DeepNet: An Extensible Data Acquisition and Curation Framework Supporting Computer Vision Deep Learning Research

---

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

TYLER NIVIN

Dr. Grant Scott, Thesis Supervisor

DECEMBER 2018

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

DeepNet: An Extensible Data Acquisition and Curation Framework
Supporting Computer Vision Deep Learning Research

presented by Tyler Nivin,

a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Grant Scott

Dr. Curt Davis

Dr. Tim Matisziw

ACKNOWLEDGMENTS

I would like to thank Dr. Scott for his continued support and mentorship through not only my graduate studies, but also through my undergraduate career. His patience, stoicism, and expertise are what have guided me on the path to where I am today. I also would like to thank Dr. Davis. He had a large part in designing DeepNets use-cases, and turning an implemented requirements spec into a project that now is being used by multiple people, for multiple scenarios. His efforts helped make the difference between a project that might not have been used again after I leave, and a project that will continue on with a life of its own. Finally, I want to acknowledge, some by name but not all, the efforts of my peers who helped implement and discussed design decisions of DeepNet with me. James Alex Hurt, Raymond Chastain, David Reese, and Alex Lasley all contributed significantly to the success of DeepNet; your contributions were vital, and appreciated. A project such as DeepNet eventually must either die, or become supported by more than the original authors; it outgrows either its usefulness, or its support base. The consistently growing and changing support base of DeepNet is what has made it into what it is today, and I'm glad to have been able to see it grow into what it has.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

We present a system, DeepNet, for ingestion, curation, and management of geospatial data images to facilitate a range of geospatial research. The system allows for the semi-autonomous ingestion of geospatial data from a variety of sources while preserving data integrity and provenance. This data repository can then be utilized in a number of ways, with our focus being on the creation of high resolution remote sensing imagery data sets (HR-RSIDS) of a variety of modalities. In addition to aggregating public data sets, the framework includes open source research data points which are found during the course of curation and quality assurance of data while utilizing the framework. These features are of particular value since there is a real possibility that features found during the utilization of the system do not exist in any previously ingested geodata source; the result being that the system grows through its utilization without the reliance on third-party geodata data sets being released publicly.

By using DeepNet, we can substantially accelerate the production of HR-RSIDS from public data that is available without imagery. While curating the imagery and data does take a non-trivial amount of time, DeepNet has been designed to support multiple-user curation via a web application component. Allowing multiple people the ability to curate the data and imagery simultaneously allows for the curation process to scale linearly with the number of people doing the curation work. Once the data have been curated, creating a new data set from the data can be done in a matter of minutes. This is especially useful because it allows the data sets created with DeepNet to be corrected, augmented, and extended with new data.

We present both the design and implementation of the framework, as well as a number of current and potential uses for the data that the framework manages.

Chapter 1

Introduction

The domain of image understanding and scene classification has been an actively researched area in the last several years. In an effort to promote the furthering of this research, there have been several open image understanding challenges[1][2][3][4]. These challenges include a number of tasks including scene classification, object detection, road segmentation, image retrieval and land-use classification. It has been seen time and again that machine learning techniques, and specifically deep convolutional neural networks, excel at these sorts of image understanding tasks.

However, one of the well-known and discussed challenges of using deep learning techniques is the need for large amounts of training data that are numerous in per-class examples and high in intra-class diversity, inter-class similarity, and image quality. In recent years, there have been several novel data sets published as benchmark data sets to measure image understanding techniques against. A brief discussion of some of the current benchmark datasets follows:

UC Merced Land Use Dataset The UC Merced Land Use Dataset [5] contains 21 classes of different types of land use. Published in 2010, it has been extensively used and has reached saturation with many current DCNN available today. It includes class such as *agricultural*, *airplane*, *baseball diamond*, *buildings*, *river*, *sparse residential*, and more. There are 100 images per class, with each image being 256x256 pixels. These are RGB images.

Aerial Image Dataset AID[6] was created for benchmarking aerial scene classification techniques, and the classes in the dataset reflect that. Published in 2016, It has a total of 30 classes with 200-400 images per class. The image size is 600x600 pixels, and is available as RGB imagery. The classes in the dataset include *airport*, *bare land*, *baseball field*, *beach*, *bridge*, *commercial*, *dense residential*, *resort*, *river*, *school* and more.

NWPU-RESISC45 NWPU-RESISC45 was published in 2017 and contains 45 scene classes. Each class has 700 images, each 256x256 pixels. The GSD of this dataset varies widely, ranging from 30m to 0.2m. NWPU-RESISC45 addresses the lack some of the problems with UC Merced in [7], and tries to address some of these identified problems, discussing the need for high intra-class diversity and inter-class similarity. NWPU-RESISC45 is a remote sensing image scene classification dataset, which inspired it's name.

IARPAs Functional Map of the World Dataset FMOW[8] is one of the most extensive imagery datasets available to-date, consisting of over 1 million images spanning 63 categories. The dataset is available as both 4-band and 8-band multi-spectral imagery, and RGB only imagery. The FMOW dataset was originally part of the FMOW challenge in 2017. With a number of image modalities and temporal views of the same features, FMOW is an extensive dataset.

PatternNet PatternNet was created as one of the first benchmark imagery datasets to be used in the context of remote sensing imagery retrieval (RSIR). [9] discusses the problems with trying to use land use / land cover (LULC) imagery datasets in the context of RSIR. The authors call out the fact that LULC imagery frequently contains more than just the primary feature of interest, which presents challenges in the context of RSIR. PatternNet contains 38 classes and 800 images per class. PatterNet is a dataset of RGB imagery.

While benchmark data sets are necessary and valuable assets in the image understanding area, their usability is limited. Each new benchmark dataset published has covered the reasons for why the preceding datasets are not sufficient in various contexts. Many times the reasons cited include lack of intra-class diversty and inter-class similarity, like discussed in [7]. Other problems were the inability to take current benchmark datasets, and apply them in new contexts, such as the motivation that led to the creation of PatternNet. Sometimes, like with UC Merced, the current techniques simply just perform too well, and there is no room for measurable improvement using a particular benchmark. All of the current benchmark datasets suffer from the inability to easily publish updated versions, that could try to address these problems. The techniques employed to create these datasets are time and labor intensive, taking tens of thousands of hours in the case of FMOW. Notably, the authors of [8] describe the system used to create the FMOW dataset in detail and the system described is actually not far from what we have done with DeepNet.

Producing imagery data sets for use with deep learning continues to be a labor and time intensive task. The collection of information, acquisition of imagery, curation of imagery and semantic data, and packaging of the data into datasets is an involved and multi-phased process. Attempting to produce quality image datasets quickly can lead to problems with quality or limit the amount of data able to be included in a timely manner. One can not simply trust public data at face value, and even the best imagery acquisition systems can not be blindly trusted due to the variety of potential problems inherent to acquiring large volumes of imagery.

The limited amount of benchmark datasets, their limited applicability to ”in-the-wild” use-cases, the time and labor involved with trying to source imagery and data publicly, and the fragmented nature of most open geodata sources are all problems that limit the development of deep learning techniques for image understanding.

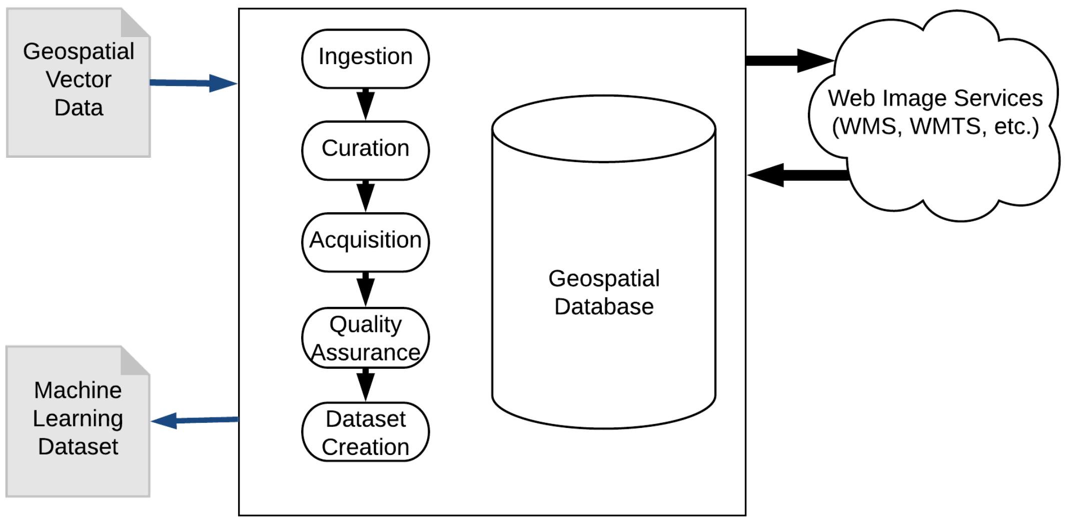


Figure 1.1: DeepNet Data Flow Overview

To this end, we have designed and implemented DeepNet: a system that aims to address some of these problems. DeepNets primary function serves to agglomerate and curate open geospatial data (OGD), pair it with HR-RSI from a number of providers and of a number of modalities, and create and publish open HR-RSI datasets. DeepNet was designed to accelerate the process of creating HR-RSI datasets: The ingestion of OGD is a semi-automated process, involving minimal developer interaction. The curation of OGD and associated imagery is done with a web user interface (WUI), against live tiling services from a number of imagery providers. The acquisition of HR-RSI is parallel and scalable, allowing for mass image acquisition. Each image and its associated data go through a secondary quality assurance process which, like the curation phase, is done via WUI and can be done by multiple concurrent users. Finally, the selection of data to include in a data set, image preparation, and dataset creation can happen in a time-scale of minutes for tens of thousands of images. Figure Fig. 1.1 contains a high-level overview of the framework pipeline.

