these large data objects have low memory utilization and can be generated/updated relatively quickly to adapt to the fast-evolving nature of the underlying data store.
2) Data pre-processing, be it in the form of efficient indexing, metadata extraction, or data-compression, to support interactive query analytics over high-dimensional data objects can negatively impact the data ingestion rate. Furthermore, it is important to ensure that the processed data retain enough information to be amenable for reuse in different analytical queries.
3) Efficient orchestration of clients' spatiotemporal query evaluations, such as mapping and executing individual queries to the list of data objects required for model evaluation, is a challenge. To significantly reduce the overall query latency, effective buffering/batching of image tiles as model inputs is necessary. Also, avoiding redundant or overlapping operations in a multi-user system introduces additional challenges.

B. Research Questions

Research questions that we explore in this study include
RQ1: How can we identify the existence or absence of wildfire (or relevant characteristics) automatically and rapidly without relying on human-generated annotations or labels? This also involves identifying multiple related characteristics such as the level of severity.
RQ2: How can we effectively index and rapidly evaluate geospatial queries over voluminous, high-dimensional satellite imagery while ensuring low memory footprints and computing requirements?
RQ3: How can we design an indexing scheme that flexibly increases the scope of the searchable target characteristics without adding a significant overhead for memory or computing capacity? The indexing scheme should also be able to cope with additional computing requirements for fire-tracking while maintaining a low memory footprint.

C. Overview of Methodology

We use multi-spectral satellite imagery as our proving ground and attempt to create versatile low-dimensional latent representations from the underlying image objects. These latent representations(embeddings) are loaded into a distributed in-memory cache and used to evaluate the level of relevance of the satellite tile to a wildfire event, while completely circumventing the need to access original data objects stored on disk. This approach enables us to (1) reduce memory footprints of the in-memory data objects, (2) improve the cache hit ratio, and (3) facilitate faster data processing and reduce response times.

The generated embeddings are amenable to analytics over a collection of related target phenomena by training a set of corresponding deep-learning models without generating a separate set of embeddings for each phenomenon. The embeddings are generated by an autoencoder network [14] that is composed of multiple model heads, where each head uses the generated embedding as input and is optimized to estimate the level of relevance to a target phenomenon. These models are trained simultaneously through multi-task learning [15] as a conjoined network to enhance the accuracy and convergence speed of the overall network. As a result, a single set of latent representations is versatile enough to serve multiple types of target geospatial events. The multi-task learning approach enables these networks to have better generalization throughout the shared representations. Training over these concise, representative inputs also speeds up the model training (faster convergence rates) while ensuring lightweight models.

We explore the problem of identification of wildfire regions from satellite imagery from the Visible Infrared Imaging Radiometer Suite (VIIRS) Thermal Anomalies (VNP14) [16] dataset using the available bands (e.g., thermal and infrared) as the inputs to our models. To evaluate the versatility of our embeddings with various characteristics of the wildfire, we have selected a set of related tasks that can benefit from a multi-task learning setup. The set of related tasks that we attempt to simultaneously train over our embeddings are - 1) classification of wildfire image tiles, 2) classification of wildfire severity, 3) bounding box object recognition for wildfire-affected regions, and 4) semantic segmentation of wildfire regions for each image tile.

Our benchmarks demonstrate that, with **a compression factor of** 120**x** for the generated embeddings, ARGUS segmentation models achieved an accuracy of 88% of that achieved with a SegCaps segmentation model trained with actual satellite image objects. Additionally, we demonstrate up to 27x reduction in evaluation times for varying levels of overlapping queries by fast identification of previously evaluated spatiotemporal domains and avoiding re-evaluation through our models.

D. Paper Contributions

Our framework facilitates model-driven value extraction over low-dimensional latent representations generated from multi-spectral images to reduce memory footprint and improve query evaluation speed for real-time wildfire monitoring. In particular, our contributions include:
1) Our model-based evaluation scheme accurately predicts the relevance of pixels to the ongoing wildfire event without human intervention. The framework generates a concise, representative latent representation of multispectral satellite images, and parameterizes them for machine learning models that evaluate relevance to the wildfire event.
2) Our approach significantly reduces the amount of data to be stored in-memory, while ensuring comparable accuracy to cases with actual data. Since satellite imagery is voluminous, repeated retrievals and the corresponding network and disk I/O would result in inefficiencies: our methodology circumvents this inefficiency.
3) Our framework allows the storage system to enhance the

set of target phenomena while still maintaining a single set of embeddings. A set of different deep learning networks, each optimized to evaluate relevance to different phenomena can be trained in parallel using multi-task learning [17].

E. Paper Organization

The remainder of this paper is organized as follows. Section II outlines related works, followed by the background in Section III that introduces the nature of the actions and spatiotemporal user queries. Section IV describes the various components of ARGUS's deep-learning architecture and its in-memory data model. Section V details our in-memory data store and its role in fast query evaluation. Experimental setups, performance benchmarks, and analysis of results are outlined in Section VI. Finally, Section VII outlines our conclusions, followed by acknowledgements in Section VIII.

II. Related Work

There have been several research studies on active fire detection and burned area segmentation to automatically identify the regions of an image that correspond to fire and separate them from non-fire regions. Dataset collections such as those from Landsat, Sentinel-2, and MODIS are popular open databases that have been used for model-driven monitoring of phenomena [18]–[20]. Specifically, semantic segmentation through deep-learning has commonly been applied to satellite images for fire detection due to their ability to learn complex relationships from multi-spectral data [21]–[23].

Caching of highly-requested data elements is a common strategy in enabling scalable visual analytics over voluminous datasets. Forecache [24] implements a prefetching scheme that predicts the data tiles to be queried in the future into the memory to improve latency. In other words, multivariate data tiles at various resolutions are precomputed to provide scalable panning and zooming as done in Google Maps [25]–[28]. Since disk I/O is significantly more time-consuming than in-memory operations, loading data objects in memory has been an effective strategy to reduce latency. However, this is not a feasible strategy if memory space is limited or if the data objects involved are large in size. A compressed representation of large data objects could be an alternative in such scenarios.

Deep neural networks-based autoencoders [29], [30] are commonly used for a number of different applications, including feature extraction and dimensionality reduction. The driving principle behind an autoencoder is that the high-dimensional data has a significantly smaller lower-dimensional embedding in a latent space that is sufficient to represent the information of the original data. The encoder part of the autoencoder compresses the input data into a bottleneck representation, which is then used by the decoder for reconstruction, the goal being to minimize the reconstruction error. Autoencoders have been used for data compression [31], [32] by training the network to learn a lower-dimensional representation of the input data. A common approach is to use a deep autoencoder network with multiple layers in the encoder and decoder. The deeper layers of the network learn higher-level features of the input data, which can be used to accurately reconstruct the original data with a lower-dimensional representation [33], [34].

