Accepted for publication in IEEE-JSTARS, 2019

Mining Concise Datasets for Testing Satellite-Data Based Land-Cover Classifiers Meant for Large Geographic Areas

Tommy Chang, Avinash C. Kak

***Abstract*—Obtaining an accurate estimate of a land-cover classifier's performance over a wide geographic area is a challenging problem due to the need to generate the ground truth that represents the entire area which may be thousands of square kilometers in size. The current best approach for solving this problem constructs a test set by drawing samples randomly from the entire area — with a human supplying the true label for each such sample — with the hope that the labeled data thus collected captures statistically all of the data diversity in the area. A major shortcoming of this approach is that, in an interactive session, it is difficult for a human to ensure that the information provided by the next data sample chosen by the random sampler is non-redundant with respect to the data already collected. In order to reduce the annotation burden caused by this uncertainty, it makes sense to remove any redundancies from the entire dataset before presenting its samples to the human for annotation. This paper presents a framework that uses a combination of clustering and compression to create a concise-set representation of the land-cover data for a large geographic area. Whereas clustering is achieved by applying Locality Sensitive Hashing (LSH) to the data elements, compression is achieved by choosing a single data element to represent a cluster. This framework reduces the annotation burden on the human and makes it more likely that the human would persevere during the annotation stage. We validate our framework experimentally by comparing it with the traditional random sampling approach using WorldView2 satellite imagery.**

***Index Terms*—land-cover classifiers, performance evaluation, satellite data representation, ground-truth annotation, big data.**

I. Introduction

Constructing training and test sets for land-cover classifiers that are effective over large geographic areas — areas that may be as large as tens of thousands of square kilometers — places a large burden on the human annotators for supplying the ground truth. The most commonly used approach for creating the datasets in such cases consists of drawing samples randomly in a uniform manner from the entire geographic region. More sophisticated approaches use a random sampler based on the Metropolis-Hastings algorithm [1]. In this paper, we compare our proposed concise-set representation method with the traditional random sampling approach for testing land-cover classifiers.

What is significant is that even with the best random samplers, the datasets that are generated tend to be highly redundant. As to what we mean by redundancy and diversity in the dataset, we illustrate these two concepts with a challenge often encountered in creating a labeled dataset from a large unlabeled dataset. Consider a human-computer interaction involved in creating a labeled training/test set. As each new sample is shown to a human annotator for its true label, it is virtually impossible for the human to remember all of the previously seen samples in order to determine whether the new sample is merely redundant vis-a-vis all the samples collected previously, or whether it really adds additional diversity to the data already collected. Note that this challenge is exacerbated by the fact that the human-computer interaction may last a long time.

So if it is impossible to avoid redundancy in the datasets, the reader may ask if that is really such a bad thing. Most datasets that are out there for the training and testing of machine learning algorithms carry no guarantee of being non-redundant. What is most important for a dataset is whether or not it captures all of the diversity in the data as it exists in the real world. As long as this diversity constraint is satisfied, the only price to pay for any redundancy in the data is that it may take longer to train and test a classifier — but the classifier performance would not be impacted by the redundancies.

Unfortunately, the consequences of redundancies in the datasets collected from large geographic regions tend to be not so benign. In light of the challenges created by the prolonged human-computer interaction, it is difficult to guarantee that a given redundant dataset would adequately capture the diversity associated with the different classes. And when it is practically impossible to assume that a dataset adequately captures all of the diversity associated with the different classes, any redundancies in the data may result in erroneous bounds on the performance of the classifiers when such data is used for testing them. We illustrate this effect with the following simple example: Suppose we are creating a test set for a binary classifier and that 80% of the data samples collected happen to fall in a small neighborhood of the same point in the feature space. With such a dataset, regardless of the actual performance of the classifier on a "true" dataset, the computed classifier performance would be controlled by the two numbers, 20% and 80%. The classifier would have no choice but to give the same class label to the 80% of the data. If the class label chosen was correct, the computed performance of the classifier could exceed 80% depending on how the classifier performs on the rest of the data. On the other hand, if the label given to the 80% was incorrect, the computed performance of the classifier could be less than 20%,

The authors are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907 USA (e-mails: chang177@purdue.edu; kak@purdue.edu).

again depending on how the classifier performs on the rest of the data. Therefore, it is particularly important to eliminate redundancies from the datasets created from large geographic areas for the purpose of designing and testing land-cover classifiers.

