Lightweight, Embeddings Based Storage and Model Construction Over Satellite Data Collections

Kevin Bruhwiler, Paahuni Khandelwal, Daniel Rammer,
Samuel Armstrong, Sangmi Lee Pallickara, Shrideep Pallickara
Department of Computer Science
Colorado State University
Fort Collins, Colorado, 80521
Kevin.Bruhwiler@rams.colostate.edu, paahuni@colostate.edu, rammerd@rams.colostate.edu,
Sam.Armstrong@rams.colostate.edu, Sangmi.Pallickara@colostate.edu, Shrideep.Pallickara@colostate.edu

***Abstract*—There has been a substantial growth in remotely sensed hyperspectral satellite imagery. These data offer opportunities to understand phenomena and inform decision making. The nature of these collections introduces challenges stemming from their volumes, variety, and spatiotemporal resolutions. The crux of this study is to facilitate effective training of deep learning models over satellite data collections. We describe our novel embeddings (multidimensional latent space representations) based approach to effectively support model training, refinement, and inferences. We rigorously explore several aspects relating to embeddings, including their dimensionality, single vs multiple bands, and preservation of inter-band metrics. We also incorporate support for transfer learning over spatiotemporal scopes to address issues relating to cold start and alleviate resource pressure. Our methodology addresses disk, network, CPU/GPU, and accuracy implications of several aspects relating to model construction. Our empirical benchmarks assess the suitability of our methodology using the MODIS and Sentinel-2 satellite data. We demonstrate that our methodology reduces storage requirements by more than 10,000x and reduces model construction times by 75%.**

***Index Terms*—hyperspectral satellite imagery, deep learning models, spatial computing, storage systems, spatiotemporal transfer learning**

1. Introduction

Over the past few decades there has been an exponential growth in satellite data volumes. This growth has been driven by the proliferation of satellites and advances in the sophistication of on-board sensing equipment. Different satellites have varying resolutions at which they perform sensing operations. In several cases, multiple bands (corresponding to varying wavelengths) are sensed simultaneously during the same sweep. Furthermore, for various wavelengths (or bands) at which sensing is performed, the resolutions may change. This results in each pixel (across bands) representing a different spatial scope.

Scientists build models over this data, which are used to make sense of the phenomena and to make forecasts or draw inferences. The complexity of the model building process is high because of the number of bands within the optical imagery and also because of differences in the resolutions across these bands. As a result, preprocessing overheads are quite high.

The crux of this paper is to facilitate construction of deep learning models at scale over satellite data. We consider data storage, preprocessing overheads, workload orchestration, and effective support for transfer learning during model building.

1.1. Challenges

There are several challenges in constructing models, in particular deep learning models, over voluminous satellite data.

1) Not only is the data voluminous, they are also in diverse formats. These are compounded by the fact that bands within the same spatial extent may be at different resolutions. As a result, data preparation and preprocessing costs in these settings are high.
2) Model building is an iterative process involving tuning parameters and the structure of the underlying networks. As a result, duplicate preprocessing requirements are quite high.
3) Building deep learning based models is computationally expensive. To be effective, model training must leverage co-processors and operate in a distributed environment.

1.2. Research Questions

The overarching theme in this study is to facilitate effective construction of deep learning based models over satellite data collections. We have identified three key research questions within this broader context that we explore as part of this study.

RQ-1: *How can we account for systems implications during model construction (training, inferences, and refinement)?* This encompasses reconciling the heterogeneity and volumes inherent in satellite data collections. Furthermore, to preserve scaling characteristics it is important to reduce performance hotspots, amortize processing workloads, and ensure concurrency.

RQ-2: *How can we support effective and expedited model construction?* This involves reducing cold start times

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

for model training and also the overheads, both data access and computing, involved in model construction.

RQ-3: *How can we reuse or transform the knowledge gleaned from different but related models to effectively scale model training while preserving desired accuracy?* This relates to the ability to leverage *transfer learning* over the voluminous satellite datasets dispersed within a cluster.

1.3. Approach Summary

DaOur methodology encompasses the following key steps: (1) partitioning of satellite imagery to ensure co-location, data locality, and effective dispersion of workloads, (2) a rigorous exploration of generating multidimensional latent space representations (embeddings) from the images to accomplish several objectives relating to representativeness and space-efficiency, (3) leveraging these embeddings to support model constructions while ensuring effective resource utilization, and (4) incorporating support for effective transfer learning across spatial and temporal scopes to reduce systems overheads (relating to CPU, GPU, and I/O) while reducing cold-start times and ensuring faster convergence of models.

Satellite images are partitioned based on a particular geohash precision across a distributed hash table, ensuring data locality during spatiotemporal model training and transfer learning. Additionally, the geohash algorithm enables efficient load balancing and incremental scalability.

We leverage *representative learning* to extract embeddings from the data. Rather than produce an all-encompassing embedding over the entire image, we generate an ensemble of embeddings - one for each partitioned spatial scope. Besides allowing dispersion of workloads to facilitate scaling, this allows the embeddings to extract local patterns within the data.

We use deep convolutional encoder-decoder networks to generate the embeddings. The embeddings represent higher-order latent features extracted by the network. Both the embeddings and the network play a key role in our framework. For each spatial scope, we store one network. Multi-band optical satellite imagery is partitioned, passed through the corresponding trained encoder network, and once these latent-space representations are produced we use them, and not the raw imagery, for subsequent operations. This is unlike compression-based schemes that require decompressions and data sweeps.

Our methodology rigorously explores several aspects of how these embeddings are produced. These include the choice of loss functions, preservation of inter-band metrics, and the dimensionality of the embeddings. We leverage two methods to inform the dimensionality of our embeddings: image compressibility and principal component analyses (PCA).

We posit that training models directly on the embeddings has two distinct advantages. First, it reduces computational footprint by reducing duplicate pre-processing costs and generation of features from the raw data. Second, because the embeddings are compact, we expect to see reductions in memory consumption and faster completion times during training. This facilitates richer and more diverse analyses.