Multi-task learning (MTL) [17] is an effective modeling technique where multiple models learn related tasks jointly to support a mutual exchange of knowledge that facilitates generalization. MTL allows the model to learn shared representations between tasks, which can lead to more efficient and effective learning [35]. Feature-based multi-task learning aims at learning common features through generalization among related tasks as a way to exchange common knowledge. Multi-task learning involves training of machine learning models with data from multiple tasks simultaneously, using shared representations. This enables the models to acquire shared knowledge between a set of related tasks and also improve its robustness by assimilating knowledge across multiple domains. These shared representations increase data efficiency and can potentially yield faster learning speed for related or downstream tasks, helping to alleviate the well-known weaknesses of deep learning: large-scale data requirements and computational demand. Additionally, MTL can also reduce the amount of data needed to train a model, as the model can use data from one task to improve performance on another task [36].

MTL has been successfully applied to the problem of object detection and semantic segmentation through models such as Faster R-CNN [37] and Masked R-CNN [38], where related tasks like object boundary detection and image segmentation are trained collaboratively through a shared trunk (backbone) followed by branched heads for downstream individual tasks. A potential pitfall of this approach is that training multiple models jointly can be both compute and resource-intensive since all the model layers combined have to be optimized simultaneously. This is especially true for deep learning models.

III. Background

In this study, we explore rapid analytical keyword query evaluation over voluminous multi-spectral satellite imagery. Each of these data-objects have high spatial resolutions and multiple data bands, making them large in size. Caching them in their original form would significantly strain cache capacity and negatively impact the hit-rate and the interactivity of spatiotemporal query analytics. Our goal is to ensure high fidelity for model-driven analytical information compared to geoprocessing over actual data objects. Scalable management of voluminous data collections [39], [40] underpins effective training of deep networks [41]; data accesses are also predicated on effective queries [42] and federation [43]. Several systems also rely on outlier detection [44] to preferentially identify training data of interest.

Useful analytical information can be extracted from these multi-spectral images using geoprocessing algorithms such as the computation of slopes, and curvatures from raster images using spatial tools provided in frameworks like ArcGIS [45]. However, these algorithms are often not optimized for parallel or GPU-driven execution, making them significantly slow,

especially when the number of candidate tiles required to be processed is large. This is particularly true in cases of sparse events like wildfires.

For instance, the following SQL query provides a sample of the type of spatiotemporal queries that the ARGUS framework aims to evaluate for multiple simultaneous users. In particular, we show a wildfire segmentation query for identifying and demarcating potential wildfire regions (*has_fire*) from multi-spectral satellite imagery dataset (VIIRS) over a given spatial and temporal range specified through the *Query_Polygon* and *Query_Time_Range*, respectively.

```
select has_fire(band_1, band_2, ..., band_n)
from VIIRS_Data
where coordinates in Query_Polygon
and time_stamp in Query_Time_Range
```

The evaluation of the above query over a distributed storage system would involve identifying image tiles with intersecting spatiotemporal bounds, evaluating their wildfire-affected regions, if any, and returning the compiled results back to the users. ARGUS attempts to speed up the above process by implementing the following - (1) the creation of embeddings out of image tiles for easier storage in a distributed cache for rapid identification, (2) training deep-learning analytical models that use embeddings as input, circumventing the need for on-disk data access as well as geoprocessing algorithms, (3) using a combination of classification models for identification of potential tiles with wildfires and running segmentation model on those only, and (4) using efficient in-memory caching and indexing schemes to avoid both disk access and re-evaluation of embeddings. Since events like wildfires are sparse, running a simpler classification model to weed-out unnecessary tiles can significantly improve interactivity.

IV. METHODOLOGY

A. System Overview

ARGUS is designed to work in conjunction with any distributed hash table (DHT)-based spatiotemporal storage system [46]. The overall ARGUS framework can be partitioned down into two main components -

1) ARGUSNET: A collection of deep learning models trained to perform encoding and keyword-based evaluation from the unlabeled satellite data collections, and

2) Hierarchical Embedding Store: A graph-based in-memory caching framework built to house latent representations generated by the ARGUSNET module.

Fig.1 shows the various components of our framework. The ARGUSNET models utilize data from the underlying DHT storage for their training through distributed modeling. Once trained, the ingestion module utilizes the encoder portion of the network to intercept data ingestion requests and house them in the hierarchical embedding store. The classification and segmentation models are used during query evaluations over the cached latent representations in the embedding store.

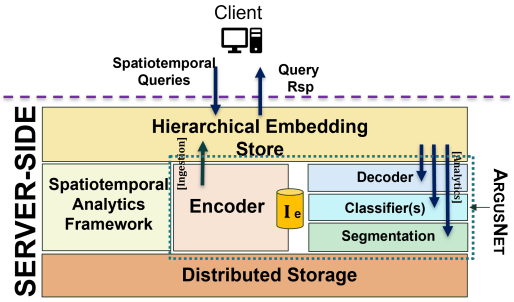

Fig. (1) ARGUS System Overview: Hierarchical Embedding Store is our distributed in-memory caching system. Encoder, Decoder, Classifiers and Segmentation constitute the various components of ARGUSNET

B. ARGUSNET

*1) **Model Overview**:* Ideally, the latent representations of our image tiles must be versatile enough to support multiple keywords (e.g., occurrence of wildfire and the level of severity) without maintaining multiple embeddings for each problem. To accomplish this, we train the models in a conjoined manner through multi-task learning and generate a single embedding for each tile that is used for multiple analytical models later on. Related tasks trained through multi-task learning have been shown to have better accuracy and convergence speed and as evidenced by our benchmarks. Our models demonstrate improved accuracy as well. Fig.2 depicts the overall model architecture. We can see that it consists of the following main components – an **autoencoder** network, **classification** networks, and a **segmentation network**. Additional models for the extraction of related analytical information from the embeddings can be added to our network as needed. Apart from the autoencoder network, all other networks (heads) use embeddings as their input.

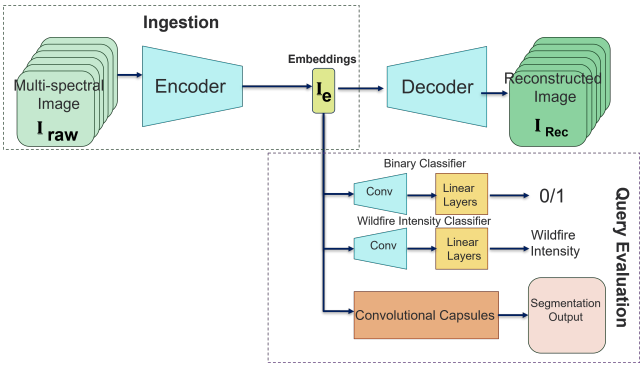

Fig. (2) ARGUSNET Architecture: Encoder forms the backbone of the network used during data ingestion to generated embeddings. Decoder, Classifiers, and Segmentation heads are used during query evaluations.