A major focus of this paper is on how to sample a large-scale scene to create a test set from the perspective of minimizing the burden on the human annotator. Although hierarchical sampling approaches such as stratified sampling and multistage cluster sampling [2] are often used when sampling from a large population, these methods may still end up with large samples. In addition, multiple trials are needed to compute the statistics. Therefor, these methods are not practical for the purpose of creating a single concise test set that is diverse and small. Active learning [3], on the other hand, are also commonly used to select informative data samples from a large population/dataset. Active learning is an online sampling approach and needs real-time human-machine interaction to be practical. Because its stopping criteria is often based on the number of samples collected or the amount of time elapsed, active learning may be "incomplete" in the sense that it may not get to process the entire dataset.

Generating a concise representation from potentially hundreds of satellite images covering a large geographical region is made extremely challenging by two reasons: The first is, of course, the sheer volume of the data. The second equally important reason has to do with the fact that a pixel cannot be shown in isolation to a human annotator for eliciting its class label for creating the ground truth. It is now well known that for reliable annotation, humans require both the pixel itself and its immediate surround — which we refer to as its background context. Therefore, any automated algorithm for creating a concise representation for human interaction, must compare the pixels both on the basis of the spectral signatures at the pixels themselves and on the basis of whatever it takes to represent the background contexts for the pixels.

A reader might ask: Why is it not sufficient for concise representations to be created from just the pixels themselves, without the need to also factor in the background context for each pixel? Even for a computer algorithm, pixels considered in isolation can result in their being considered similar when in fact they are highly dissimilar. For example, the multispectral signature for a pixel from a concrete road would be very similar to the signature from any number of other structures on the ground — building rooftops, water towers, bridges, etc. Creating a concise representation from just the pixels, without also including a portion of the background for each pixel, would only create a frustrating experience for the human annotators.

When you include the additional high-dimensional background context for each pixel, you end up having to deal with voluminous amounts of high-dimensional data. The sheer size of the dataset, which consists of all pixels, and its high-dimensionality mean that we are dealing with what is loosely referred to as a big-data problem. When it comes to data clustering, such problems do not allow for exhaustive pairwise comparison of the data elements for the purpose of establishing data similarity. And, since the data dimensionality can still be high even after applying dimensional reduction, such problems also do not lend themselves to the use of spatial search techniques such KD-trees, SR-trees, and cover trees [4]–[8] because their time or space complexity degrades exponentially with data dimensionality.

In addition to the issues created by the size of the data and its dimensionality, we must also cope with the fact that comparing pixels on the basis of their spectral signatures and on the basis of their background context are two semantically different actions. That is, it would make no sense to lump both the background and the foreground for each pixel into a single vector representation for creating a concise representation.

Yet another source of complexity arises from the fact that similarity constraints for clustering data are generally not transitive. To explain this point, a data element A can be similar to another data element B because the magnitude of the difference between their attribute vectors is below some threshold. And, the data element B may be similar to another data element C for the same reason. Yet, A may not be similar to C. That is, if we were to directly measure the magnitude of the difference between the attribute vectors for A and C, it may exceed the threshold being used for establishing similarity. Many clustering algorithms get around this problem by assuming that the number of clusters, k, into which the data must be partitioned is known a priori. Subsequently, a clustering algorithm must find the optimum partitioning of the data so that the average distance of every data element from the center of the cluster to which the data element is assigned is minimized. Deterministic variants of this approach lead to the k-means and other such algorithms. And the probabilistic variants of the same basic idea result in expectation-minimization sorts of algorithms. One can loosen the need for the a-priori knowledge k by testing for different k until some overall quality metric is satisfied. Unfortunately, big-data problems do not lend themselves to such experimentation. In the absence of such logic, a blind application of similarity checking, no matter how it is actually enforced, is highly likely to result in all of the image data elements extracted from all the satellite images to form a single similarity neighborhood.