The remainder of this thesis details the design of the DeepNet framework. Chapter 2 describes the system, discussing each of the components in Fig. 2.2. Chapter 3 dives into the design of the geodatabase behind DeepNet. Chapter 4 discusses the automated and semi-automated processes that ingest the data, acquire the imagery, and instantiate the data sets. Chapter 5 examines the web applications that enable the curation and quality assurance processes.. Chapter 6 contains one of the studies done using the DeepNet framework, as well as discusses other uses of DeepNet. Chapter 7 is a summary of the DeepNet framework, use-cases, and future plans.

Chapter 2

System Overview

2.1 Process Flow

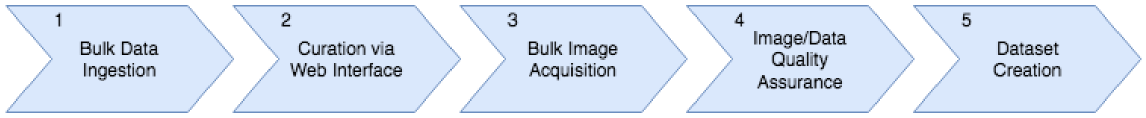


Figure 2.1: DeepNet Process Overview

Fig. 2.1 shows the path that data takes as it flows through the DeepNet framework. Beginning with bulk data ingestion, each datum in the geodatabase is then curated against one of many different live map tiling services. Once the semantic label and position have been verified / corrected, we acquire an image for this datum, and proceed to the quality assurance (QA) phase. In QA, we mainly look to verify that the semantic label and position are correct (as a second-pass check on the accuracy of our data) and also to ensure that we have successfully acquired a quality image for this data point. Once a data point has made it through QA, it is made available for inclusion in new data sets. Each of these steps will be explored more thoroughly later. Chapter 4 discusses ingestion, acquisition, and data set creation, while Chapter 5 covers the curation and QA processes.

2.2 Connected Components

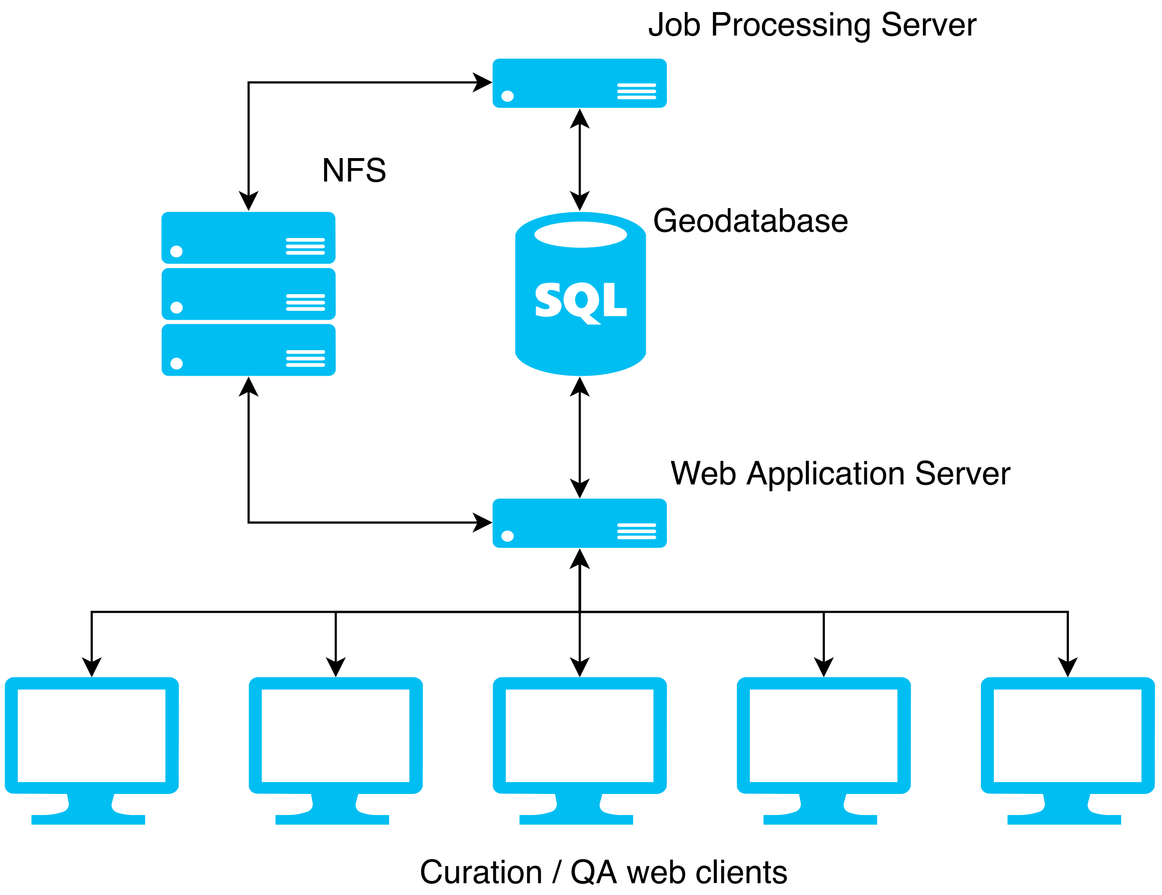


Figure 2.2: DeepNet System Overview

DeepNet has been segmented into three connected components. Doing so allows for each of the components to be developed, tested, and function without much concern for the other portions of the framework. Additionally, as with any system architecture that employs a separation of concerns approach, the resources utilized by each of the three components are independent of each other, allowing for the creation of datasets, ingestion of data, and acquisition of imagery to have no impact on the availability of resources for the web application server or the geodatabase.

Fig. 2.2 shows the system architecture for DeepNet.

2.2.1 Database

The geodatabase (GeoDB) is the central component of the framework, with both the job processing server (JPS) and web application server (WAS) interfacing with it as not only the host of the data that they operate on, but also to inform the JPS and WAS of what work they have to do.

2.2.2 Job Processing Server

The job processing server (JPS) is responsible for the execution of the various programs that enable DeepNet. Specifically, this is the ingestion, acquisition, and dataset creation applications. Each application ran on the JPS checks the GeoDB for what work it has available. When public data is ingested into DeepNet, the code is executed on the JPS and the data is written to the GeoDB. Image acquisition is driven by which records in the GeoDB have been marked as curated, signaling that they are ready to have an image associated with them. The parallel image download to NFS, and the recording of the images as successfully acquired happens on the JPS.

2.2.3 Web Application Server

The web application server (WAS) hosts the web application that enables the online curation and QA phases. The WAS communicates with the geodatabase for the data necessary to facilitate the curation and QA (e.g. users, roles, assigned work, the temporary staging of work in progress, etc.). The design of hosting curation and QA as web applications means that any number of users can simultaneously perform curation and QA with no more than access to a web browser.

Chapter 3

Database Design

When designing the schema for the database behind our framework we wanted to ensure that it was both extensible and consistent; the geospatial data that we ingested can come from anywhere in any format, so long as it has location information. Additionally, we wanted to design the framework to be able to handle imagery from multiple image providers, image layers, and image modalities. This way, if we were to gain access to other imagery sources we could rapidly bolster the imagery we have without significant changes to the database design. One example of this would be if we were to gain access to hyper-spectral imagery; with access to the new imagery we can simply create new records in the database for the location-imagery pair with minimal inherent redundancy. We also knew that the secondary data that exists across datasets was useful for driving the semantic labeling of the feature and thus we wanted to preserve it.

To meet these design constraints, we needed to define the driving datum in DeepNet. This datum is from here on referred to as a *record*. A record in DeepNet in the following way:

1. A location, represented by a geometry entry in deepnet.geom

2. A semantic label, associated with the geomtry and represented by deepnet.geom.tag_id
3. An image, which is catalogued by deepnet.record.filename, and is conceptually the marriage of a geom with an image source, at a certain point in time.

This is not to be confused with the generic term record in the context of a database.

With the two main design constraints satisfied by the chosen definition of a record, the rest of the schema design was straightforward: 1) Have the ability to associate a semantic label with a location on the earth. 2) Pair this label and location with an image from a provider and layer. 3) Facilitate the processing of the record, including adding a schema to enable the web-based curation and QA processes.

As such, the database has been segmented into two main schemas: deepnet and deepnet_web. The deepnet schema serves as the main driving schema. It holds all of the information for every record in the database, and also facilitates it's movement through the framework.