*2) **Model Input**:* Let us denote our multi-spectral image input as $\mathbf{I_{Raw}}$ - this is the input to our ARGUSNET network, which gets converted to its corresponding latent representation, denoted by $\mathbf{I_e}$, which is significantly smaller in size. In our case, $\mathbf{I_{Raw}}$ is complied by integrating various distributed spatiotemporal datasets and extracting relevant bands from them that are relevant to wildfire prediction. The target band, along with the classification labels is extracted by combining the fire-band available in the VIIRS dataset [47] along with historical wildfire perimeter (and duration) information to create a single-channel target mask for each image tile ($\mathbf{I_{Target}}$).

C. Selection of Training Data:

Our input is created through a combination of multiple remote-sensing data sources and includes bands relating to emissivity, soil moisture, vegetation index, and land cover type, all of which are known to be contributing factors that influence wildfire. In total, our input, $\mathbf{I_{Raw}}$, consists of 15 bands/channels.

Class imbalance is common in case of wildfires since the majority of the image tiles will not contain fire-pixels. In order to circumvent models prioritizing the majority class, we oversample the wildfire-containing tiles. We use the California Fire Perimeter Database [48] for historical information on wildfire perimeters and dates to identify tiles that have fire pixels in them. Additionally, to reduce the uncertainty in the training data, we ignore tiles that contain wildfires with an overall perimeter area of more than 10 km^2. Finally, we use a 1:1 distribution of fire and non-fire tiles in our training.

D. Network Architecture:

Our goal is to perform semantic segmentation through convolutional networks for the detection of wildfires from multi-spectral imagery. ARGUSNET consists of two main stages: a set of convolutional layers for feature extraction and a set of heads for performing reconstruction, classification, and segmentation. We explain each section of the overall deep learning model below.

Encoder: The encoder constitues the backbone of the ARGUSNET architecture. In the first stage, this encoder portion of our autoencoder, comprising a set of convolutional layers, generates a dense representation of a multi-spectral image tile. We expect these convolutional layers to take a multispectral image vector as input and encapsulate complex and abstract features from the input image for analytics. This extracted feature map serves as an input to the three heads of the ARGUSNET network.

The encoder network comprises a series of convolutional layers followed by a downsampling through max-pooling that incrementally reduces the spatial dimension while increasing the number of channels leading to bottleneck. We introduce batch normalization between the two layers to stabilize the training process to avoid bias during training by normalizing the input to each layer and accelerating the convergence speed of the training. The output of the encoder network produces our embeddings (I_e), a compressed representation of the abstract features of the input image (as shown in Fig.2).

The main goal is to be able to utilize the generated I_e for the extraction of multiple analytical observations. In order to make it versatile enough, we have to ensure that the training process takes into consideration the loss of each of these model-driven analytical tasks during the construction of the embedding and not just the reconstruction loss. **Multi-task learning** (MTL) is an effective approach to training and optimizing a combined model to perform multiple tasks simultaneously. This conjoined training methodology, where a cumulative loss from all the related tasks affects the weights of the network, enables the model to leverage shared information between tasks. This has been shown to produce better representation learning, regularization, transfer learning, and improved data efficiency. Multi-task learning can improve the accuracy of all models in several ways, including the ability to learn more general representations of the data, prevent overfitting, and facilitate the reuse of learned features for related tasks. Overall, training our models through the combined architecture, as shown in Fig. 2, can significantly improve the accuracy and generalization of machine learning models.

Decoder: The compressed latent representation is passed on to the decoder network, which uses upsampling of feature maps through a series of transposed convolutional layers (deconvolution) and increases the spatial dimensions of the data to eventually reconstruct the input ($\mathbf{I_{Rec}}$). Maintaining a decoder head trained for image reconstruction serves two purposes. First, it allows us to use the embedding to recreate the bands of the original image in case of queries over the actual bands. Secondly, it allows us to introduce new heads into the network while ensuring faster re-training convergence speed.

Classifier: The classifier heads are responsible for generating a probability distribution over either a binary flag that predicts whether an image tile has wildfire, or over the possible wildfire intensity classes. The classifier head takes the latent representation, I_e, as input and flattens it into a 1D vector. This vector then passes through a set of fully connected layers to produce a vector of scores, one for each object class. A softmax activation is applied to these scores to generate a probability distribution over the classes.

Semantic Segmentation: We implement a SegCaps [49] architecture as the deep semantic segmentation head for our network. We leverage capsules, which are a variation of a neuron or *instantiations* that represent a specific aspect of the object and can encapsulate various properties of an object, including, its spatial orientation, and scale. This feature of capsule network helps us adapt to wildfires of different geospatial scales and additionally leverages the capsule's ability to understand relative positions and orientations of objects in an image to train on wildfires which also have regional characteristics. Similar to [49], our network also contains a set of 8 primary capsules, followed by digit capsules as the output layer that generates the segmentation output. Another important benefit of using capsule networks for wildfire segmentation is their

ability to reduce the number of labeled datapoints for training, which is useful for wildfire events, that tend to be pretty sparse.

E. Loss Function:

Our multi-task loss function is a combination of losses from the heads of the ARGUSNET network. We explain each of these components below.

Reconstruction Loss: This is the loss component from the decoder head. Since we mainly want I_e to be useful in extraction of model-driven analytical information, we prioritize minimizing the reconstruction loss (Mean Squared Error) of bands that are more closely correlated to wildfires, such as NDVI, land-cover type, and soil-moisture. We update reconstruction loss and give more weight(W_1) to the bands with more correlation(B_1) to wildfire than the remaining bands(B_2), i.e., $W_1 > W_2$:

$$\mathcal{L}_{rec} = W_1 * \mathcal{L}_{rec}(I_{Rec}[B_1]) + W_2 * \mathcal{L}_{rec}(I_{Rec}[B_{Rem}])$$

Classification Loss: Our classification loss ($\mathcal{L}_{\mathbf{class}}$) is computed as Cross Entropy loss. For predicting the presence of a wildfire in a tile, we train using Binary Cross Entropy Loss (BCE) and for the multi-label prediction of fire-severity, we use BCE with Logits loss (combination of a Sigmoid layer and BCE Loss).

Segmentation Loss: Wildfire-affected regions can comprise varying portions of the pixels in an image. In case of smaller fire perimeters, we ensure that we avoid a biased segmentation model that prioritizes classifying the background pixels. Dice loss has been shown to be suitable for such class-imbalance problems [50]. So we make our overall segmentation loss ($\mathcal{L}_{\mathbf{seg}}$) a combination of the BCE loss and the Dice Loss.