This paper presents a solution to all of the issues we have outlined above for reducing a large volume of satellite data to relatively small number of similarity neighborhoods in the underlying feature space and representing each such neighborhood by an exemplar data element that the human is asked to annotate. Subsequently, all of the data elements within any given similarity neighborhood acquire the annotation of its exemplar.

With a data abstraction we refer to as an image patch, we represent each pixel by its foreground spectral signature and a high-dimensional vector that captures its background context. Subsequently, we first reduce the data dimensionality of the background context and then use Locality Sensitive Hashing [9] to compare the pixels on the basis of just the background characterizations. That is followed by refining the clusters obtained with foreground comparisons based on the spectral signatures at the pixels themselves. However, before we carry out the foreground comparisons, we get around the difficulties created by the non-transitivity of the background

similarity constraint by associating with each patch a *similarity neighborhood*, which is the set all other patches that directly satisfy the similarity condition with respect to the former patch. These similarity neighborhoods are converted into what we call a *similarity graph*. A concise set is derived from the similarity graph after the enforcement of the foreground similarity constraint.

What is particularly novel about the solution we propose is the combination of ideas we have used to address the big-data challenges in multi-satellite-image land-cover classification performance evaluation. With regard to the big-data challenges, our work provides answers to the following three important questions: (1) How does one handle all of the data variability in a big data setting? We address this question in Section IV-E where we show how LSH can be used to create a concise-set representation of all of the data in a given set of satellite images. (2) How does one cope with the difficult logistical issues related to the generation of the ground truth when you have hundreds of gigabytes of data to contend with? We address this question in Section VI where we show how to use our concise-set representation to remove redundancies from a large dataset so that what the human must annotate is a relatively small dataset. And (3) What sort of similarity grouping logic can one use when it is necessary to simultaneously use more then one similarity measure to compare image patches? We address this question in Section IV-C where we show how to apply two disparate similarity constraints conjunctively to solve this problem.

In the rest of this paper, we start with Section II by presenting a brief review of the relevant literature. Our main contribution is presented in Section IV that starts by defining the notation of a concise-set representation of the satellite image data and subsequently describes the various stages for its construction. The end of that section illustrates how these stages are put together to create a complete workflow. Section V describes how the algorithm parameters are set and shows their values for the WorldView2 imagery. Subsequently, we validate the effectiveness of our concise representation in Section VI by comparing its performance with the traditional random sampling approach. Finally, we conclude in Section VIII.

II. Related Work

In addition to the straightforward Simple Random Sampling (SRS) approach, researchers have also proposed "Stratified Sampling" and "Systematic Sampling" for creating representing datasets from large populations. The systematic sampling approach first sorts the population into a list and then subsamples from this sorted list. On the other hand, the stratified sampling approach divides the population into homogeneous subgroups and then samples within each subgroup using either SRS or systematic sampling [2]. The authors of [10] have proposed a complex 3-level stratified sampling approach and make use of various prior knowledge about the dataset. In their work, the first level of stratified sampling organizes the mapped areas by their meta-data such as the mapping method, remote sensing source, resolution, acquisition date, etc. The second level uses prior knowledge about the content of the mapped area (eg., road, building, green areas, etc). And the third level uses finer features present in the mapped areas such as the location of the individual object/inspection unit. On the other hand, [11] proposes a simpler two-stage cluster sampling approach based on the classification map. The approach we present here can be considered to be a combination of the basic notions in Stratified Sampling and Systematic Sampling.

With regard to land-cover classification, early classifiers that could be employed over large geographic areas use only low and medium spatial resolution imagery ranging from 8km to 15m per pixel [12]–[14]. Due to the low spatial resolution, the datasets involved do not create a big-data problem for these early works. However, during the last ten years, spatial resolution in satellite images has improved rapidly. It is now common to find satellite data at a very high resolution (VHR) of 0.5m per pixel or better. Fortunately, during the same time period, computer processors, memory, and storage all have become faster and cheaper. Today, it is not uncommon any longer for research labs to work on land-cover classifiers involving large datasets [15] and VHR satellite images [16].

III. The Proposed New Workflow

Fig. 1 presents the new workflow for creating a concise-set representation from multi-image satellite data. Each of the steps of the workflow described below is explained in detail in the various subsections of the next section.