DeepNet has been designed to be as data-agnostic as is feasible. Due to this design decision there are no assumptions about the image modality, format, file size, location, or even type of imagery. While we use DeepNet in the context of HR-RSI, there is nothing inherent in the database design that requires that; DeepNet could feasibly be used to curate and create data sets for any sort of feature of interest that has location information and an image. It makes sense then, that the only assumptions made in the database are that the record has a location, a semantic label, and an image. This can be seen in Fig. 3.1.

This flexibility is what supports some of the key features of DeepNet:

1. Any arbitrary image modality can be supported.
2. Each feature can have multiple representative images, including time-series and a variety of modalities.

All of this is made possible by the main deepnet schema design. Fig. 3.1 shows the tables and their inter-relationships. The two main driving aspects of this schema are the geometries in deepnet.geom and the records in deepnet.record. Every feature of interest in DeepNet has its location and semantic label captured in the geom table. The deepnet.tags table enumerates all of the semantic labels used in the DeepNet ontology. Each geom can only have a single semantic label associated with it, removing the need for the complexity of handling multiple semantic labels. However it is true that a place on the earth could correctly have more then one semantic label associated with it. To allow for this, there is no unique constraint on the geometry column (deepnet.geom.the_geom).

In general, each geom will have one or more records associated with it. Fig. 3.2 shows the distribution of geoms that have more than one record. Recall that the definition of a record is the marriage of a geom with an image. Each record has a ”curation status” (enumerated by deepnet.curation_status_xlat) which keeps track of where in the framework curation pipeline this record is; as the record moves through each phase, the curation status will be updated.

For a practical example consider: We have ingested a data point that tells us that there is a Baseball Diamond in St. Louis, MO. We created a record for this geom with image_source 1, Digital Globe Maps API. This means that we have began the process of acquiring an image from Digital Globe Maps API service for the St. Louis Cardinals stadium. As the curation of the image is done, the records curation status is updated accordingly. The image will have a nominal GSD, filename (which is an implied location on disk), and date/time that the image was captured. Since DG Maps API does not provide image capture date/time information, this record has image_dttm set to 01-01-2000, a default value that is before the first remote sensing satellite launch. For this record, that's all there is; it will proceed through the framework and eventually be included in a dataset.

However, DeepNet allows us to have time-series imagery, and imagery from multiple providers *for the same feature.* Continuing our example of the Cardinals stadium above, say we want to have a view of the stadium over time. Digital Globe Maps API, in it's most basic usage, returns only a single image for the location requested. However, DeepNet also supports Digital Globe Cloud Services (DGCS), a different service offered by Digital Globe that allows for getting historical imagery, not just the best imagery. So we use the geom we created initially, and create a new record for each image available from DGCS. Each of these records will be processed independently from each other, allowing them to be handled differently as necessary. However, since we have already confirmed that this feature existed at this location at least during some point in time, these new records go straight to the acquisition phase. We have seen an average of 38 time-series images for each record that we manually curate. Skipping the curation phase saves days of unnecessary man power, and sacrifices nothing since we have at least one record for this feature. Now we have both an image from DG Maps API Premium Imagery layer, as well as a number of time-series images. If we wanted to add even more images from a different image source for this geom, say a synthetic aperture radar imagery provider, we would do so in the same way.

The deepnet schema also has a number of views that enable convenient analysis of the data in DeepNet. The view_record_stats view simply displays counts for each record status, allowing for a quick glimpse at the state of things. Similarly, the view_wait_stats view pulls together information from a series of tables, aiding in diagnosing what went wrong when a record fails to move forward in the processing pipeline.

The last piece worth discussion is the collection of tables named ds_ready, ds_finalized, and ds_failed. These tables exist to facilitate the data set creation process. When a dataset has been prepared successfully, and is ready to be instantiated, it will be logged in ds_ready. Once a dataset has been successfully instantiated, it moves to ds_finalized. If something goes wrong during the dataset creation process, the data set will be added to ds_failed.

The other schema, deepnet_web, has been designed to accommodate the web workflow that is used during curation and QA. It contains all of the requisite data for the web application, such as users, user access controls, work assignments, and temporary storage for work that is in progress. Fig. 3.3 is a diagram of the deepnet_web schema.