We construct diverse types of models using these embeddings. To assess the suitability of these models we contrast their performance with same models constructed using raw data across several dimensions. This includes the space-time efficiency of models and also the accuracy and quality of the constructed models. During inferences, the multi-band satellite imagery is passed through the encoder network portion of the network.

We also explore avenues for transfer learning, a method in which training beings with a model that has already been trained on another task to achieve faster training and lower error, in our methodology. There are two aspects to this: across different spatial regions and within the same spatial extent. We leverage the degree of spatial similarity across regions to inform how transfer learning can occur across models. The degree of similarity is determined by a cluster-based meta analysis and the dimensionality of the embeddings. Transfer learning within a spatial region allows us to initiate layers for new bands and models.

Datasets used in this study: As part of this study, our empirical benchmarks have been performed on hyperspectral imagery from MODIS and Sentinel-2. MODIS (Moderate Resolution Imaging Spectroradiometer) is an instrument aboard the Terra and Aqua satellites, both part of NASA's Earth Observation System, which takes low resolution images to track large-scale phenomena, such as weather patterns. Sentinel-2 comprises a constellation of two orbiting satellites maintained by the European Space Agency that takes much higher resolution images with the aim of monitoring land-surface conditions.

1.4. Paper Contributions

In this study we describe our methodology to leverage latent space representations to support effective storage and processing of voluminous satellite data. Specific contributions of our work include:

1) Reduction of data storage and data processing operations over voluminous satellite imagery. The embeddings are data format agnostic.
2) A rigorous exploration of how the embeddings are constructed and their suitability for transfer learning across spatial scopes.
3) Leveraging embeddings for model training in diverse settings.
4) A framework that scales with increases in data volumes, the number of models, and the number of machines.
5) Our methodology can be leveraged using diverse deep learning frameworks (for example, in this work we have leveraged PyTorch and Tensorflow). Furthermore, the methodology is agnostic of the satellite systems, projections, and the formats in which the images are sensed as evidenced by our benchmarks with MODIS and Sentinel-2 datasets that represent two of the major publicly available satellite data collections.

1.5. Paper Organization

The remainder of this paper is organized as follows. Section 2 describes related work in autoencoders, spatial imagery processing, restoring missing satellite images, and generative models. Section 3 covers the methodology, including the way in which data is partitioned and embeddings are generated. The experimental setup, storage reduction, and performance of models trained on embeddings are analyzed in Section 4. Finally, conclusions and future directions for research are outlined in Section 5.

2. Related Work

Autoencoders. Autoencoders are a well-explored application of deep neural networks and can be used for a number of different applications, including dimensionality reduction, feature extraction, data denoising, generative modeling, and pre-training [1]. They work by reducing a given input down to a small vector (the embedding), then reconstructing the input, sometimes slightly modified, from that embedding. The result is that the embedding becomes a condensed version of the input, whose values represent high level rather than low level features [2]. For example, an embedding generated by encoding a picture of cat would contain information about the color, head position, fur length, etc., rather than information about individual pixels.

There are a number of variations on autoencoders, including sparse autoencoders, frequently used for denoising [3], hierarchical autoencoders for structured data[4], graph-centric autoencoders [5], semantic autoencoders for zero-shot learning [6], and many others. However, of primary interest to us are variational autoencoders [7], which constrain the generated embeddings by forcing them to approximate samples from a Gaussian distribution. Variational autoencoders have a number of favorable properties, including reducing over-fitting [8], providing more control over the embeddings [9], enabling the generation of new images via embedding interpolation [10], and creating embeddings amenable to downstream classification and regression tasks [11]. Consequently, we choose to use variational autoencoders for generating our embeddings.

Processing Spatially Partitioned Imagery. Effective spatial partitioning of satellite imagery is necessary in a plethora of domains. Geographic Object-based Image Analysis (GEOBIA) is the most popular technique used to produce high-quality Land Cover / Land Use maps. Recent advances in GEOBIA [12], [13] frequently stress the importance of the spatial paradigm in rectifying inclusion of multi-source imagery to improve accuracy. Spatiotemporal data management systems such as Galileo [14], [15] target storage of point-based observational data, and not imagery; as such, the query semantics and retrieval schemes are tuned for point observations. Georganos et. al. [16] applies non-uniform spatial partitions to GEOBIA. Similarly, these same spatial partitioning techniques have been effectively applied to efforts in building [17], farmland [18], and irrigation [19] extraction. In our application, we draw from these previous works, developing a verbose satellite imagery spatial partitioning system enabling partitioning of satellite imagery on various spatial bounds given in diverse Spatial Reference Systems and up / down sampling of imagery to rectify differences in image resolutions from different sources.

Generating Missing Images. Several approaches have been proposed to deal with missing satellite imagery. In Das et. al. [20], the authors proposed a spatiotemporal forecasting ensemble model using previous and future high-resolution images to reconstruct and impute missing images. The model is based on DSN models, which consist of stacked perceptrons containing a single hidden layer. The proposed model further incorporates pixel intensities of neighboring pixels to extract spatiotemporal features in making predictions of a given pixel. In more recent work, an encoder-decoder approach was developed in [21], where authors propose a CNN-based model to learn feature information from multiple low-resolution images and then reconstruct these encoded low-resolution states to generate missing high-resolution images. In Kim et. al. [22], the authors use an LSTM network to forecast the amount of rainfall using four-band weather radar data. The model uses a convolution operation at the input, forget and output gates of LSTM cells to capture spatial feature changes over time. We train a similar time series prediction network comprising LSTM units *directly* on embeddings and predict embeddings for the missing timestamps.

Generative Adversarial Networks. Generating artificial data is useful for a number of applications, including creating synthetic datasets and transforming existing data. Generative Adversarial Networks (GANs) are the most popular generative neural network architecture. They are generally composed of two parts: the generator, which generates the samples, and the discriminator, which determines if the sample came from the dataset [23]. GANs are traditionally used for image generation (often called Deep Convolutional GANs (DCGAN)), but they can be used to generate nearly any kind of data. They have also become popular in the fields of civil and environmental engineering [24], where they are used on satellite images and synthetics aperture radar. However, DCGANs suffer from scalability issues, resulting in long training times and high memory usage [24]. GAN variants have been proposed to better scale with large images, such as ProGANs [25] which slowly up-sample images during training .