1) Automatically collect all the image patches in the given collection of satellite images. (See Section IV-A)
2) Extract both background and foreground features for all image patches. (See Section IV-B)
3) Using FastMap, apply dimensional reduction to the background histograms. (See Section IV-D)
4) Using LSH, create background similarity neighborhoods for the entire population. (See Section IV-E)
5) Create the similarity graph by refining the background similarity neighborhoods with the foreground similarity constraint. (See Section IV-E)
6) Find dominant clusters within the refined similarity graph and create the concise-set representation using Algorithm 1.
7) Annotate the cluster representatives with the ground-truth labels (See Section IV-H).
8) Calculate the ground-truth consistency using Eq. 3.

The goal of this workflow is to produce a maximally non-redundant dataset for evaluating land-cover classifiers. The greater the elimination of redundant image patches, the less the burden on the human annotator.

IV. Generating a Concise Representation

A. Representation of the Population: Content, Unit, and Size

The first issue to resolve when eliciting ground-truth annotations from a human is the content of each "unit" of the data that is shown to the human. Even though the end-goal of annotation elicitation is to collect the class labels for a collection of pixels, a human observer is often not able to make a judgment about the class label of an individual pixel

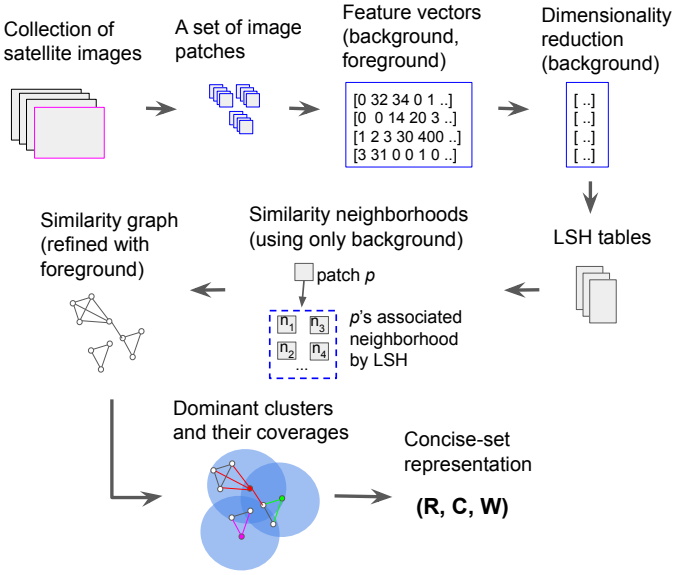

Fig. 1. Workflow for generating the concise-set representation from a given collection of satellite images that cover a large geographic area.

if it is shown as a single piece of data without any neighboring pixels. Experience has taught us that it is best to show a pixel along with its immediate surround, which we will refer to as a *patch*, in order for a human to figure out what the ground-truth label should be at the center of the patch. The human would be asked to label only the central pixel in a patch, with all the surrounding pixels merely providing a geographic context for the pixel at the center.

B. Measuring Similarity Between Satellite Image Patches

To determine whether two image patches are similar, we first represent each patch with two different feature vectors, one for the color histogram that represents the background pixels and the other for the spectral values at the foreground pixel. For a patch to be considered similar to another patch, the similarity criterion must be satisfied for both the background and the foreground.

To compute the color histogram for the background pixels, we first transform the RGB color space to the perceptually uniform CIELAB color space. Then, we quantize that space into b^3 bins, where b is the number of bins along each of the L*, a*, and b* axes. Note that some of these b^3 bins will always be empty since the RGB color space is a subset of the CIELAB color space [17]. Representing this histogram with a b^3-dimensional vector, we subsequently reduce its dimensionality by, first, retaining only the valid L*a*b* histogram bins, and, then, by applying FastMap as described in Sections V-B and IV-D, respectively. We will use d to denote the retained dimensionality for the color representation of the background pixels.

To measure the similarity between any two background histograms, we use the angular distance metric defined by Eq. 1:

$$\text{dist}_{\text{Angle}}(\vec{v}_1, \vec{v}_2) = \frac{180}{\pi} \cos^{-1}\left(\frac{\vec{v}_1^T \vec{v}_2}{|\vec{v}_1|\ |\vec{v}_2|}\right) \qquad (1)$$

where $\vec{v}_1$ and $\vec{v}_2$ are the d-dimensional vector representations of the two histograms, respectively. Generally, we normalize the histogram vectors so that they are of unit magnitude, which does away with the denominator in the above formula.