The overall loss of the network is computed as follows:

$$\mathcal{L}_{\mathbf{rec}} = \mathbf{W_R} * \mathcal{L}_{\mathbf{rec}} + \mathbf{W_C} * \mathcal{L}_{\mathbf{class}} + \mathbf{W_S} * \mathcal{L}_{\mathbf{seg}}$$

The weights of each loss, i.e. W_R, W_C, W_S are hyperparameters that we optimize during the training process.

F. Distributed Training

Our ARGUSNET module is trained over a distributed spatiotemporal filesystem. The server-side cluster tracks the freshness of the trained models with respect to new, incoming data and triggers a fresh round of training iterations to further fine-tune old models.

The training of the models is agnostic of the underlying distributed file system. Our training leverages the spatiotemporal partitioning scheme of the storage system by collocating the training modules with the partitioned data to avoid data movements during training. The distributed training utilizes a parameter server to aggregate the weights asynchronously at certain intervals. We have used Pytorch Lightning [51] in the Distributed Data Parallel (DDP) mode for our distributed learning. Once trained, we use the encoder part of the network for our ingestion processes, while the classification and segmentation branches are used during query evaluations.

G. Hierarchical Embedding Store

The **Hierarchical Embedding Store** is a lightweight indexing scheme to better organize the in-memory latent representation of the on-disk images tile on each node in the distributed cluster. The embedding store is a decentralized in-memory metadata graph that holds the embedding object, $\mathbf{I_e}$, in its leaf nodes, along with other information that gets populated with subsequent query evaluations, such as, predicted fire-masks and flags marking the presence of wildfire pixels in the corresponding tile. The graph is organized based on the spatiotemporal metadata of the underlying tile, as shown in Fig.3 to facilitate fast query evaluation and identification of relevant in-memory embeddings for each node.

In addition to the Hierarchical Embedding Store, ARGUS maintains the trained modules, mentioned in the previous section, in-memory on each node for fast generation of embeddings (during ingestion) and for model-based query evaluation (during runtime).

V. DATA MODEL AND QUERY EVALUATION

Here, we explain the various stages of data ingestion into our embedding store and the subsequent process of querying analytical information out of it.

A. Dataset Description

Our input data is curated to incorporate attributes that are known to influence the likelihood and impact of wildfires in a region. In doing so, our approach leverages science-guided machine learning in our modeling for wildfire detection and segmentation. Science-guided machine learning [52] combines domain knowledge and scientific principles to enhance interpretability, reliability, and generalization of the trained models. In order to incorporate prior scientific knowledge into our model building, we integrate multiple data sources containing remote-sensing satellite data that provide global, near real-time information that includes active fires, thermal anomalies, and the Normalized Difference Vegetation Index (NDVI), which is computed from the red and near-infrared (NIR) bands of the VIIRS sensor [47]. Additionally, we incorporate relevant attributes such as land-cover type, soil moisture, and water retention, which are scientifically correlated with wildfires. By integrating these attributes, our model improves the accuracy of wildfire detection. The target segmentation mask is created by intersecting the active fire band from the remote-sensing data with historical wildfire perimeter information. For this study, we use the wildfires in California during 2019 [48] as our use-case.

B. Data Preprocessing and Partitioning

The raw data once downloaded and merged needs to be partitioned for efficient storage, distribution, and querying over a cluster. We split each multispectral image swath in terms of its geospatial hash (9-character quadtile key) for efficient indexing. These image tiles are then partitioned over the cluster based on their temporal metadata and quadtile key.

Quadtiles [53] is a hierarchical grid system that can recursively partition the overall geospatial coordinate space, incrementally, into a set of squares, each divided into four sub-squares of equal size. Each sub-square is assigned a unique code, which is appended to the unique code of the cumulative square, forming a unique hash string that represents the geospatial region contained within it. This quadtile key can be easily manipulated to identify both neighboring geospatial quadtiles/regions, along with encapsulated sub-regions.

In this work, we use the entire coordinate space of California as the overall geospatial region, represented by a single square with a key of "0", and recursively partition them into incrementally smaller quadtile boxes, each divided into four sub-squares, and appending each with either "0", "1", "2", or "3". The generated tiles are distributed among the cluster nodes based on their quadtile key and within each node, are organized based on their date and time of recording. This distribution scheme helps the zero-hop DHT efficiently identify relevant tiles both during training and query evaluation.

C. Embedding Store Population

The lightweight indexing scheme of the Hierarchical Embedding Store enables fast population of entries. During data ingestion, each incoming image tile, before being stored on disk, gets converted into its low-dimensional latent representation, $\mathbf{I_e}$, through the encoder module (E), and stored in-memory. The spatiotemporal information of the tile is used to add it to Hierarchical Embedding Store, with the reference to the in-memory $\mathbf{I_e}$ object being added to the leaf-node. This ensures co-location between the on-disk data objects and their latent counterparts and the embeddings follow the partitioning scheme of the underlying distributed system.

The creation of latent embeddings using the encoder, along with its population into the hierarchical embedding store constitutes the computation overhead during data ingestion. ARGUS ensures that this computational overhead is insignificant compared to the actual ingestion process by (1) ensuring that the tree-based structure of the embedding store facilitates fast indexing, (2) the encoder-decoder network is kept relatively shallow, and (3) the multiple incoming tiles are ingested as batched inputs to the encoder network for faster computation.

D. Containerized Data Ingestion

Since our query analytics framework relies on the effectiveness of in-memory embedding cache, along with the trained deep-learning models, we need to ensure that a sufficient amount of memory and computational resources is available at all times. We ensure that our data ingestion process, which also requires GPU for the encoder network, does not adversely affect the query analytics.

In order to ensure a scalable ingestion throughput, while maintaining an upper bound on resource utilization, we parallelize and containerize our data ingestion processes. Ingestion requests for each node in the distributed system are inserted into a job queue from which they are handled by one of our available ingestion processes. We use Kubernetes [54] replica sets to ensure parallelization. We set a limit on resource utilization by configuring maximum resource limits on the overall utilization of our data ingestion containers. The threshold is set at 10% of the overall CPU, memory, and GPU cores.