3. Methodology

Our methodology for model construction over satellite data accounts for both the systems and model performance implications. In particular, we:

- Partition images to facilitate workload dispersion
- Leverage encoders to extract embeddings from the data
- Explore aspects relating to tuning the embeddings
- Facilitate training models directly over the embeddings
- Leverage transfer learning to speed up model training times at different spatiotemporal scopes

3.1. Partitioning of Satellite Imagery [RQ-1]

A key component of our methodology is the partitioning of images and embeddings. Satellite data are staged within the

cluster to ensure effective utilization of systems resources during model construction and refinement.

Geohashes. Satellite images are partitioned to apportion training workloads. Our partitioning scheme leverages the geohash algorithm [26]. Geohashes deterministically partition the globe into a set of non-overlapping bounding boxes and are represented as 1-dimensional strings. The length of the geohash determines the spatial extent of the bounding boxes; the greater the length of the string the smaller the extent of the geohash bounds. We use precision length 5 geohashes representing approximately 5 km x 5 km.

Each pixel within a satellite image has a spatial extent associated with it (for example, it is 250-1000 m x 250-1000 m, depending on the band, for MODIS, while it is 10-60 m x 10-60 m for Sentinel). Satellite imagery also have accompanying metadata representing the <latitude, longitude> coordinates for the first pixel. Based on the resolution of the particular satellite imagery, we partition the satellite imagery into a set of smaller images representing the spatial extents for precision length 5 geohashes.

Data Partitioning. Effective dataset partitioning and distribution depends on a number of components. The partitioning scheme must reconcile differences between images. These may include pixel resolutions, image formats, and the number of bands (each of which may have different data types and resolutions). Distribution must result in load-balanced storage across the cluster, therefore alleviating subsequent analytical hot-spots that degrade performance. However, these objectives introduce a number of difficulties. Foremost of which, support for the immense varieties of image attributes (ie. formats, sizes, resolutions, etc). These difficulties are compounded by the intrinsic coordination challenges imposed by distributed environments.

To address these issues, hyperspectral satellite images (one image per band) are partitioned in a set of smaller images based on their geohash spatial extents. Once partitioned, each image alongside their geohash is managed using our distributed hash table (DHT). DHTs, and the consistent hashing scheme that underpin them, facilitate load balancing; more importantly, DHTs offer scalability by allowing commodity hardware to be added incrementally. DHTs leverage consistent hashing to allow pair-wise load-shedding during addition of nodes and load-aggregations during node removals. In our DHT, each node is responsible for managing a portion of the hash space. Our partitioning scheme is deterministic, decentralized, and load-balanced.

The partitioning is deterministic because imagery for a particular spatial scope are routed to, and reside on, the same machine. The apportioning scheme is decentralized because the geohash based partitioning can be computed at each node independently, and does not require centralized coordination. Finally, we are able to achieve effective load-balancing because geohashes are passed through the SHA-1 cryptographic hashing function [27] to ensure uniform distribution over the hash space. This, in turn, facilitates effective distribution of storage loads across the cluster.

Our partitioning scheme has implications for the training and refinement of models. Model training occurs for the spatial extents represented by the geohash. The images that models are trained on are smaller and allow the models to tune themselves to a particular spatial extent. More importantly, the models have data locality during training, eliminating training-related network I/O.

An added benefit of our partitioning scheme is that storing embeddings and images based on geohash groupings reduces access latencies during transfer learning. Both the images and embeddings are available for diverse model types being built for the same spatial scope. Furthermore, updates to the encoder networks to keep pace with feature space evolution over time also benefit from data locality.

Cumulatively, our partitioning and apportioning scheme reduces network I/O, ensures data locality, balances workloads, and avoids centralized coordination – all key elements in ensuring that a system scales to high data volumes.

3.2. Generating Embeddings [RQ-1]

For each precision 5 geohash there are three components: the raw images, the network, and the learned embeddings. The network architecture used in our methodology is a convolutional variational autoencoder, illustrated in Figure 1, which consists of two logical parts: the encoder and the decoder. The number of channels in the input and output vary based on whether the embedding is single or multi-band (see Section 3.2.1). The encoder contains several convolutional layers, which reduce the size of the input using a number of filters, followed by two independent fully-connected layers (also called a multilayer perceptron). The decoder is essentially an inverted encoder, starting with a fully-connected layer and followed by a series of deconvolutional layers (or convolution-transpose layers). The purpose of the encoder is to reduce a given image down to two one-dimensional vectors of fixed size, which are then used to compute the embedding, while the decoder is tasked with reconstructing the input from that embedding. Intuitively, this can be thought of as a form of non-linear data compression, reducing the size of an image and then reconstructing it with some error.

The embeddings are computed from the vectors generated by the encoder using the reparamaterization trick [28], a method that approximates sampling from a distribution in a differentiable manner. Reparameterization is necessary because, during model training, we will be minimizing both the reconstruction error of the decoder and an approximation of the Kullback-Liebler divergence (KLD) [29], that measures

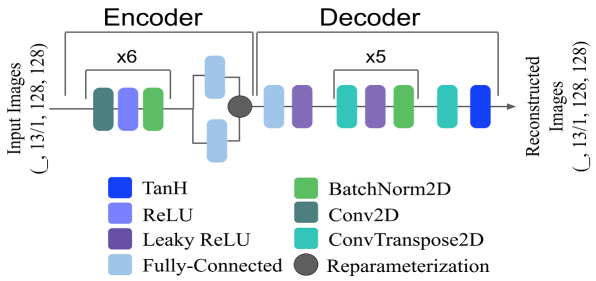

Figure 1. The architecture for the variational autoencoder. Repeated cells of convolutional layers followed by reparamaterization create the embedding. Repeated cells of conv-transpose layers reconstruct the image.