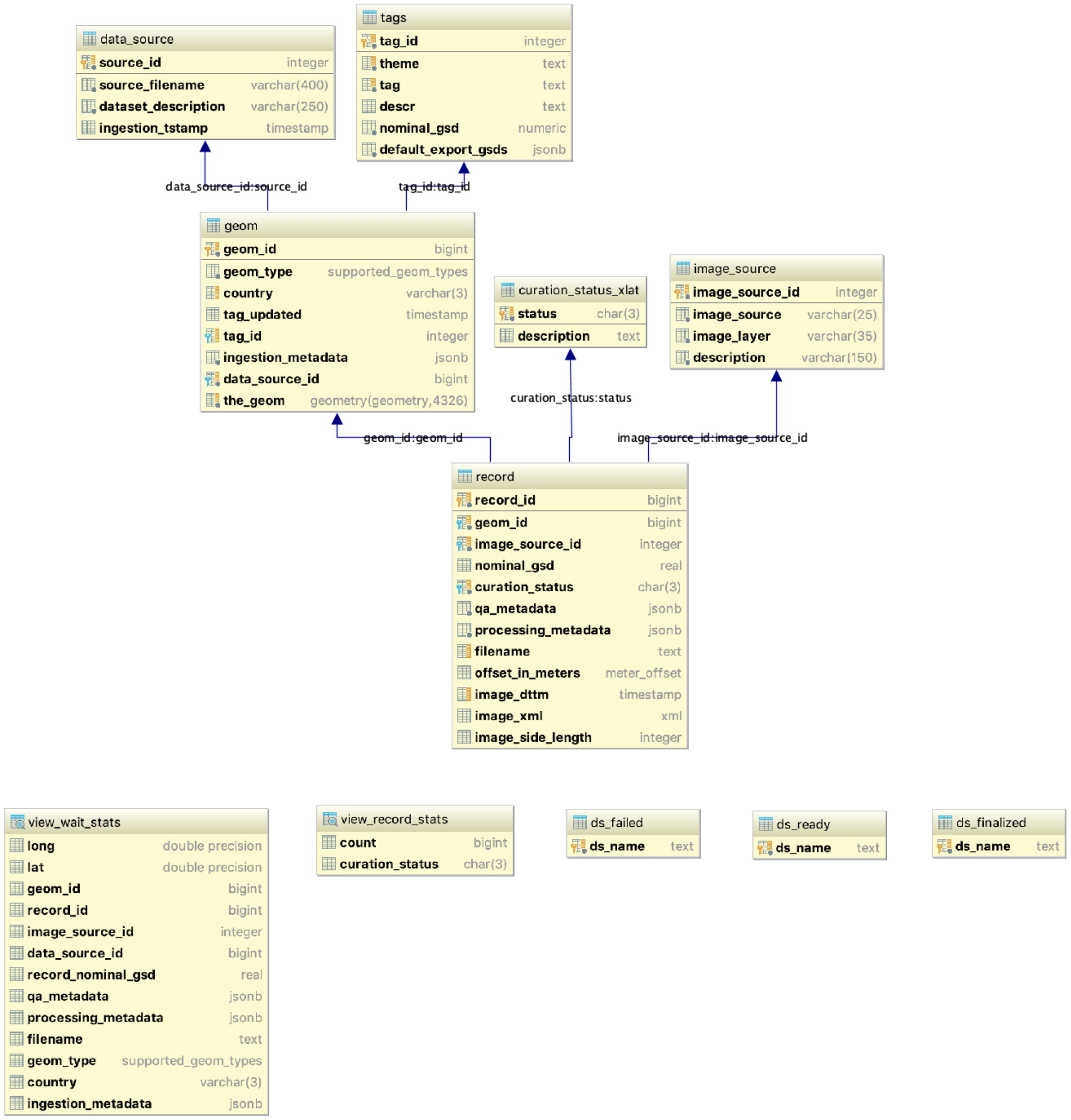


Figure 3.1: DeepNet Schema Diagram

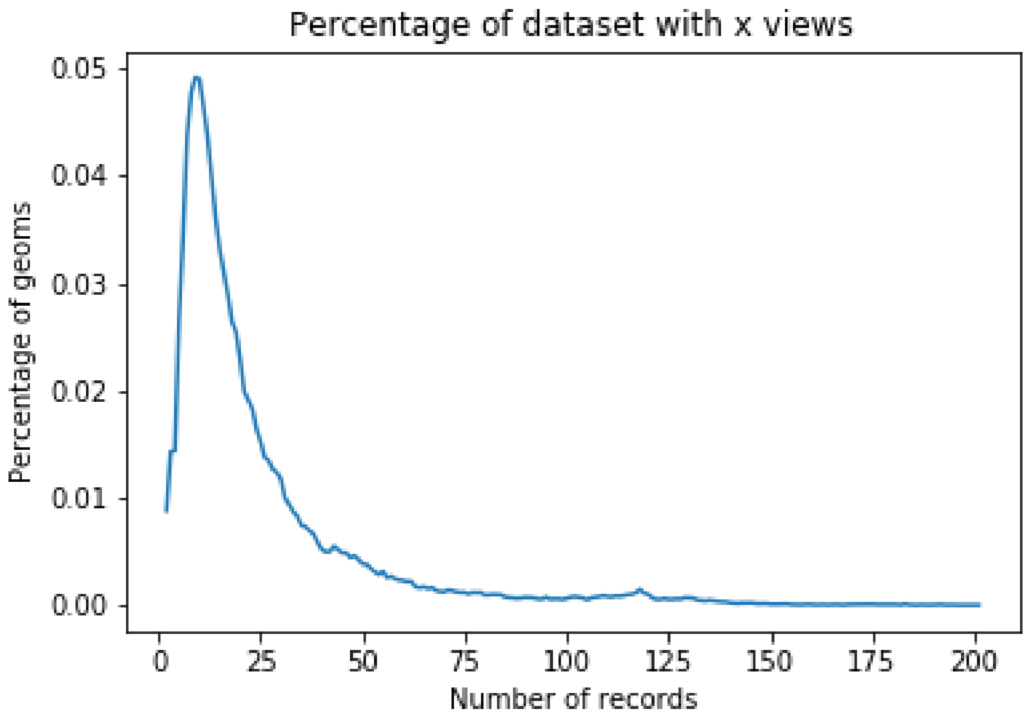


Figure 3.2: Distribution of Geoms with More Than 1 Record

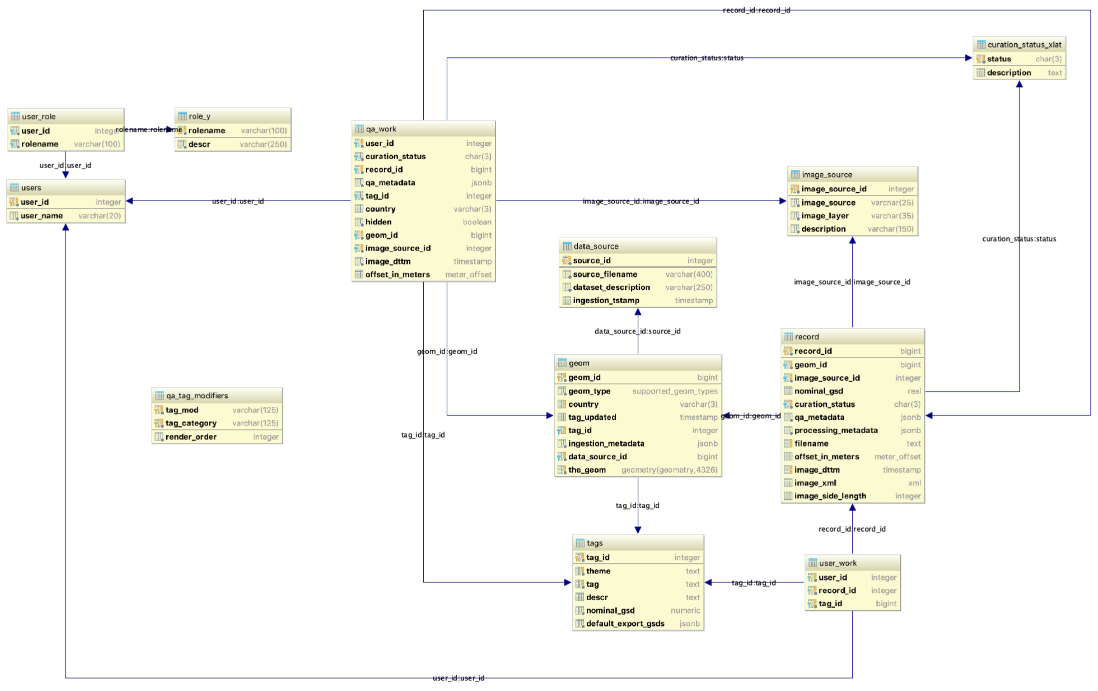


Figure 3.3: DeepNet_Web Schema Diagram

Chapter 4

Automated Processes

Fig. 2.1 shows the processing steps taken as records make their way through the framework. Steps 1 and 5 are mostly automated processes, with step 3 being completely automated. This chapter takes a look at steps 1, 3, and 5, with steps 2 and 4 being covered in the next chapter. After steps 1-4 in Fig. 2.1 have been completed, the imagery is a candidate to be included in a new imagery data set. These datasets are then used for deep learning research and, with the approval of the imagery provider, released to the public, with hope that they serve useful in the computer vision field.

4.1 Data Ingestion

The lifecycle of a record in the framework begins when the data is ingested into the geospatial database. To date, we have ingested over 80 geospatial datasets of features of interest, with plans to expand the database in the immediate future. These datasets have largely been in the *keyhole markup language* (KML) format, however a collection of them were in *comma separated value* (CSV) format. In addition to KML and CSV we have the ability to ingest nearly any format of geospatial information (web services,

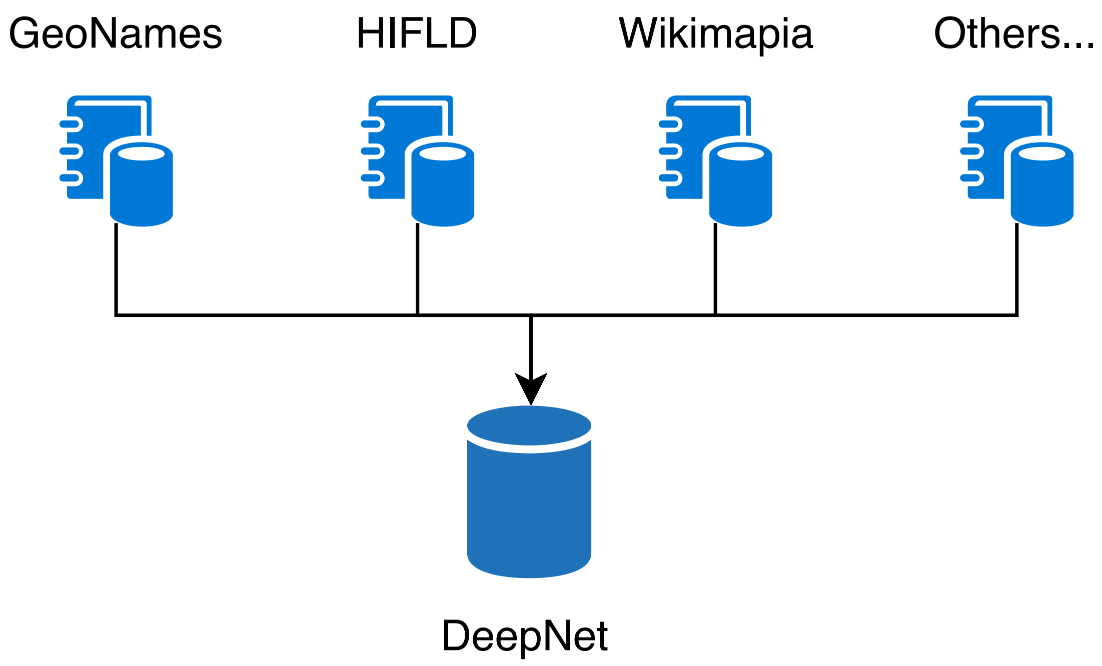


Figure 4.1: DeepNet Data Ingestion Sources

JSON, *well-known binary*, *well-known text*, etc). This is due to the modular and object oriented design of the Python ingestion script. We have defined a number of ways to ingest vector and raster geospatial data, with each implementation only needing to provide a minimum set of data. This is the same data mentioned in Chapter 3 when we discuss the definition of a record.

The main concerns when developing a parser for the framework are complete data ingestion and preserving data provenance. The focus on preserving any and all data within the original data source is part of what makes the framework so powerful. For example, the well-known GEONames [10] dataset contains over 9 million features of interest. While the associated metadata of each feature allows us to efficiently select a sub-set of these features for ingestion, preserving this data allows us to sub-set features for curation after ingestion. In this way we can bulk-load many potential features of interest, and assign them to proceed through the framework at our discretion. All of this was designed in order to maximize the amount of available information, while also allowing us to quickly identify feature sets that we wish to prioritize.

DeepNet has been designed to handle both 1) publicly available geospatial vector data (e.g. data sets of latitude/longitude pairs with a semantic description of the feature) and 2) publicly available labeled raster image data sets. Raster data (typically imagery of various geospatial features of interest) may or may not be accompanied by location information, but always has at minimum a semantic label. This type of data is ingested into the framework and immediately ready to be assigned for QA; We bypass the curation and acquisition phases since this data already has associated imagery. Vector data, conversely, *must* have location information associated with each feature label in order to be of any use. This being the case, we create a geom in deepnet.geom, and immediately create a record for this new geom, using a default imagery source. Having both a geom and a record, the ingested data is ready to be assigned for curation.

When ingesting both types of data, the semantic label used in DeepNet is either 1) the exact label provided in the original data source or 2) a mapping of the original label to a label in the DeepNet ontology.

Being able to process both types of data gives us more flexibility. By being able to ingest data that has nothing more than location and semantic label, we can acquire imagery for features that do not have public imagery associated with them. Conversely, whether it be across time, or for different image modalities, ingesting data that has imagery and its location and semantic label means we can expand upon the current public imagery data sets. When raster data does not have location information available, we decided that it is still worth-while to ingest these datasets, as we can combine current public datasets with other imagery from DeepNet, complimenting public raster data with imagery that we have aggregated.