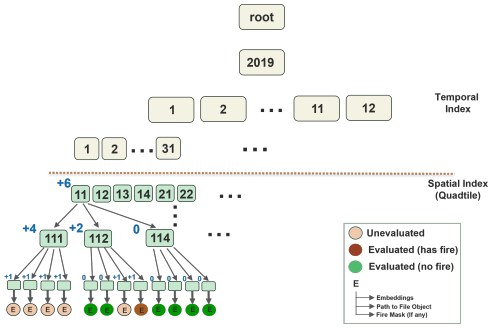

Fig. (3) Hierarchical Embedding Store

E. Query Evaluation

The Hierarchical Embedding Store enables fast identification of relevant embeddings for each spatiotemporal query on each node of the distributed cluster. In a cold start scenario, the graph is evaluated against the spatiotemporal bounds of the query in a top-down fashion to identify the latent embeddings that satisfy the specified predicate. These embeddings are meant to be probed for potential wildfire regions using our trained models. However, exhaustively evaluating all candidate tile embeddings against our segmentation mode, which has the most complicated architecture, for a sparse event such as wildfire might lead to prolonged response times. Rather, ARGUSNET maintains a simpler binary classification model for the identification of wildfires, which is first run to identify potential embeddings (containing wildfire) that get evaluated against the segmentation model. As we show in our benchmarks, this strategy leads to significantly reduced response times with comparable accuracy. Additionally, similar to the case of the encoder, the evaluation of embeddings for both classification and segmentation models is done in batches.

F. Avoiding Redundant Evaluations

Geospatial access patterns have been shown to follow spatial and temporal locality [55], i.e., at a given instant, users' queries over the entire dataset are focused on a small spatiotemporal neighborhood. While effective caching can leverage these patterns and greatly improve the hit-rates of our in-memory structures, this does not avoid redundant evaluation of our embeddings against the classification and the segmentation model, which require GPU resources.

The structure of our Hierarchical Embedding Store makes it conducive to support simultaneous query evaluations and collaborative analytics. Since it is structurally a feature graph, it can easily be traversed in a top-down manner while evaluating a spatiotemporal query. Fig.3 demonstrates our hierarchical

strategy of tagging potential wildfire nodes in the Hierarchical Embedding Store.

Upon evaluation of a tile against a spatiotemporal query, if a non-zero fire-mask is returned by our segmentation model, we store a compressed representation of the 2d array along with the embedding object in the leaf node. When no wildfire is detected, it will have a blank object and all unevaluated tiles have a null object attached to their leaf. This helps our framework avoid redundant computations of the same image tile against our trained models when there are subsequent overlapping requests. Additionally, it is to be noted that the memory overhead of having this fire-mask info is quite low - given the sparsity of the event, very few nodes will actually require the fire-mask object to be stored.

G. Optimized Graph Evaluation

Due to the large number of tiles (leaf nodes) that might be involved in queries over large spatiotemporal extents, we attempt to identify parent nodes that do not have any tiles of interest in their sub-tree. We keep track of these nodes through successive query evaluations over our emdedding store by maintaining an additional attribute (**node importance**) on each node of the Hierarchical Embedding Graph. This attribute signifies the combined number of unevaluated and wildfire tiles under each parent node in the graph - a non-zero importance value means that during evaluation, a parent node may contain a tile of interest in one of the leaves in its subtree. Over time, with a meaningful number of queries being evaluated over our system, a majority of these tags should amount to 0 (since wildfire is a sparse spatiotemporal event), which would help us avoid unnecessary traversals down the graph for irrelevant spatiotemporal extents.

Fig.3 demonstrates the update strategy of node importance during the evaluation of queries over ARGUS. When the segmentation output on a tile embedding is found to have no fire pixels, the node importance of its immediate parent is decremented. If the count of the parent is 0, we decrement the count of its immediate parent, and so on, up the graph. In general cases, this upward traversal of a tree would require bidirectional links or extra computation. Since our Hierarchical Embedding Store is a metadata graph, actual upward traversal up the tree can be avoided, since, given the spatiotemporal metadata of a node, we can easily deduce the exact parent node. In successive query evaluations, any node with importance of 0 will not need to be traversed further since it signifies non-fire tiles underneath.

H. Pruning: Node Replacement Strategy

Due to the limited size of the cache, we need an efficient strategy to maintain frequently accessed and relevant information in the cache in case of overflow. In order to facilitate interactivity in evaluations over the metadata graph, we need to maintain the most relevant regions in memory, and to efficiently detect stale nodes and swap them out for more requested regions, in case we breach the threshold due to overpopulation.

In case of an overflow, we utilize the **importance** attribute of a node in conjunction with the product of the number of accesses to a node (updated every time it gets accessed), and a time decay function that takes into consideration the last time the node was accessed. Using this metric, which we call `adjusted node imporance`, our cache pruning strategy takes into consideration both the relevance of a node at a particular instant alongside the contents of the sub-tree. Subtrees in ARGUS are replaced based on this adjusted importance score. To leverage the spatial and temporal locality of access patterns, when a request for a spatiotemporal region comes in, we mark both the set of Cells in that region and the immediate spatiotemporal neighborhood of that region as being of future interest as prescribed in [56].

VI. SYSTEM EVALUATION

A. Experimental Setup

To evaluate compute-intensive operations with high-density observations, we profiled our system while performing spatiotemporal queries with spatiotemporal data on a cluster of 50 nodes. Each node in our distributed cluster is an Intel Xeon E5-2620v3, with 64 GB RAM, each with a Quadro P2200 GPU (5GB of memory) with 1280 cores and several local 7200RPM SATA hard disks. The data is partitioned throughout the cluster uniformly based on the first 9 characters of their Quadtile key.

Throughout our experiments, we use 2 types of spatiotemporal queries - state-level and county-level. These represent 2 geospatial query sizes with a fixed temporal extent of 2 weeks. We experiment with these 2 ranges to demonstrate the scalability of our query evaluation. The state-level query has a $Query\_Polygon$ (see section III) that covers the state of California. The county-level query is set to mimic the size of an average county-wide region, with a latitudinal and longitudinal extent of 4° and 8° respectively.

B. Study Region and Datasets

The dataset we use is a composite from multiple remote-sensing satellite datasets. The first dataset we use is the VNP14 Active Fire Dataset [47], which provides global, and near real-time information on active fires and thermal anomalies. The dataset is produced by the Visible Infrared Imaging Radiometer Suite (VIIRS) instrument aboard the Suomi National Polar-orbiting Partnership (SNPP) and NOAA-20 satellites [57]. The data has a spatial resolution of 750 meters and is updated daily and encapsulates information regarding the location, temperature, radiative power, fire mask, and confidence level of active fires. We coalesce this dataset with VIIRS VNP21/NPP Land Surface Temperature and Emissivity 6-Min L2 Swath 750m [58] which contains information derived from satellite observations using physics-based algorithms. The dataset contains multiple data fields, including land surface temperature, emissivity, quality indicators, and other attributes for both active fire and non-active fire regions. Both datasets have a spatial resolution of 750 meters at daily temporal resolution.