As for characterizing the foreground (ie., the center pixel of the image patch), we use its spectral signature consisting of the spectral responses measured at the corresponding surface location on earth. For example, in a 4-band satellite image, each pixel has 4 values, one per spectral band, and the spectral signature is a vector consisting of these four numbers. To measure the similarity between any two spectral signatures, we use the "L1 distance metric". Note that both the Cosine distance metric for the histogram vectors and the L1 metric for the foreground spectral vectors are fast to compute – an important consideration in big-data processing.

C. Similarity Search

Since a patch is represented by two *semantically different* characterizations — a d-dimensional vector for the background color distribution and a 4-dimensional vector for the foreground spectral values — that raises the question of how to actually form the similarity groups, especially because we want to enforce the similarity constraint on the two characterizations conjunctively. Note that, in our big-data context, we do not have the luxury of comparing every pair of patches to decide whether or not they belong to the same similarity group. If it were possible to compare *every* pair of the patches, the two similarity conditions could be enforced simultaneously in each pair-wise comparison. Therefore, we are left with three options:

Option 1: Concatenate the d-dimensional color-histogram vector for the background with the 4-dimensional spectral-property vector for the foreground to form a single vector representation for a patch and then apply one of several distance metrics to the vectors for comparing the similarity of the patches.

Option 2: First cluster the patches with respect to just the foreground pixels and then subject each of the clusters thus obtained to further sub-clustering on the basis of the similarity of the background color-histogram vectors.

Option 3: By reversing the two steps in the previous option; that is, by first clustering the patches with respect to the background color-histogram vectors and then further sub-clustering those clusters on the basis of the similarity of the foreground spectral vectors.

Despite its appearance to the contrary, the first option listed above is not appropriate since it cannot guarantee conjunctive enforcement of the two separate and distinct similarity constraints. And the second and the third options are logically equivalent.

What is interesting is that while the second and the third options are logically equivalent, they entail different degrees of computational effort to arrive at the same final conclusion. The main reason for that has to do with how the vectors that represent the color histograms for the background are

distributed vis-a-vis the distribution of the spectral vectors that represent the foreground.

The distribution of the histogram vectors is such that with the linear-time LSH algorithm applied to such vectors, it is possible to implement a fast approximated solution for creating sufficiently small clusters so that a subsequent pairwise comparison of the samples within each histogram-similarity based cluster for enforcing the similarity of the spectral vectors results in an overall computationally efficient structure for the conjunctive enforcement of the two different similarity constraints. The opposite approach would consist of first applying a Euclidean-distance based LSH to cluster the entire data on the basis of the similarity of the spectral vectors and then subjecting the data elements withing each resulting cluster to the histogram based similarity constraint. Unfortunately, the clusters generated by the second approach tend to be much loo large, making the overall computation relatively inefficient. For the reasons explained above, we chose Option 3 for the conjunctive enforcement of the two different similar constraints.

Permeating all three options listed above, including obviously our chosen Option 3, are the consequences of the non-transitivity of the similarity constraints that we mentioned earlier in the Introduction. As stated there, if we were to apply any of the three options to the entire data set, we are highly likely to end up with a single concise set, which is not a very useful thing to happen. To get around this difficulty, we introduce the notion of *similarity graph* in Section IV-E. A similarity graph is generated by applying a pairwise spectral comparison criterion to all the patches considered similar by the LSH algorithm (on the basis of the background similarity through the histograms associated with the backgrounds). These pairwise comparisons yield what we call similarity neighborhoods. Every patch in a given similarity neighborhood is *directly* within the similarity distance of the patch for which the similarity neighborhood was constructed. The collection of all such similarity neighborhoods constitutes the similarity graph. Note that it is likely that there would patches that would be shared by different nodes.

In the rest of this section, we first describe how we reduce the data dimensionality of the color-histogram vectors before clustering them using LSH. Subsequently, we bring in the spectral data vectors for the foreground pixels to further refine the clusters.