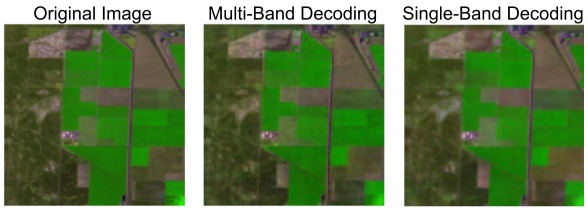

Figure 2. An example of encoding and decoding a satellite image with multi and single-band embeddings. Encoding each band individually results in a slightly blurrier image, but both are difficult to distinguish from the original.

the distance between the distribution of "samples" generated by the encoder and a Gaussian distribution. Minimizing the KLD is what differentiates variational autoencoders from regular autoencoders.

We chose to use variational autoencoders because they provide a much greater degree of control over the embeddings by mapping similar inputs to similar *regions* in the multi-dimensional embedding space. The fact that the encoder can approximate generating samples from a distribution makes the decoder *generative*: it can decode embeddings that it has not been trained on [30]. This opens a number of possibilities for future work, including manipulating embeddings to change the season or cloudiness of an image, sub-linear time identification of similar images [31], and image-based forecasting [32].

3.2.1. Multi vs. Single-Band Embeddings

Satellites monitor many frequency bands, both in the visible and non-visible spectrums. One important design decision is whether image bands should be encoded individually or collectively. The Sentinel-2 satellite monitors thirteen bands, ranging from 21-185 nanometer wavelengths. Encoding bands individually is more flexible, allowing new bands to be added to the system without constructing a new model. However, it also requires the storage of additional embeddings, one for each band rather than one for all thirteen, and, as can be seen in Figure 2, images reconstructed from individual bands show a minor loss in resolution. Additionally, as profiled in Section 3.3 and Figure 4, single-band embeddings are approximately the same size as multi-band embeddings. These differences can most likely be attributed to the fact the multi-band encoder learns something about the relationship between bands and is both better able to align the regions in the image and represent their relationships more efficiently. The single-band encoder is not aware of relationships between bands, resulting in more minor discrepancies and less efficient embeddings.

3.2.2. Metric-Aware Embeddings

Inter-band metrics are an essential aspect of any satellite image analysis, and can be used to reliably identify urban regions (NDBI), bodies of water (NDWI), and cloud cover in satellite images. The most commonly used metric is the Normalized Difference Vegetation Index (NDVI), which quantifies both the amount and the general health of the vegetation in each pixel of an image [33]. Given the number of applications that depend on these metrics, it is critical that our methodology preserves them within a reasonable degree

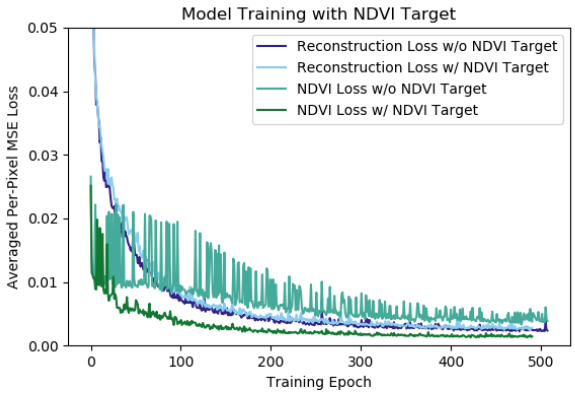

Figure 3. A comparison of reconstruction and NDVI loss during model training, with and without NDVI loss as a second target. The second target dramatically reduces NDVI loss, without impacting reconstruction loss.

of accuracy. Even if a decoded image appears indistinguishable from the original, it is possible that minor errors on each band could cumulatively alter these metrics to a significant degree, with deleterious effects for any applications that depend on them.

Figure 3 backs up this hypothesis, demonstrating that the difference between the NDVI of the original and reconstructed images is approximately twice as large as the difference between the individual bands. This is unacceptable for many applications such as crop biomass estimation [34]. To mitigate the issue, we experiment with adding the NDVI loss as a second target for our encoder. The equations for most common inter-band metrics, including NDVI, NDBI, and NDWI, are differentiable, allowing us to simply add the metric loss to the reconstruction loss and back-propagate through both. Figure 3 demonstrates performing this operation with NDVI dramatically reduces the NDVI loss without impacting the reconstruction loss. Interestingly, it appears that the encoder is able to learn and preserve relationships between bands more quickly and effectively than the values of the bands.

3.3. Fine Tuning Embeddings [RQ-1, RQ-2]

In order to optimize the storage and computational savings of an embedding-based data store it is important to make the embeddings as small as possible while remaining within a given error-bound. In the case of storage with a single encoding model, determining the minimum size is relatively straight-forward; the model can be trained repeatedly with smaller embedding sizes until the optimal size is found. However, in the case of our spatiotemporal-based storage system, this processes is markedly less efficient. It would be necessary to go through this process for each new spatial and temporal scope, tens of thousands of times. To resolve this issue, we compute and examine several heuristics to quickly choose the optimal embedding size.

To compute these heuristics, we train encoders with a variety of embedding sizes on 100 randomly sampled regions. The encoders are trained until their loss has not decreased for 50 iterations, and the best loss is recorded. We also compute three different scores for each region:

Function	Complexity	Linear PCA	Kernel PCA
Linear	0.196	0.197	0.197
Polynomial	0.402	0.402	0.418
Exponential	0.852	0.849	0.849

TABLE 1. R^2 FOR ESTIMATING MULTI-BAND EMBEDDING SIZE

Function	Complexity	Linear PCA	Kernel PCA
Linear	0.183	0.229	0.218
Polynomial	0.456	0.491	0.495
Exponential	0.869	0.869	0.858

TABLE 2. R^2 FOR ESTIMATING SINGLE-BAND EMBEDDING SIZE

Complexity: An approximation of its Kolmogorov Complexity [35], calculated as the ratio between the size of the images in that region in bytes with and without gzip compression.

Linear PCA: The number of components that explain $\geq 95\%$ of the variance in the images of that region, determined by the Principal Components Algorithm. [36]

Kernel PCA: The number of kernels required to explain $\geq 95\%$ of the variance in the images of that region, determined by the Kernel Principal Components Algorithm [37], a non-linear variation of PCA.