4.2 Imagery Acquisition

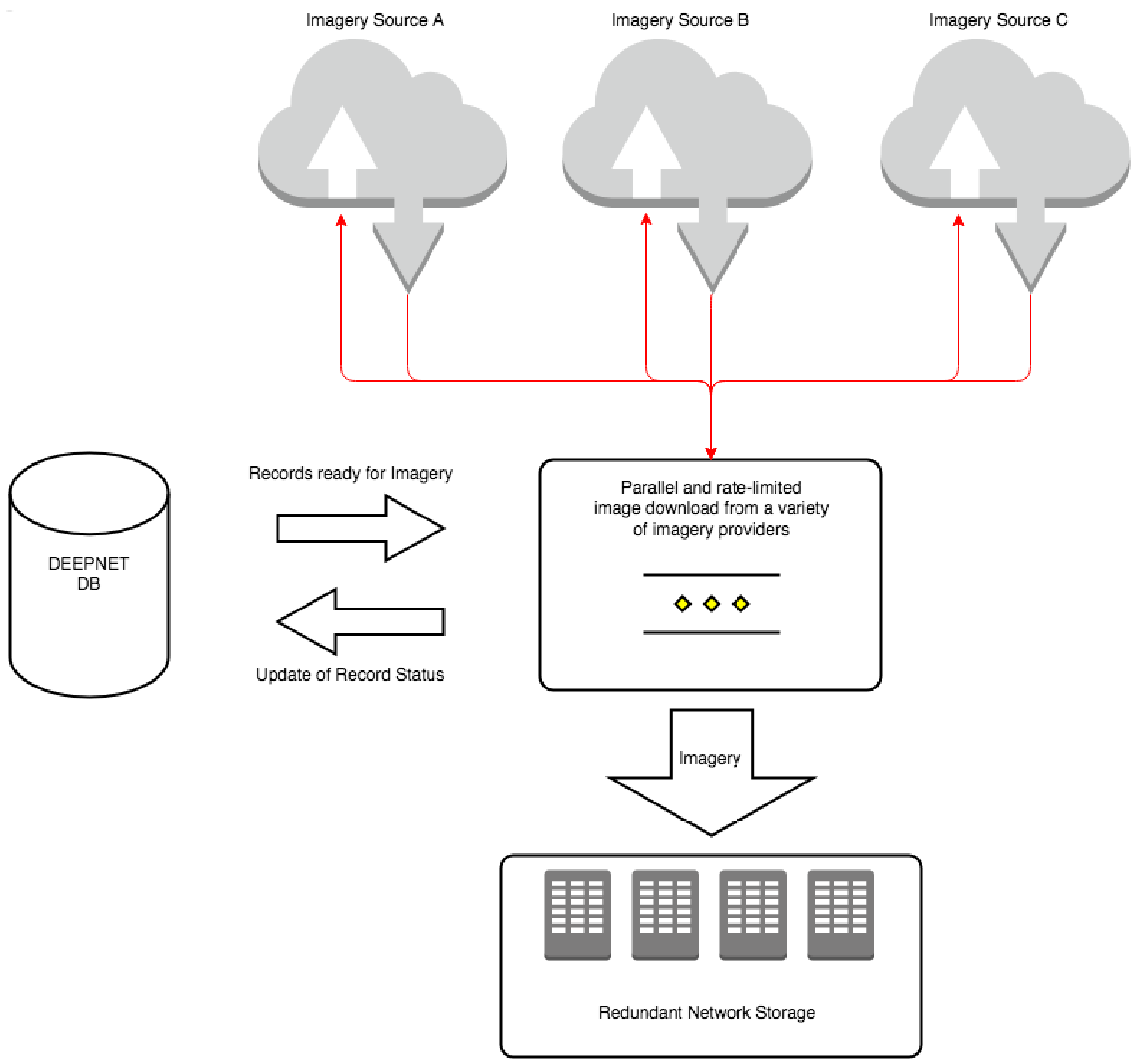

Imagery Source A
Imagery Source B
Imagery Source C
DEEPNET DB
Records ready for Imagery
Update of Record Status
Parallel and rate-limited image download from a variety of imagery providers
Imagery
Redundant Network Storage

Figure 4.2: DeepNet Imagery Acquisition Diagram

Imagery acquisition is done as a batch-processing step. Once a record has been marked as curated in the database, the image download process will attempt to acquire an image for it. The downloader reads from the record table, and dispatches a concurrent process to handle downloading of the image. If all goes well, the image is written to a network file system, and marked as downloaded in the database. However, there are a number of reasons that an image download could fail, with some being outside of our control. Network "hiccups" such as random disconnections, timeouts, or even HTTP 500 responses from the imagery provider are all things the downloader has to gracefully handle. When any of these types of problems occur, the downloader reports the failed download to the database by setting the records curation status to ”wait”, with as much contextual information about the reason for failure as it can provide. Later, these records can be examined by a DeepNet administrator, who then makes a judgment call for these records; either try to acquire them again to see if the failure was coincidental (which happens more often than one would think) or remove the record as something has gone wrong that can not be fixed. When the decision is made to do something other than simply try again, investigation is done to try to determine the source of the error.

Sometimes, the downloader fails for reasons that are due to our systems. For example, if the network file storage goes offline, the downloader will be unable to save the image. In this case, the database may still be active; if so, the downloader will mark the record as "wait" and report that it failed to save the image. This is helpful as it allows us to identify and fix the issue, and retry the "wait" records that were associated with this failure with high confidence that we should be able to acquire those previously failed records. Further, if the downloader itself should completely fail, say lose power in the middle of downloading the imagery, there is no cause for concern. We have designed the downloader so that the records in the database will not be updated until and unless the downloader finishes its task; the records will either be left untouched, completely marked as downloaded, or completely marked as failed.

4.3 Dataset Creation

The last step in the framework is to package the curated and quality assured data points into a dataset ready for publication. After a record has been quality assured (step 4 in Fig. 2.1), it is eligible to be included in a dataset. The selection of which records to include in a dataset can happen in one of two ways:

SQL selection With SQL selection, the user identifies which records to include in the dataset by specifying metadata to match the records against. For example, a user could decide to include all records with tag "Bridge", which also have qa_metadata field "small" and limit those matches to records from DG Maps API. This allows for dynamic selection of records based only on their metadata.

Record id based selection As an alternative to SQL selection, it is possible to create a dataset through a list of record ids. When the dataset is created this way, one uses a Python script that has been made for this process. The user provides the Python script with a CSV of record id and semantic label. Like with SQL selection, the semantic file for the dataset does not need any correlation with the DeepNet tag ontology. With the record id based selection, the user can use arbitrary SQL statements to select which records they want, iteratively building up the CSV file. They also can manually edit the CSV file, making sure that specific records are included or excluded.

Regardless of which method of record selection is used, dataset creation is a two-step process. First, the dataset is prepared by defining the list of semantic class labels that define the dataset. During the preparation step, the used also specifies the image size and resolution on a per-class basis. This preparation step creates a staging schema whose name is specified during dataset preparation. This schema contains a number of tables that enumerate the dataset semantic labels, their associated export resolution and image size, and the records to be included in the dataset. Additionally, metadata about the records is preserved in the schema so that it can be included in tertiary data files when the dataset is instantiated. Later, when the dataset is instantiated, the source images will be resized and cropped around the feature in order to meet the specified constraints. The last step in the dataset preparation process is the specification of which records to include in the dataset, and what their semantic label should be. If no invalid constraints have been specified, then the dataset will be added to deepnet.ds_ready, at which point the dataset can be instantiated.

4.4 Dataset Instantiation

Instantiating the dataset is done by invoking another Python script with the name of the dataset (specified during the preparation phase), and the location to write the dataset to. The dataset creator also allows for converting the image file type during the instantiaton phase, should this be desired. The dataset creator reads the information from the schema created in the dataset preparation phase. The dataset creator then does the following:

1. finds the source file for each of the records listed in the dataset
2. performs any image transformations necessary to meet the constraints specified in the dataset preparation phase
3. writes the transformed images in class-based folders in the destination
4. creates a metadata file containing metadata of each record in the dataset
5. creates a netadata file for the entire data set, containing statistics about the images in the dataset

The resulting dataset consists of two metadata files and several folders containing the potentially transformed imagery. Each folder is named with the semantic label defined during the dataset preparation phase. The dataset metadata file details the name of the dataset, the version, the classes contained in the dataset, per-class image counts, and any other data that the researchers find might be useful for the consumption of the dataset. The per-image metadata file contains nearly all metadata associated with the record, including the QA metadata, original metadata from ingestion, location information, semantic tag, theme, and original data source.

Should any of the constraints specified during the preparation phase be impossible to meet (e.g. specifying a resolution and image size that can not both be met) the dataset creator will report the error and stop. This is due to the fact that we have decided to take a prudent approach toward dataset creation; if a record was specified as to be included in the dataset, we do not want to finally fail to add it to the data set. This prevents erroneously excluding records that were intended to be included, and also calls attention to which records have caused the problem. When the dataset creator has finished successfully, the instantiated dataset is ready for use in machine learning experiments, or distribution as necessary.