D. Reducing the Dimensionality of the Histogram Representation for the Background Pixels in a Patch

As explained in Section IV-B, the background pixels in a patch are represented through a three dimensional histogram in the L*a*b* space. As we will show in Sections V-A and V-B, the bin structure used for the histogram results in a vector representation of the background that has 9024 elements in it. This is obviously much too large a dimensionality. Fortunately, with dimensionality reeducation, we can bring it down to less than 100, depending on the data in the ROI (Region of Interest).

There exist many dimensionality reduction strategies for multidimensional data, however most are not appropriate for the big data scenarios involving tens of millions of patches extracted from satellite images. Consider, for example, what is perhaps the most commonly used method for dimensionality reduction, PCA, which carries out an eigendecomposition of the covariance matrix of the data. In our case, the data would consist of 9024-element vectors for the L*a*b* color histograms, whose covariance matrix would be of size 9024×9024, a matrix with close to 100 million elements. Now if we only had a small number of patches from which to generate the reduced-dimensionality representation, we could take advantage of the fact the rank of the covariance matrix would not exceed the number of patches available and translate that fact into a highly efficient algorithm for the eigendecomposition of the covariance matrix [18]. However, that is not case with the work described in this paper — with the number of patches running into millions, we have no hard constraint on the rank of the covariance matrix. We run into similar problems if we try to use the other methods, such as Incremental PCA [19] and Sparse PCA [20]. In light of these difficulties associated with the more traditional approaches, we have chosen to use the FastMap [21] method for our work.

FastMap works by preserving, up to certain level of precision depending on the stopping criterion used, all pairwise distances among the entire dataset while reducing the data dimensionality as much as possible. The algorithm has time complexity of $O(d \times n)$ where d is the number of dimensions after reduction and n the size of the dataset.

The FastMap algorithm involves just three basic steps: First, it takes the entire dataset and quickly finds the two furthest points (ie., two furthest data elements) away from each other. Then, taking the line joining these two points as the first axis in the reduced dimensionality representation of the data, the algorithm then projects all the data points onto this line to calculate the first coordinate value of the points in the reduced-dimensionality representation. Finally, the algorithm projects all the data points into a hyperplane perpendicular to the axis just constructed. These three steps are repeated with the data projected into the hyperplane until the stopping criterion is met. Fig. 2 summarizes these three basic steps.

As for choosing the stopping criterion, one could obviously stop when the desired number of dimensions is reached. In general, though, that is not likely to be a useful criterion since one would not know the desired dimensionality in advance. A more useful criterion consists of stopping the iterations when the low-dimensional subspace retains a certain specified fraction of the total variation in the data.

To elaborate, note that each axis is formed by a line segment like the (P_a, P_b) segment shown in Fig. 2. The length of this segment determines the range of data variation on that axis. The data variation decreases in each iteration. Therefore, we can stop when the segment length is less than some fraction of the sum of the segment lengths encountered so far. More formally, let L_i be the length of the line segment at iteration i, we stop when the following inequality is satisfied:

$$\frac{L_d}{\sum_{i=1}^{d} L_i} < T \tag{2}$$

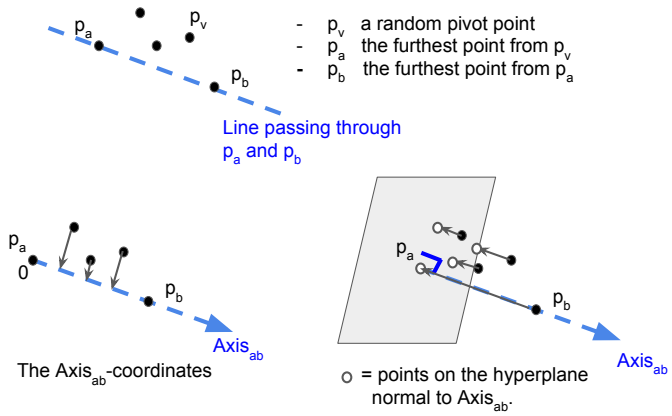

Fig. 2. FastMap first finds p_a and p_b, two furthermost points (ie., data elements) away for each other. This step takes $2 \times n$ comparisons, where n is the number of data elements. It then maps all points onto the line segment formed by this pair of points for estimating the first coordinate of all the points in the reduced-dimensionality representation. Finally, it maps all points into a hyperplane that is normal to the line passing through p_a and p_b. These three steps are repeated with the points in the hyperplane. Each such repetition adds one more dimension to the low-dimensional representation of the entire dataset.