The algorithms chosen for computing these scores are intended to measure the informational content of each region. For example, a region comprising entirely of unchanging desert is likely to be highly regular, more easily compressed, represented by fewer components, and requiring a smaller embedding size.

We attempt to fit three different functions against each set of losses and scores. We use a linear function, a single-degree polynomial function, and an exponential function. The results for multi-band embeddings can be seen in Table-1, and single-band embeddings in Table-2. The scores shown are the coefficients of determination (also known as R^2) which represent the proportion of variance in the data explained by the model, with 1.0 being a perfect fit.

In both cases, the exponential function fits the data considerably better than either linear or polynomial functions do. Intuitively this makes sense, as the loss is unlikely to decrease when the embedding size is increased past its optimal point, while shrinking it should dramatically increase the reconstruction loss. It also appears as though the complexity score is slightly more informative than the variants on PCA. Consequently, we use the exponential function and the complexity score to estimate the optimal embedding size of each region.

The process of using this heuristic to determine the optimal embedding size for the region identified by geohash "9qdfr" is illustrated in Figure 4. First, the complexity score for the region is computed, in this case it's 0.866, then a variety of embedding sizes are passed through the heuristic, creating the plot shown. In our experiments, we observed that any reconstruction with a loss lower than 0.002 is largely indistinguishable from the original image, so we'll use that as our error bound (although Figure 4 suggests that 0.0015 is the best we can do). For "9qdfr", the first single-band embedding size to meet our error bound is 32, and

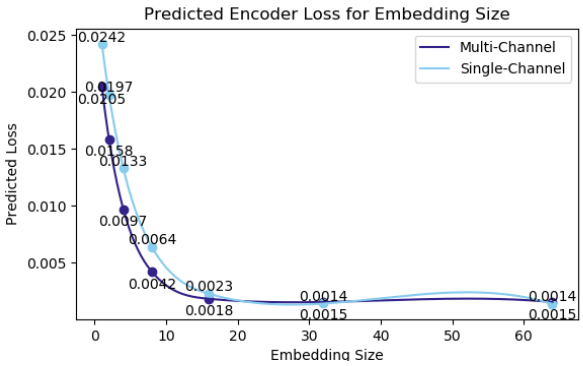

Figure 4. The predicted loss for encoders of varied embedding size for geohash 9qdfr (complexity score of 0.866), using exponential models.

the first multi-band embedding size is 16. This gives us a very quick way to estimate the ideal embedding size without training many networks for each spatiotemporal scope.

3.4. Effective Transfer Learning [RQ-2, RQ-3]

Training deep neural networks is time consuming and computationally expensive, requiring hours or even days. Any system that depends on the continuous training or re-training of many networks, especially one such as ours that requires many networks on a single machine, must be able to do so quickly and efficiently to avoid excessive resource consumption or contention. To reduce model training time we depend heavily on transfer learning.

Transfer learning is an essential part of many modern deep learning applications. Generally, it involves taking the weights of a model trained on one dataset and training it on another, in the hope that it will converge more quickly and to a more optimal minimum than if it had been trained from scratch. Our methodology makes use of transfer learning in two different ways: across spatial scopes and across temporal scopes.

3.4.1. Spatial Transfer Learning

Spatial transfer learning is guided by the similarity between spatial regions. Specifically, we leverage land classification from the National Land Cover Database [38]. It provides images encompassing the continental US which label each pixel as one of 16 unique types or as unclassified, for a total of 17 potential labels. Classification examples include Evergreen and Deciduous Forest, Cultivated Crops, and four levels of Developed Land. Our methodology posits that geohash-defined regions with similar classification proportions will have similar high-level features and may benefit significantly from transfer learning.

We chose to use the NLCD for this task because it is effectively a meta-analysis of many different data sets. NLCD's classifications are an amalgamation of a variety of sources, including satellite imagery and government allocations, and consequently it captures information at a very high level (at the loss of some resolution) making it ideal for very broad tasks such as clustering. Additionally, while the NLCD is specific to the continental U.S., similar classifications can be created for the entire planet.

We begin by computing the proportion of each classification for each geohash-defined region. This results in a normalized vector of 17 values (one for each of 16 unique labels and unclassified), where each value is between 0.0 and 1.0. We then use the k-means clustering algorithm [39] to group similar regions based on their proportion vectors. We tested multiple techniques to determine the optimal number of clusters, including canopy clustering, silhouette scores with a variety of distance functions (ie. Manhattan, euclidean, and cosine, etc), and aggregate cluster distortions. Analysis of aggregate results determines clustering should be performed with seven to nine clusters, so we chose eight.

Figure 5 illustrates the effects of training a model on a region without transfer learning, with transfer learning from a randomly chosen cluster with the same embedding size (excluding the cluster of the region), and with transfer learning from another region of the same cluster and embedding size. In each case, loss was averaged over the course of training 10 different models. It can be seen that transfer learning has a substantive impact on convergence time: starting with a model from a different cluster cuts convergence time in half, and starting with a model from the same cluster cuts the time in half again, down from 400 epochs (iterations) to approximately 50. Additionally, transfer learning appears to reduce the final loss of the model, albeit only slightly. In summation, a naive transfer learning methodology reduces the computational cost of adding new regions by 50%. Our methodology, using the NLCD dataset and k-means clustering, reduces training times by 75%.

3.4.2. Temporal Transfer Learning

Temporal transfer learning is required for two reasons: spatial regions change over time, new data may not resemble old data, and decoding old embeddings requires that the network originally used to encode them remains unchanged. Consequently, we must occasionally train another network for some regions as new data arrives, since the old network will no longer meet the error threshold. However, we can reduce the cost of generating new networks by transfer learning, under the assumption that the high level features of the region in question have not changed dramatically. To simulate this process we train an encoder on the first thousand images for a region, then keep a running average of the losses as it encodes new images. Once that running average exceeds the error threshold (0.002) we begin training two new networks, one initialized from the old network, and one initialized from scratch.