TABLE (I) Breakdown of Data Staging: Comparison between time taken to download and pre-process the data against time to index and load it into ARGUS

	Fetch	Processing	Staging	Indexing
Time (Seconds)	132.07	539.53	303	**2.04**
Percentage	13.52%	55.24%	31.02%	**0.2%**

Normalized Difference Vegetation Index (NDVI) information is incorporated in our data using the VNP09 product, which provides the atmospherically corrected surface reflectance, derived from the VIIRS/NPP Atmospherically Corrected Reflectance (ACR) algorithm. It provides the surface reflectance values at a spatial resolution of 375 meters for the red and near-infrared (NIR) bands of the VIIRS sensor. The red band of the VIIRS sensor has a wavelength range of 0.6 to 0.7 micrometers and is sensitive to the reflectance of vegetation, soil, and water surfaces. The NIR band of the VIIRS sensor has a wavelength range of 0.84 to 0.88 micrometers and is sensitive to the reflectance of vegetation, especially the chlorophyll absorption feature. We downscale these 375m spatial resolution surface reflectance bands (I-bands) to integrate the information into our dataset. All the VIIRS [59] data products mentioned above are provided in NetCDF format tiles that are approximately 3248×3200 pixels in size, covering an area that is roughly 2436×2400 km^2. The scans of both satellites are co-located, that is, both satellites capture the same regions at the same timestamp, resulting in overlapping tiles that can be correctly merged together.

We also incorporate land-cover type information using the National Land Cover Database (NLCD) [60], as well as attributes relating to soil moisture, erodibility, and water capacity through the STATSGO [61] dataset. Since the land-cover and soil-related attributes are static in nature, they are generated only once for each spatial bound of the image tiles through the spatial analytics tool over ESRI's ArcMap. We use the California Fire Perimeter Database from the Fire and Resource Assessment Program [48] to create the fire mask using the methodology explained in section V. The database includes records of perimeters of wildfires that occurred in the state of California between the years 1950 and 2019 (inclusive). Our merged tiles are split into non-overlapping bounds for regions with quadtiles hashes 9 characters in length over California and uniformly distributed across a cluster of 50 machines. Our quadtile distribution is deterministic such that neighboring tiles are co-located on the same machine. We leverage data locality by avoiding any network I/O to transfer input tiles by locally reading data available on each machine.

C. Model Training: Rate of Convergence

Fig.4 shows the rate of convergence of our MTL setup with encoder output connected to the decoder, segmentation, and classification heads. We profile the segmentation loss, which is a combination of Binary Cross Entropy and Dice Loss for our predicted wildfire-affected area (mask) for both our training and validation data.

TABLE (II) Comparison of ARGUSNET Evaluation Performance against Standalone Segmentation Model

	Convergence Time(minutes)	Epochs
ARGUSNET	6.31	31
SegCaps	155.8	164

We compare the performance of our multi-task learning setup with a Classifier, Decoder, and Segmentation head against that of the dedicated SegCaps modeled after [49]. We can see, from Table II that the model stabilizes at around 31 iterations. The convergence of the SegCaps model is significantly slower, around 155 minutes and 164 epochs, which can be attributed to both the larger input size and the number of model parameters involved – the **total number of optimizable parameters** for our MTL setup was $\mathbf{47.35}\%$ that of a standalone SegCaps model for image segmentation.

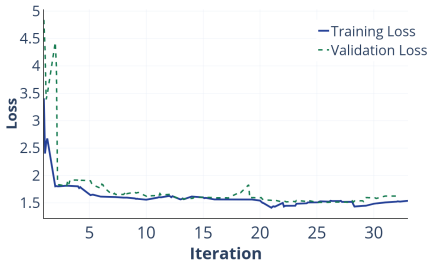

Fig. (4) Convergence Speed of Model: Variation of training and validation error for ARGUSNET over epochs.

D. Model-driven Evaluation Latency vs Query Accuracy

We demonstrate the improvement in evaluation latency of ARGUSNET over a standalone SegCaps model by profiling the average evaluation latency of a single-batched input. As expected, due to the simpler structure of the Segmentation head compared to a standalone SegCaps, the evaluation latency of **our network is ∼27x faster** than that of the SegCaps using raw image (Table III). This improved latency is achieved while maintaining a comparable accuracy, as shown in Table III; the accuracy achieved by the ARGUSNET model is nearly 88% of that achieved by the SegCaps model. The accuracy can be attributed to the stability that model training of related tasks through multi-task learning provides.

E. Data Ingestion Latency

We profile the ingestion rate of our framework. Since our data is a combination of multiple satellite datasets, we have evaluated the time taken for downloading, pre-processing, staging, and indexing time of a day's worth of data over the entire state of California. ARGUS is responsible for the indexing phase of this process, where embeddings are generated and entries are populated into the ARGUS metadata graph. Table I demonstrates a breakdown of time for these ingestion-related operations. We can see that the indexing phase of the process is

very fast and insignificant (0.2%), compared to the remainder of the process, which requires downloading and processing of satellite images.

Fig. 5 shows a further breakdown of the indexing process. Here we index and load a total of 5000 image tiles into our distributed in-memory graph and evaluate the overall throughput on each node. Here, we show the overhead of the encoding process that creates embeddings for each incoming image tile and stores a reference of the embedding object on the in-memory graph. Our encoder is simple enough and combined with the batched computation of embeddings, we can see that the indexing throughput is only reduced by 16.7%.

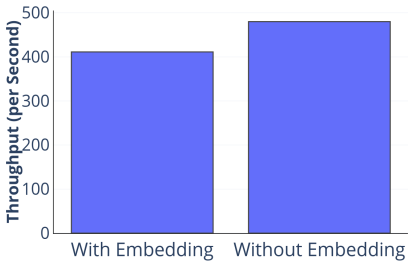

Fig. (5) Ingestion Throughput With/Without Embedding: Comparison of throughput of indexing the in-memory metadata graph with and without generation of embeddings through encoder during ingestion.

F. Query Evaluation

We demonstrate the scalability of our model-driven query evaluation in Fig.6. We profiled the average query latency for state and county-level spatiotemporal queries for 2 scenarios - one over spatiotemporal regions where we know had wildfires and second over random spans and regions. We execute 1000 different queries over our cluster and evaluated the average response time at a client node.

As expected, state-level queries take longer to evaluate than county-level queries, but the average query time is reasonable. Additionally, we can see that in fire-prone scenarios, the query latency is higher than in average case scenarios, since the number of tiles that are actually subjected to the segmentation model is significantly low, due to the classifier model filtering them out. The box-plots in Fig.6 show that a majority of the queries have significantly lower query latency in the average case since the majority of the spatiotemporal queries have no wildfire in them and most of the server-side overhead is simply from evaluation and classification over the in-memory metadata graph.