Chapter 5

Web Application

5.1 Curation Process

Fig. 5.1 shows the different tasks and possibilities for a user during the curation phase. The main goals of the curation process are to confirm that:

1. The ingested location does indeed have a feature of interest.
2. Our image provider has imagery for that location that includes the feature of interest.
3. The ingested location is centered on the feature of interest .
4. The assigned semantic label is accurate and complete for the feature of interest.
5. There are no unnecessary duplicates in DeepNet.

After a feature is ingested, an initial semantic label is assigned to the feature before curation; this initial label can come from a variety of places, but generally it is based on the tertiary data in the features original data source. Pre-assigning a label this way reduces the amount of labor required during the curation step; rather

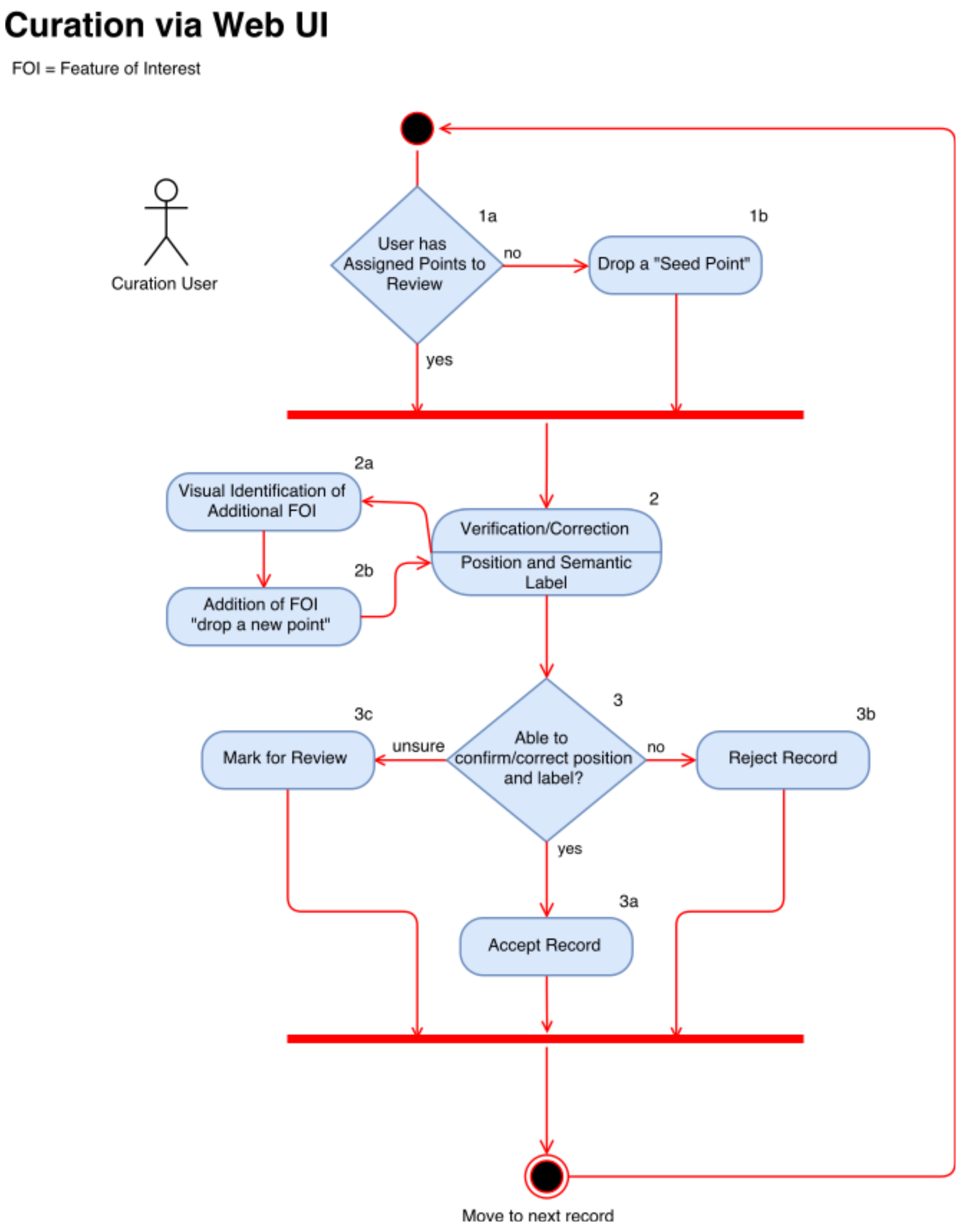

Curation via Web UI
FOI = Feature of Interest
Curation User
1a
User has Assigned Points to Review
no
1b
Drop a "Seed Point"
yes
2a
Visual Identification of Additional FOI
2
Verification/Correction
Position and Semantic Label
2b
Addition of FOI "drop a new point"
3c
Mark for Review
unsure
3
Able to confirm/correct position and label?
no
3b
Reject Record
yes
3a
Accept Record
Move to next record

Figure 5.1: DeepNet Curation Activity Diagram

than having to decide what a feature is, the curation user need only confirm that the proposed label is accurate. If it is not, they have the opportunity to correct the label.

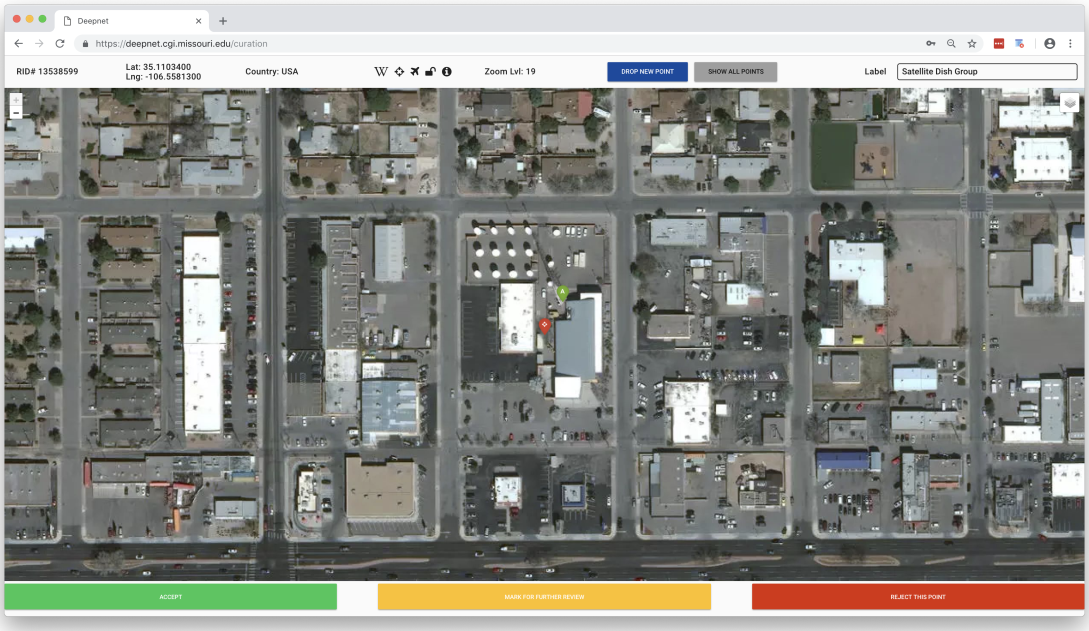


Figure 5.2: Main Curation Interface

It occasionally happens that while curating ingested features we find new features of interest that are co-located. When this happens the curator can add a new feature to the database with the correct semantic label. This is shown as path 2a in Fig. 5.1. Because we use a live tiling service we can meet all five curation criteria for the original feature and the new feature simultaneously.

Additionally, our curation interface allows a curation user to view any arbitrary place on the earth, looking for potential features of interest without the requirement of having ingested a location from a public data source. This allows us to enhance our database with more than just pre-compiled public data, seeking and collecting features of interest that may not be a part of any public dataset to-date. The same mechanisms that allow us to do this arbitrary area exploration also allow us to do area exploitation using the ingested feature locations as starting places. For example, if we are interested in adding a number of parking lots with buses in them, we could use the location of grade-shool campuses that are already in DeepNet as a starting position for finding these parking lots.