The results of this experiment are visualized in Figure 6, averaged over 10 runs. It can be seen that using transfer learning modestly reduces convergence time and increase consistency. Additionally, just like spatial transfer learning, temporal transfer learning leads to a better overall performance. While the experiment does not suggest that temporal transfer learning is effective in meaningfully reducing computational load, it does lead to improved model performance at no computational cost.

3.5. Training Models On Embeddings [RQ-2, RQ-3]

A key capability provided by data stores is data analyses. In our system it is entirely possible to decode an arbitrary subset of the data and perform any typical remote-sensing analysis on the decoded images. However, storing embeddings of high-level image features introduces some intriguing possibilities. In Xi Chen et. al. [11], it was observed that embeddings may be more amenable to classification than the original data. Additionally, the fact that embeddings are significantly smaller than the original data not only reduces both the time required to read them and their memory footprint, it also reduces the size of the models required to analyze them.

To asses the impact of performing statistical analyses directly on embeddings, we examine two popular types of deep learning models: a generative adversarial network (GAN) and an image-to-image time series prediction. Both were chosen due to their long training times, high memory requirements, and the fact that they work on images. To asses our methodology, the models are trained on raw images and on embeddings, and the error, resource consumption, and training times for each model are contrasted.

3.5.1. Generative Sampling

Generative adversarial networks are a popular deep learning application that typically require a large volume of data and a long training time. Here, we train a GAN capable of

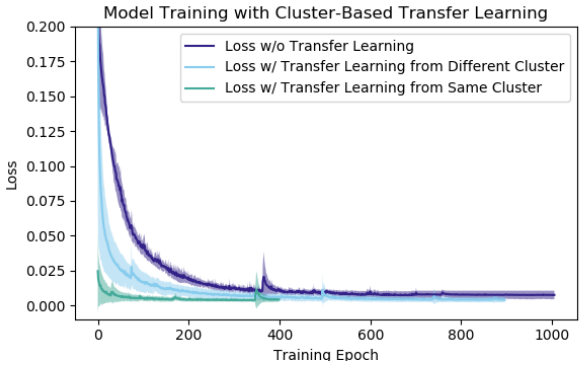

Figure 5. A comparison of average losses (shown with standard deviation) during model training with and without targeted transfer learning. Transfer learning results in much faster convergence and lower total loss.

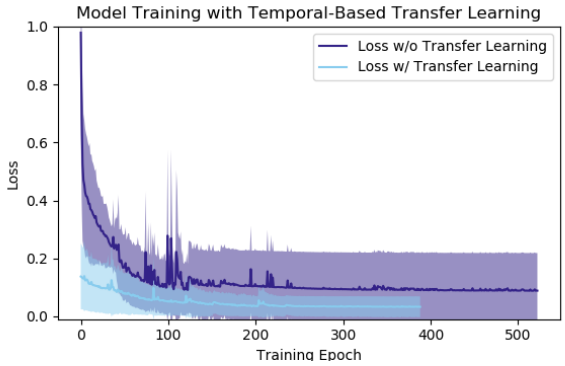

Figure 6. The average losses (shown with standard deviation) during model training with and without temporal transfer learning. Transfer learning results in more consistent and lower total loss, with reduced training time.

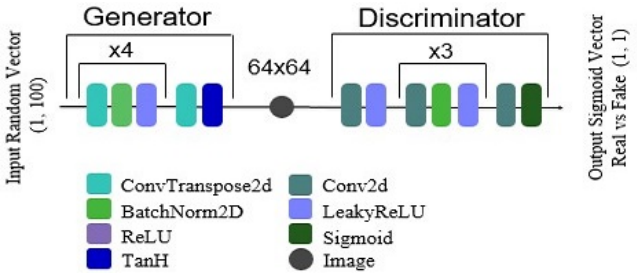

Figure 7. The architecture for the raw satellite image DCGAN

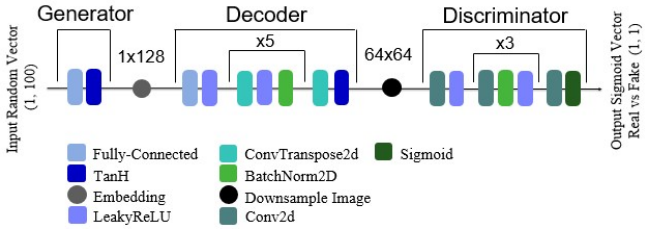

Figure 8. The architecture for the multi-band embedding GAN

generating representative images for a given geohash region on both raw images and multi-channel embeddings and show that models trained on embeddings are dramatically less resource intensive and suffer almost no increase in error.

We use two different types of GANs for this experiment. The first is a deep convolutional GAN (DCGAN) trained on raw satellite images, whose architecture can be seen in Figure 7. The second GAN is the embedding GAN which generates multi-band embeddings, visible in Figure 8. The embedding GAN comprises three separate networks: the generator, the decoder, and the discriminator. The decoder is taken directly from the encoding network (Figure 1), is pre-trained, and its weights remain constant during training. The discriminator in the embedding GAN is identical to the discriminator in the satellite image GAN.

3.5.2. Time Series Convolutional Nets

Missing satellite images are a common problem caused by clouds and cloud shadows, distortions from satellite sensors, and lack of satellite coverage. This often results in contaminated pixel values across multiple bands which hinder any further analyses of data such as weather monitoring, soil monitoring, or rainfall predictions. It is critical to gather quality sensor data with high accuracy and to be able to estimate data that are not available.

In this experiment, we trained a convolutional neural network to predict missing Sentinel-2 satellite images using the embeddings or raw-images from the previous timestamps. Sentinel-2 satellite images are captured every five days, however, due to occasional sensor malfunction or clouds, data loss is unavoidable. For instance, for a given geohash, after removing images with a cloud coverage of more than 15%, only 10% of the images remained. Our model learns the temporal changes from the latent space vector extracted by the encoder model for multiple bands at timestamp T_{i-2} and T_{i-1} to generate the encoding image for missing timestamp T_i as shown in Figure 9. The same model is trained on raw images, with only the input/output sizes changed. The training set comprised 565 geohashes with an average of 40 time sequences from months March-November of 2018.