Fig.7 demonstrates the utility of having a trained classifier module to filter candidate tiles before being subject to segmentation models. The evaluation was done using spatiotemporal queries over a set of county-level wildfire scenarios. We see a significant improvement in average query evaluation latency in Fig.7 for evaluation with a filtering using classifier compared

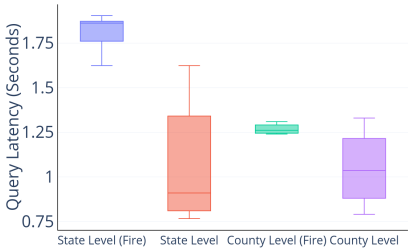

Fig. (6) Query Latency vs Query Size: Evaluation of increase in latency with the scale of the query's spatiotemporal extent.

TABLE (III) Comparison of ARGUSNET Evaluation Latency against Standalone Segmentation Model: ARGUSNET-based evaluation is 27x faster

	Eval. Latency	Error(BCE)
ARGUSNET	0.0026	0.195
SegCaps	0.054	0.1714

to segmentation-driven evaluation for all candidate tiles. This is due to the much simpler structure of the classifier network than the segmentation head of the ARGUSNET.

G. Avoiding Redundant Computations

Fig.8 demonstrates the effectiveness of our framework in avoiding duplicate evaluations for overlapping queries from multiple user requests. We compare the average query evaluation latency for a county-size query for wildfire regions at various levels of cache population. Fig.8 shows the query evaluation times for a cold-start scenario, where no embedding has been evaluated yet, a 50% evaluated tree, where we remove half of the evaluated nodes, and a case where all candidate tiles in a spatiotemporal query have evaluated entries in the hierarchical metadata graph. We can see that there is a significant improvement in query latency for overlapping queries and our framework effectively identifies and avoids redundant tile evaluation.

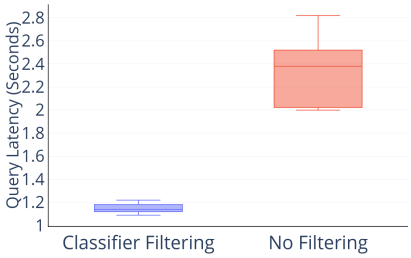

Fig. (7) Latency Improvement With Classifier Head: Comparison of latency of a State-level query with and without running candidate tiles through a classifier first.

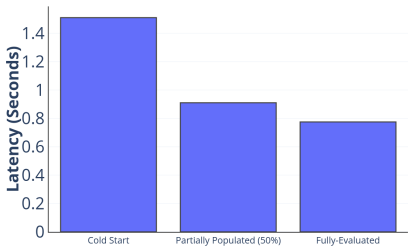

Fig. (8) Improvement in Query Latency with cached evaluations from historical queries.

VII. Conclusion

We described our methodology to track key characteristics of wildfires over unlabeled high-dimensional satellite image data collections and enable keyword search for image tiles in a distributed storage system.

RQ1: Our approach leverages multiple machine-learning based models to evaluate keyword search queries and the models demonstrate reliable accuracy. Instead of storing original image tiles in memory, Argus maintains representative (space-efficient) embeddings in memory and these embeddings are used as inputs for the model execution as part of keyword query evaluation. Our empirical evaluation demonstrates that the multi-task learning effectively trains the encoder network that generates a single set of embeddings for multiple wildfire keywords.

RQ2: Our distributed, in-memory hierarchical embedding store is structured in the form of a metadata graph for fast evaluation of spatiotemporal queries and identification of candidate embeddings. Also, the collaborative update of the node importance metric of each node in the graph with successive evaluations leads to a further reduction in evaluation time for future queries.

RQ3: Our approach reduces the required memory footprints significantly; this, in turn, allows the number of data objects indexed in the memory to increase substantially. More importantly, these embeddings reduce model complexity significantly by lowering the number of parameters. This allows our models to consider larger spatiotemporal extents to capture more comprehensive conditions from the surrounding area during keyword query evaluations.

VIII. Acknowledgement

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

References

[1] *Europe's summer wildfire emissions highest in 15 years*, 2022, https://atmosphere.copernicus.eu/europes-summer-wildfire-emissions-highest-15-years.

[2] *2022 Incident Archive*, 2022, https://www.fire.ca.gov/incidents/2022/.

[3] *National Wildland Fire Situation Report*, 2022, https://cwfis.cfs.nrcan.gc.ca/report.

[4] S. N. Koplitz, L. J. Mickley, M. E. Marlier, J. J. Buonocore, P. S. Kim, T. Liu, M. P. Sulprizio, R. S. DeFries, D. J. Jacob, J. Schwartz *et al.*, "Public health impacts of the severe haze in equatorial asia in september–october 2015: demonstration of a new framework for informing fire management strategies to reduce downwind smoke exposure," *Environmental Research Letters*, vol. 11, no. 9, p. 094023, 2016.

[5] E. Chuvieco Salinero, F. Mouillot, G. R. Van Der Werf, J. San Miguel, M. Tanasse, N. Koutsias, M. García Alonso, M. Yebra Álvarez, M. Padilla Parellada, I. Gitas *et al.*, "Historical background and current developments for mapping burned area from satellite earth observation," 2019.

[6] D. Fornacca, G. Ren, and W. Xiao, "Performance of three modis fire products (mcd45a1, mcd64a1, mcd14ml), and esa fire_cci in a mountainous area of northwest yunnan, china, characterized by frequent small fires," *Remote Sensing*, vol. 9, no. 11, p. 1131, 2017.

[7] J. Engelbrecht, A. Theron, L. Vhengani, and J. Kemp, "A simple normalized difference approach to burnt area mapping using multi-polarisation c-band sar," *Remote Sensing*, vol. 9, no. 8, p. 764, 2017.

[8] U. FEMA, *OpenFEMA data sets*, 2022, https://www.fema.gov/about/reports-and-data/openfema.

[9] J. Welty and M. Jeffries, *Combined wildfire datasets for the United States and certain territories, 1878-2019: US Geological Survey data release.*, 2020, https://www.usgs.gov/data/combined-wildfire-datasets-united-states-and-certain-territories-1878-2019.

[10] M. H. Ismail and K. Jusoff, "Satellite data classification accuracy assessment based from reference dataset," *International Journal of Geological and Environmental Engineering*, vol. 2, no. 3, pp. 23–29, 2008.

[11] Y. Ban, P. Zhang, A. Nascetti, A. R. Bevington, and M. A. Wulder, "Near real-time wildfire progression monitoring with sentinel-1 sar time series and deep learning," *Scientific reports*, vol. 10, no. 1, p. 1322, 2020.

[12] Z. Tang, X. Liu, H. Chen, J. Hupy, and B. Yang, "Deep learning based wildfire event object detection from 4k aerial images acquired by uas," *AI*, vol. 1, no. 2, pp. 166–179, 2020.

[13] H. U. A. Tahir, A. Waqar, S. Khalid, and S. M. Usman, "Wildfire detection in aerial images using deep learning," in *2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*. IEEE, 2022, pp. 1–7.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2018.