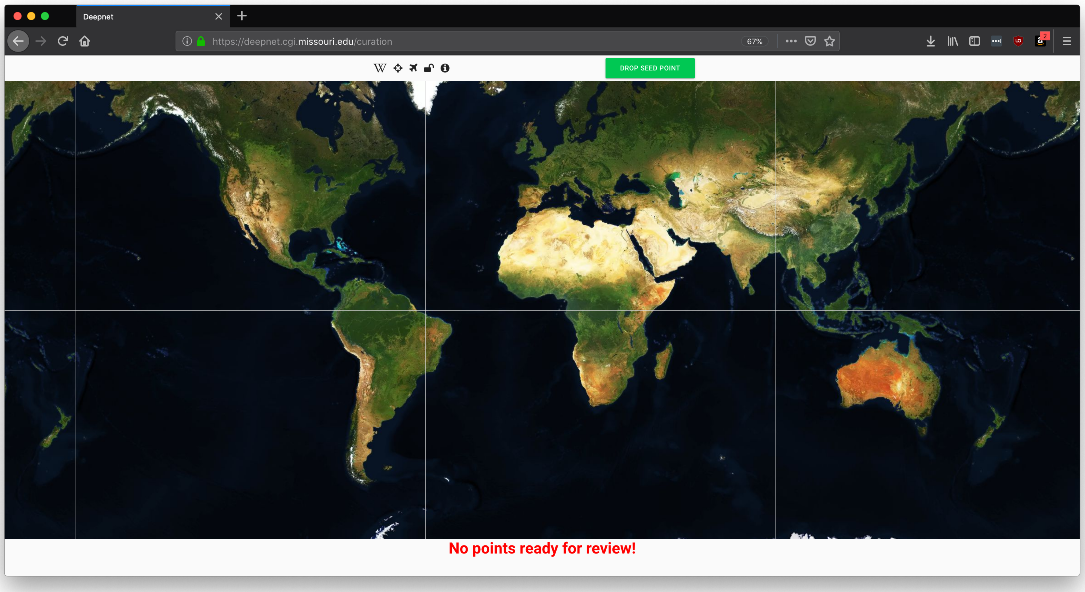

Deepnet
https://deepnet.cgi.missouri.edu/curation
DROP SEED POINT
No points ready for review!

Figure 5.3: Seed Point / No Records Curation Interface

If the curation user is unable to come to a decision about whether or not the feature is at the location in question, they have the option of marking the record for further review. This removes the record from the normal processing flow, but doesn't remove it entirely; it is set aside so that it can be assigned and reviewed by someone with more experience, or against a future map tiling service.

Figs. 5.2 and 5.3 show the web application interface during the curation phase. Fig. 5.2 shows the ability to edit the semantic label and drop a new point for a found feature of interest. Not clear from a still image is that the location of the currently under review record can be adjusted by clicking and dragging the marker to the correct position. The red marker in the image is the record that is currently under review. The green marker with the letter "A" is another record that is already in DeepNet. The ability to see other co-located DeepNet records allows the curation user to know if they should add the neighboring feature or not. Additionally, if there is doubt about whether or not the feature is in the image (blurry, cloudy, small feature, etc.) seeing other features that should be co-located can help solidify the decision.

The tiles icon in the upper right corner of the user interface allows for changing of the current map tiling service. This is useful when someone wants to view this location on the earth against another map tiling service, such as Google Earth, or OpenStreetMaps.

5.2 Quality Assurance Process

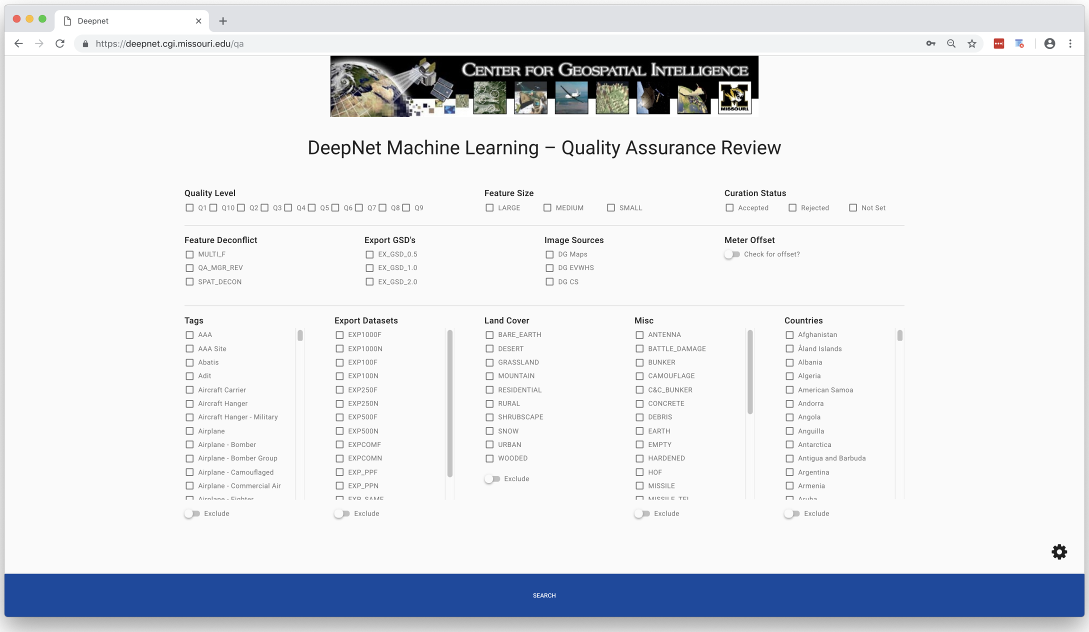
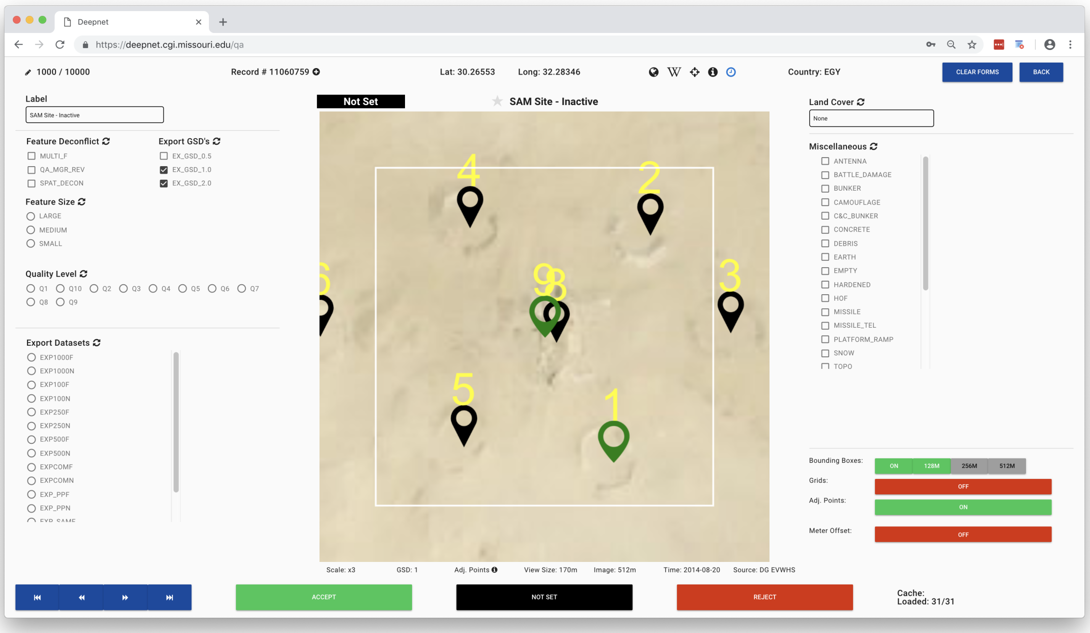


Figure 5.4: QA Search Interface

The quality assurance (QA) process that we have established allows us to do a final verification that the image we have downloaded from the imagery provider for this record (the same image that will eventually be part of a publishable dataset) is of the same quality and content as what was viewed during live curation. This is necessary because, it is feasible that we may use a different service for curation than the service we acquire the image from. For example, our time-series imagery come from a couple of different DigtalGlobe services; what is typically done is that



1000 / 10000

Record # 11060759

Lat: 30.26553 Long: 32.28346

Country: EGY

Label: SAM Site - Inactive

SAM Site - Inactive

Export GSD's: [✓ EX_GSD_1.0] [✓ EX_GSD_2.0]

Land Cover: None

Bounding Boxes: ON 128M 256M 512M

Grids: OFF

Adj. Points: ON

Meter Offset: OFF

Scale: x3 GSD: 1 Adj. Points View Size: 170m Image: 512m Time: 2014-08-20 Source: DG EVWHS

Cache:
Loaded: 31/31

Figure 5.5: Main QA Interface

we curate records against the DigitalGlobe Maps API service, check to see if time-series or other instances of the feature are available from the other services, and then acquire and QA all of these records. This "curate once, acquire multiple" approach is what has allowed for the explosion of records in DeepNet in such a relatively small time-frame.

While quality assuring an image the QA user has the ability to do a number of things:

1. Verify the image was acquired successfully, and is worth inclusion in future datasets.

2. See other feature points in our database that are physically co-located (for feature de-duplication and for confirming the semantic label).

3. Verify and correct the semantic label if necessary.

4. Assign various metadata to the record to be used for dataset production and to

better define the record.

5. Assign secondary descriptive semantic labels that could be useful but are not part of the established ontology.

Many aspects of the quality assurance process are a simple verification of the work that is done during curation. However, there are a couple of tasks that are introduced at this stage in the framework. Thevisual de-duplication ability from the curation step is replicated during QA to ensure that we both do not have semantic duplicates in the database and that we have not missed any features that we would like to capture. Additionally, the QA user can assign various other supplementary labels (e.g. camouflaged, earth, concrete, various types of land cover, etc.) that are potentially useful for both the authors in creating discrete datasets and the consumers of the published datasets. This is particularly useful with features that are both numerous and varied in appearance, allowing the potential to release disparate datasets that have the same feature class.