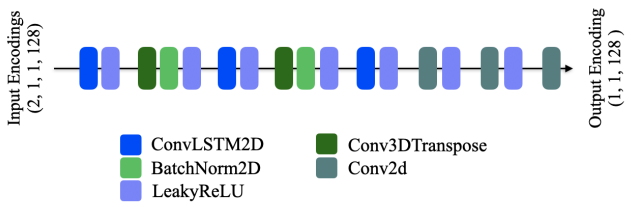

Figure 9. The Convolution Model architecture to predict missing timestamp satellite image for Sentinel-2 data using multi-band embeddings

Dataset	Format	Size	Ratio
Sentinel-2 Tiles	SAFE Zip Archive	120.88GB	2.1x
Partitioned Images	LZW Compressed GeoTIFF	250.89GB	1.0x
Single-Channel	Float Array	253.26MB	990.6x
Multi-Channel	Float Array	19.48MB	12,879.3x

TABLE 3. STORAGE SIZES OF SENTINEL-2 TILE T11SKA

4. Systems Benchmarks

To assess the suitability of our embedding-based data store it is necessary to demonstrate that it both results in meaningfully reduced storage requirements and in faster and more efficient analyses, while not significantly impacting their performance. To do so we quantify the storage space required under our various schemes and perform two types of analysis (GAN training and time-series prediction) chosen for their ubiquity and high resource requirements.

Experimental Setup. Experiments were performed on a cluster of 50 nodes (Xeon E5-2620, 64 GB Memory), each with a single Quadro P2200 GPU (5GB of memory).

4.1. Storage Reduction [RQ-1]

In this experiment we aim to profile dataset size reduction achieved using our approach. We performed this on a dataset containing Sentinel-2 tile images from the T11SKA region (a 100km x 100km region centered on Fresno, CA) ranging from 2016 - 2020. This dataset contains 433 images; which, after partitioning, produces 428k geohash bounded images.

Table 3 reports aggregate dataset sizes covering a variety of formats. Sentinel-2 tiles are the raw imagery provided by the Copernicus project. Geohash partitioned images are the GeoTiff images (with LZW compression) produced by partitioning the Sentinel-2 tiles along geohash bounds. Both the multi-channel and single-channel embeddings, the result of applying our encoder network to each partitioned image, are stored as arrays. We see that **single and multi-channel embeddings reduce dataset sizes by a factor of 990x and 12,879x, respectively, compared to raw Sentinel-2 tiles**.

4.2. GAN Training [RQ-2]

We selected 485 geohashes to run each model on, each of which generated 100 samples. All of the generated embedding samples were then decoded (using the same decoder in the embedding GAN) to create a set of satellite

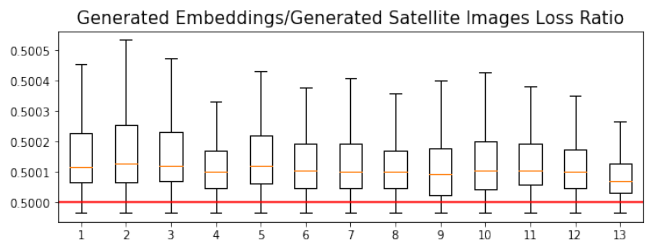

Figure 10. An illustration of the generated embeddings/images loss ratio on 485 geohashes, averaged over 100 samples and grouped by band.

Input	GPU Mem(Mb)	Memory %	Minutes
Raw Images	2113.0	41.7	5:33
Embeddings	1033.0	20.4	1:15

TABLE 4. TIME AND MEMORY CONSUMPTION DURING GAN TRAINING

images. By running both the generated images and the decoded generated embeddings through the discriminator from the image GAN we are able to get an apples-to-apples comparison between GANs. By averaging the binary cross entropy (BCE) losses over 100 samples and then dividing the average BCE loss from the decoded embeddings by the average BCE loss of the generated raw images we create a ratio where a value above 0.5 represents more raw images being classified as real. The results of this comparison can be seen in Figure 10.

We see that the quality of the generated raw images is slightly better than that of the generated embeddings. Although most (90.2%) of the geohash regions had ratios above the red line, there were some (9.8%) of the regions where the embeddings performed better than the raw images. Additionally, Figure 10 shows that the range of the ratios for all the regions only spans from 0.4999 to 0.5005. This shows that **the difference in quality between the generated images and generated embeddings is very slight.**

We are also able to compare the computational footprints from training the embedding GAN and the image GAN. This allows us to analyze the trade-off between the quality of the data and the memory-usage and training time when using embeddings compared to using raw images. The results of this comparison are shown in Table 4.

It can be seen that the multi-band embedding GAN took approximately 22.5% as long as the raw image GAN. We see a similar relationship between the amount of GPU memory used when training each of the models: the embedding GAN used approximately 49% of the GPU memory required by the raw image GAN. Although the generated raw images do produce slightly higher quality images, **the generated embeddings result in a significant decrease in the computational footprint when training GANs.**

4.3. Time Series Convolutional Nets [RQ-2]

To evaluate the performance for the time series model as described in Section 3.5.2, we trained 13 models for each band on both embeddings and raw values. In Table 5, we have reported mean absolute and mean squared errors that precisely depicts the amount of deviation of predictions from

Input	GPU Mem(Mb)	Memory %	Epochs	Minutes
Raw Images	2567.0	50.7	56.1	120.4
Multi-Band	524.0	10.3	11.5	60.1
Single-Band	515.0	10.1	24.5	56.5

TABLE 5. AVERAGE TIME AND MEMORY CONSUMPTION FOR SINGLE-BAND AND MULTI-BAND MODELS

Metrics	Multi-Band	Single-Band	Raw Images
MSE	0.0039	0.00097	0.00086
MAE	0.0473	0.01327	0.01003

TABLE 6. TESTING METRICS FOR TIME SERIES CONVNET

targets per pixel. The models trained on embeddings are at average twice as fast as those with raw images. Additionally, comparing the memory consumption of both the models, training with embeddings consumes roughly 515.0 Mb of GPU memory and is five times more efficient than training on raw images. In total, training models on embeddings significantly outperform raw images in terms of memory consumption and training times.