[16] W. Schroeder and L. Giglio, "Viirs/npp thermal anomalies/fire 6-min l2 swath 750m v001," *NASA EOSDIS Land Processes DAAC*, 2017.

[17] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[18] G. H. de Almeida Pereira, A. M. Fusioka, B. T. Nassu, and R. Minetto, "Active fire detection in landsat-8 imagery: A large-scale dataset and a deep-learning study," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 178, pp. 171–186, 2021.

[19] P. Borrelli, L. A. S. Rondón, and B. Schütt, "The use of landsat imagery to assess large-scale forest cover changes in space and time, minimizing false-positive changes," *Applied Geography*, vol. 41, pp. 147–157, 2013.

[20] K. S. Yankovich, E. P. Yankovich, and N. V. Baranovskiy, "Classification of vegetation to estimate forest fire danger using landsat 8 images: Case study," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[21] D. Rashkovetsky, F. Mauracher, M. Langer, and M. Schmitt, "Wildfire detection from multisensor satellite imagery using deep semantic segmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 7001–7016, 2021.

[22] C.-Y. Chiang, C. Barnes, P. Angelov, and R. Jiang, "Deep learning-based automated forest health diagnosis from aerial images," *IEEE Access*, vol. 8, pp. 144 064–144 076, 2020.

[23] R. Ghali, M. A. Akhloufi, M. Jmal, W. Souidene Mseddi, and R. Attia, "Wildfire segmentation using deep vision transformers," *Remote Sensing*, vol. 13, no. 17, p. 3527, 2021.

[24] L. Battle, R. Chang, and M. Stonebraker, "Dynamic prefetching of data tiles for interactive visualization," in *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2016, pp. 1363–1375.

[25] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2456–2465, 2013.

[26] C. A. Pahins, S. A. Stephens, C. Scheidegger, and J. L. Comba, "Hashedcubes: Simple, low memory, real-time visual exploration of big data," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 671–680, 2017.

[27] W. Tao, X. Liu, Ç. Demiralp, R. Chang, and M. Stonebraker, "Kyrix: Interactive visual data exploration at scale." CIDR, 2019.

[28] L. Santos, J. Coutinho-Rodrigues, and C. H. Antunes, "A web spatial decision support system for vehicle routing using google maps," *Decision Support Systems*, vol. 51, no. 1, pp. 1–9, 2011.

[29] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine learning*. Elsevier, 2020, pp. 193–208.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[31] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[32] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium (PCS)*. IEEE, 2018, pp. 253–257.

[33] S. Mitra, D. Rammer, S. Pallickara, and S. L. Pallickara, "Glance: A generative approach to interactive visualization of voluminous satellite imagery," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 359–367.

[34] ——, "A generative approach to visualizing satellite data," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2021, pp. 815–816.

[35] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

[36] B. Zoph, D. Yuret, J. May, and K. Knight, "Transfer learning for low-resource neural machine translation," *arXiv preprint arXiv:1604.02201*, 2016.

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[38] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[39] S. L. Pallickara, S. Pallickara, and M. Pierce, "Scientific data management in the cloud: A survey of technologies, approaches and challenges," *Handbook of Cloud Computing*, pp. 517–533, 2010.

[40] S. Mitra, Y. Qiu, H. Moss, K. Li, and S. L. Pallickara, "Effective integration of geotagged, ancilliary longitudinal survey datasets to improve adulthood obesity predictive models," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1738–1746.

[41] S. Mitra, M. Warushavithana, M. Arabi, J. Breidt, S. Pallickara, and S. Pallickara, "Alleviating resource requirements for spatial deep learning workloads," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2022, pp. 452–462.

[42] M. Malensek, S. Pallickara, and S. Pallickara, "Fast, ad hoc query evaluations over multidimensional geospatial datasets," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 28–42, 2015.

[43] M. Malensek, S. L. Pallickara, and S. Pallickara, "Hermes: Federating fog and cloud domains to support query evaluations in continuous sensing environments," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 54–62, 2017.

[44] W. Budgaga, M. Malensek, S. Lee Pallickara, and S. Pallickara, "A framework for scalable real-time anomaly detection over voluminous, geospatial data streams," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4106, 2017.

[45] Esri, *An overview of the Spatial Analyst toolbox*, https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/an-overview-of-the-spatial-analyst-toolbox.htm.

[46] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[47] W. Schroeder, P. Oliva, L. Giglio, and I. A. Csiszar, "The new viirs 375 m active fire detection data product: Algorithm description and initial assessment," *Remote Sensing of Environment*, vol. 143, pp. 85–96, 2014.

[48] (2023, March) Fire perimeters. [Online]. Available: https://frap.fire.ca.gov/frap-projects/fire-perimeters/

[49] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in neural information processing systems*, vol. 30, 2017.

[50] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. Ieee, 2016, pp. 565–571.

[51] W. Falcon, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[52] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Transactions on knowledge and data engineering*, vol. 29, no. 10, pp. 2318–2331, 2017.

[53] (2018, November) Quadtiles. [Online]. Available: https://wiki.openstreetmap.org/wiki/QuadTiles

[54] M. Luksa, *Kubernetes in action*. Simon and Schuster, 2017.

[55] D. Fisher, "Hotmap: Looking at geographic attention," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 6, pp. 1184–1191, 2007.

[56] S. Mitra, P. Khandelwal, S. Pallickara, and S. L. Pallickara, "Stash: Fast hierarchical aggregation queries for effective visual spatiotemporal explorations," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019, pp. 1–11.

[57] C. Cao, F. J. De Luccia, X. Xiong, R. Wolfe, and F. Weng, "Early on-orbit performance of the visible infrared imaging radiometer suite onboard the suomi national polar-orbiting partnership (s-npp) satellite," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 2, pp. 1142–1156, 2013.

[58] Hulley, G., Hook, S., Distributed by NASA EOSDIS Land Processes DAAC., "VIIRS/NPP Land Surface Temperature and Emissivity 6-Min L2 Swath 750m V001 [Data set]." https://doi.org/10.5067/VIIRS/VNP21.001, 2012.

[59] Schroeder, W., L. Giglio, NASA EOSDIS Land Processes DAAC., "VIIRS/NPP Thermal Anomalies/Fire 6-Min L2 Swath 750m V001 [Data set]," https://doi.org/10.5067/VIIRS/VNP14.001, 2012.

[60] (2016) National land cover database 2016 (nlcd2016) legend. [Online]. Available: https://www.mrlc.gov/data/legends/national-land-cover-database-2016-nlcd2016-legend

[61] G. E. Schwarz and R. Alexander, "State soil geographic (statsgo) data base for the conterminous united states," Tech. Rep., 1995.