Figs. 5.4 and 5.5 show the QA interface. QA begins with the QA user selecting which records they want to work on during the current session. As can be seen in Fig. 5.4 the ability to select records is extensive, allowing for selection based on any data point associated with the record. If the user does not wish to sub-select from their assigned QA work, they have the option of clicking search with no options selected, which takes them to a queue of all of their assigned records.

Once the user has decided which records they want to work on during a session they are taken to the interface shown in Fig. 5.5. Fig. 5.5 shows the multitude of options available to the QA user while working on a record. Here they can add various metadata about the record, see it's current semantic label, toggle off and on the viewing of co-located records, and either accept or reject the record.

Not shown here is also the ability to simultaneously QA other records that are associated with this same location and label. This capability allows for efficient review of many records without the need to visit each one individually. This is particularly helpful when there is many time-series images associated with a feature.

If a record is accepted during QA it's metadata and status is updated in main deepnet schema, and it is removed from the assigned work for that user. Once this happens, it is ready to be included in a dataset. If the record is rejected for any reason, it is not completely removed from the framework. It is cataloged as a "soft reject" since this record did make it through curation previously, and more investigation is warranted.

Chapter 6

DeepNet Research Case Study

We have created a multitude of imagery datasets to support ongoing research projects. Many of these datasets were used to support research into the effects of sensor collection geometry on DCNN performance. The imagery and location information of surface-to-air-missile sites in DeepNet are what supported the efforts in [11]. We have created a novel high-resolution remote sensing imagery (HR-RSI) dataset of nodal road network features (NRNF). This dataset was used in a study to explore the potential of certain techniques to increase the fitness of DCNN models in the face of limited or imbalanced training data. The techniques explored were class-specific augmentation and semantic class coalescence.

Near-term plans for DeepNet include: 1) Extend and supplement the imagery in the above mentioned NRNF dataset, and release it publicly so that it might be utilized by the larger scientific community. 2) The continued publishing of new and enhanced HR-RSID to continuously produce ever-more diverse and growing data sets with which to benchmark and aid in furthering of the state-of-the-art.

What follows is the research done with the NRNF dataset mentioned previously, and the results found. The research described below was submitted to IEEE for the AIPR 2018 conference, and will be published in the near future. Look for the IEEE conference paper for the full study.

6.1 Exploring the Effects of Class-Specific Augmentation and Class Coalescence on Deep Neural Network Performance Using a Novel Road Feature Dataset

We were interested in exploring the potential of two techniques for improving the fitness of deep learning models in the context of a limited dataset. The techniques we investigated were 1) class-specific augmentation (CSA) and 2) semantic class coalescence (SCC). We utilized DeepNet in order to create the nodal road network feature dataset that was used in this experiment. The dataset we generated with DeepNet was organized into two separate datasets in order to explore the semantic class coalescence. The expanded, non-coalesced dataset contained the following 18 classes: 1) *Bridge - Large*, 2) *Bridge - Medium*, 3) *Bridge - Small*, 4) *Cul-de-Sac*, 5) *Freeway Exchange - Clover*, 6) *Freeway Exchange - Diamond*, 7) *Freeway Exit*, 8) *Road Intersection - 3WT* (Three-way intersection with orthogonal angles), 9) *Road Intersection - 3WY* (Three-way intersection with a Y shape), 10) *Road Intersection - 4W* (Four-way intersection that does not meet other category criteria), 11) *Road Intersection - 4W+* (Four-way intersection with orthogonal angles), 12) *Road Intersection - 4WX* (Four-way intersection with non-orthogonal angles), 13) *Road Intersection 5/6W* (Five and Six way intersections with various shapes), 14) *Road Overpass*, 15) *Road Overpass Group* (consisting of more than one overpass), 16) *Traffic Circle - 3W*, 17) *Traffic Circle - 4W*, and 18) *Traffic Circle - 5W+*. The second dataset included all of the same images as the first dataset, with the following classes coalesced as follows: all Traffic Circle classes were combined into one Traffic Circle class, the Road Intersection classes were coalesced to 3W, 4W, and 5/6W, and the two Road Overpass classes were combined into "Road Overpass". When the dataset was created, we used all of the NRNF records that we had in DeepNet at the time. At that time, we did not yet have time-series imagery, so the datasets used in this research study did not contain temporal views of the features. We trained 12 models: using both ResNet50[12] and Xception[13], we ran three experiments per network, per dataset. The three experiments were a set of baseline augmentations, a more aggressive rotation augmentation, and our set of class-specific augmentations.

To evaluate the performance of the models, we used five-fold cross validation and measured the weighted f1-scores of the models. We also computed the McNemar statistic for measuring the statistical relevance of the model performance deltas. That is to say, per each dataset, we tested if the models performance baseline, maximal rotation, and class-specific augmentation experiments were statistically different from each other, and not just due to noise in the model training.

When considering the performance of the models, there were a few key takeaways: For the expanded dataset, for both Xception based models and ResNet50 based models, the models trained with CSA had a statistically significant positive delta from the baseline and maximum rotation experiments. Also of note, the model trained with CSA and based on ResNet50 did *not* perform statistically different from the models based on Xception. This is not what one would expect, since Xception has depth-wise separable convolutional layers which generally lead to Xception based models out-performing ResNet50 based models. For the coalesced dataset, the Xception models outperformed the ResNet50 models as one would expect, and the CSA based models did *not* significantly outperform the baseline models. This was interpreted as the CSA used for these experiments becomes ineffective after coalescing the classes, which changes the decision surfaces and also creates class with more depth and diversity of examples.

Chapter 7

Summary and concluding remarks

We created a system for ingesting, curating, acquiring, quality assuring, and packing imagery into datasets to be used to support computer vision research. We have used this system in our own research using HRRS imagery for a number of published and unpublished works. The system we created, DeepNet, not only can provide HRRS imagery, but is capable of providing imagery of a number of different modalities, as well as multiple views of a feature of interest over time. DeepNet currently contains over 4.5 million records, generated by ingestion of both raster and vector geospatial data, as well as features of interest found while utilizing the framework.

Further, we evaluated CSA as a technique to help bolster training datasets that were lacking in some classes but not others. Using DeepNet, we created a dataset with a number of classes that were strongly imbalanced, lacking in number of available examples, and semantically coalescable. CSA yielded small, but statistically significant improvements compared to our baseline experiments. It is feasible that more aggressive or alternative CSA methods would have produced better results. We also explored the effects of SCC, and the potential trade-offs it presents. SCC resulted in improvements in the five-fold cross-validation results for the coalesced classes. Improvements in weighted F1-scores for the coalesced classes averaged 0.17.

DeepNet is an extensible framework that can change the way imagery data sets for deep learning are currently created. We do not know of any other system like it that stands to be capable of producing datasets with the breadth and depth that is capable with DeepNet. What is also unique about DeepNet is the partnerships with imagery providers that allow us to acquire imagery for a feature from a number of sources. Other benchmark datasets to date typically publish imagery from a single source, and all imagery in the dataset will be homogeneous. DeepNet has the potential to combine imagery from a multitude of image layers and modalities to provide researchers with more views of a feature than has been seen to date. With 4.5 million features currently in the framework under or past quality assurance, DeepNet has a lot to offer. Additionally, its design will allow it to continue to grow and adapt as new imagery modalities and sources become available.

Bibliography

[1] *DEEPGLOBE - CVPR18*. `http://deepglobe.org/`.

[2] *Large-Scale Scene Understanding Challenge*. `http://lsun.cs.princeton.edu/`.

[3] *Top Coder RoadDetector Challenge*. `https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=17036&pm=14735`.

[4] *SpaceNet Challenge: Road Extraction and Routing*. `https://spacenetchallenge.github.io/Challenges/Challenge-3.html`.

[5] Y. Yang and S. Newsam. "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification". In: *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*. 2010, pp. 270–279.

[6] G. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, and L. Zhang. "AID: A Benchmark Dataset for Performance Evaluation of Aerial Scene Classification". In: *CoRR* abs/1608.05167 (2016).

[7] G. Cheng, J. Han, and X. Lu. "Remote Sensing Image Scene Classification: Benchmark and State of the Art". In: *Proceedings of the IEEE* (2017).

[8] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee. "Functional Map of the World". In: *arXiv:1711.07846 [cs]* (Nov. 2017). arXiv: 1711.07846. URL: `http://arxiv.org/abs/1711.07846` (visited on 10/31/2018).

[9] W. Zhou, S. Newsam, C. Li, and Z. Shao. “PatternNet: A Benchmark Dataset for Performance Evaluation of Remote Sensing Image Retrieval”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 145 (2018). URL: `https://doi.org/10.1016/j.isprsjprs.2018.01.004`.

[10] *GeoNames*. `https://www.geonames.org`.

[11] R. A. Marcum, C. H. Davis, G. J. Scott, and T. W. Nivin. “Rapid broad area search and detection of Chinese surface-to-air missile sites using deep convolutional neural networks”. In: *Journal of Applied Remote Sensing* 11 (2017). URL: `https://doi.org/10.1117/1.JRS.11.042614`.

[12] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[13] F. Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807.