In Table 6 we measure the model performance on the test data. Both the models with single-band and raw inputs perform better than multi-band embeddings. The average mean-squared error on the raw image model is 0.00086, slightly lower error than compared to single-band embeddings. Using the time series model directly on embeddings results in comparable test errors, however with low computational resources. From the results, it is clear that **using our methodology, similar CNN models can not only be efficiently trained on embeddings but with considerably reduced GPU memory consumption and training times.**

5. Conclusions and Future Work

In this study we described our methodology to facilitate effective construction of deep-learning based models over satellite data collections. Our methodology is agnostic of deep learning libraries that are used to fit models, and our empirical benchmarks profile, and demonstrate, the suitability of several aspects of our methodology. In particular:

RQ-1: Our spatial partitioning of satellite imagery allows us to ensure effective distribution of workloads. Our partitioning scheme ensures co-location of imagery for particular spatial extents. This co-location facilitates creation of models that are tuned to particular spatial extents. Furthermore, co-location ensures data locality and precludes network I/O during model training. Leveraging embeddings (latent space representations) allows us to significantly reduce data storage (by a factor of 990-12,879x) and computing requirements while facilitating faster completion times.

RQ-2: To support effective model construction we leverage embeddings. Embeddings encapsulate higher-order representational features extracted from the raw data. By rigorously exploring several aspects relating to embeddings – dimensionality of the latent space, single and multiple bands, and preservation of metrics computed across diverse bands – we ensure space-efficiency, timeliness, and accuracy. We demonstrated the suitability of our methodology with two broad classes of model fitting algorithms. Our methodology reduced training times by $\sim$75% in every case, while

requiring as little as $1/5^{th}$ of the GPU memory.

RQ-3: By accounting for spatial characteristics of different geographical extents, we are able to identify spatial similarities across different regions. These spatial similarities form the basis for identifying candidates for transfer learning. Our methodology is able to significantly reduce cold start times and do so accurately, faster, and with lower consumption of resources. We demonstrated that, using spatial transfer learning, models both converged 4x faster and achieved greater performance.

As part of future work, we plan to explore support for query evaluations over embeddings to inform targeted observations and model fitting operations. A related issue that we plan to explore is visualization of embeddings.

Acknowledgments

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

References

[1] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, et al. Autoencoders. In *Machine Learning*, pages 193–208. Elsevier, 2020.

[2] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[3] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[4] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

[5] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[6] Elyor Kodirov, Tao Xiang, et al. Semantic autoencoder for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3174–3183, 2017.

[7] Matt J Kusner, Brooks Paige, and José Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, 2017.

[8] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, et al. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019.

[9] Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

[10] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.

[11] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[12] Stefan Lang, Geoffrey J Hay, et al. Geobia achievements and spatial opportunities in the era of big earth observation data. *ISPRS International Journal of Geo-Information*, 8(11):474, 2019.

[13] Nicholus Mboga, Stefanos Georganos, et al. Fully convolutional networks and geographic object-based image analysis for the classification of vhr imagery. *Remote Sensing*, 11(5):597, 2019.

[14] M. Malensek, S. Pallickara, and S. Pallickara. Analytic queries over geospatial time-series data using distributed hash tables. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1408–1422, 2016.

[15] M. Malensek, S. Pallickara, and S. Pallickara. Polygon-based query evaluation over geospatial data using distributed hash tables. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 219–226, 2013.

[16] Stefanos Georganos, Tais Grippa, Moritz Lennert, et al. Scale matters: Spatially partitioned unsupervised segmentation parameter optimization for large and heterogeneous satellite images. *Remote Sensing*, 10(9):1440, 2018.

[17] Gunho Sohn and Ian Dowman. Data fusion of high-resolution satellite imagery and lidar data for automatic building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(1):43–63, 2007.

[18] Lu Xu, Dongping Ming, Wen Zhou, Hanqing Bao, Yangyang Chen, and Xiao Ling. Farmland extraction from high spatial resolution remote sensing images based on stratified scale pre-estimation. *Remote Sensing*, 11(2):108, 2019.

[19] Hassan Bazzi, Nicolas Baghdadi, Dino Ienco, et al. Mapping irrigated areas using sentinel-1 time series in catalonia, spain. *Remote Sensing*, 11(15):1836, 2019.

[20] Monidipa Das and Soumya K Ghosh. A deep-learning-based forecasting ensemble to predict missing data for remote sensing analysis. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(12):5228–5236, 2017.

[21] Michel Deudon, Alfredo Kalaitzis, Israel Goytom, et al. Highres-net: Recursive fusion for multi-frame super-resolution of satellite imagery. *arXiv preprint arXiv:2002.06460*, 2020.

[22] Seongchan Kim, Seungkyun Hong, Minsu Joh, and Sa-kwang Song. Deeprain: Convlstm network for precipitation prediction using multichannel radar data. *arXiv preprint arXiv:1711.02316*, 2017.

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[24] Akshat Gautam, Muhammed Sit, and Ibrahim Demir. Realistic river image synthesis using deep generative adversarial networks. *arXiv preprint arXiv:2003.00826*, 2020.

[25] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2015.

[26] Gustavo Niemeyer. Geohash, 2008.

[27] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In *Annual international cryptology conference*, pages 17–36. Springer, 2005.

[28] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

[29] John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–317. IEEE, 2007.

[30] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[31] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, et al. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698, 2018.

[32] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

[33] Toby N Carlson and David A Ripley. On the relation between ndvi, fractional vegetation cover, and leaf area index. *Remote sensing of Environment*, 62(3):241–252, 1997.

[34] Jihua Meng, Xin Du, et al. Generation of high spatial and temporal resolution ndvi and its application in crop biomass estimation. *International Journal of Digital Earth*, 6(3):203–218, 2013.

[35] Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

[36] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer, 2003.

[37] Sebastian Mika, Bernhard Schölkopf, Alex J Smola, et al. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems*, pages 536–542, 1999.

[38] Collin G. Homer, Joyce A. Fry, and Christopher A. Barnes. The national land cover database. Technical report, Reston, VA, 2012.
[39] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.