As for the value of T, we found that the value of 0.001 (which is tantamount to retaining 99.9% of the total variation) limits the pair-wise Cosine distance error to less than one degree. Note that we obtained this result using our data, which are normalized histograms.

E. Creating a Similarity Graph for the Image Patches

Keeping in mind that the vector representations for the background color histograms can still reside in a high-dimensional space after dimensionality reduction, even the supposedly efficient algorithms that avoid exhaustive pairwise comparisons, such as those based on nearest neighbor search (NNS) with KD-trees, SR-trees, and cover trees [4]–[8] are not appropriate for solving our problem of forming similarity neighborhoods from the background color-histogram vectors because their performance (either the running time or the memory requirement) degrades exponentially as the data dimensionality increases.

Locality Sensitive Hashing (LSH) [22], [23], on the other hand, has emerged as an attractive alternative to tree-based nearest neighbor search algorithms for high-dimensional data. Just like the tree-based approaches, LSH does not make exhaustive pair-wise comparisons. Additionally, and most importantly, LSH can be implemented to have constant average search time, making it highly desirable for similarity based searching in very large datasets. The only drawback is that LSH is an approximated nearest neighbor (ANN) algorithm and may not always find the exact nearest neighbor. Nevertheless, LSH is suitable for applications when datasets are large and finding the exact nearest neighbor isn't critical. It has been shown that for high dimensional data, LSH significantly outperforms SR-tree, a representative of tree-decomposition-based indexing techniques [24].

To briefly review how LSH works, as its name implies, LSH uses locality sensitive hashing for nearest neighbor search. A hash function is considered to be locality sensitive if it places "nearby" samples in the same bucket with a high probability, and if it places "far apart" samples in different buckets, again with a high probability. Two data elements are considered to be "nearby" if the distance between them is at most d_1 and two data elements are considered "far apart" if the distance between them is at least $d_2 = c \times d_1$, where $c > 1$ is the approximation factor. The quality of such a hash function is measured by two probabilities p_1 and p_2, where the former is the probability of collision for "nearby" samples and the latter the probability of collision for "far apart" samples. For obvious reasons, you'd want p_1 to be as high as possible and p_2 to be as low as possible.

In practice, it is not possible to find a single hash function with the property described above. However, it has been shown that a large number of hash functions working together in an AND-OR structure can possess this property [25]. One starts out with a basic hash function that places nearby samples in the same bucket with a high probability, but that, at the same time, places any two far-away samples in the same bucket with NOT a sufficiently low probability. Subsequently, one can require that for any two given samples to be considered similar they must be in the same bucket for a set of different hash functions, these multiple hash functions being random variations of the same basic hash function with respect to at least one of its parameters. (We'll use r to denote the number of hash functions in such a set.) This is referred to as enforcing an 'AND' operation over r hash functions to significantly decrease the probability of two far-apart samples being considered similar. Since the 'AND' operation can also somewhat reduce the probability of nearby samples as being considered similar, we take an 'OR' b sets of r hash functions to restore or further enhance that probability. Choosing r and b in order to achieve desired values for p_1 and p_2 becomes a design issue for any implementation of LSH.

Hyperplane LSH [26], a commonly used implementation of LSH for similarity measure shown in Eq. 1, consists of using randomly oriented hyperplanes for hashing. A hyperplane gives us a two-bucket hash table: When a numerical data element is projected on a hyperplane perpendicular to a hyperplane passing through the origin, the projection is either in the positive half-space corresponding to that hyperplane or the negative half space. By constructing b sets of such randomly placed hyperplanes, with r hyperplanes in each set, we can achieve the desired discriminations between nearby and far-apart data elements.

Applying LSH on the background color-histogram vectors, each patch p is associated with a set of patches that are directly within the angular similarity threshold of p on the basis of just the background color-histogram similarity. For each patch p, the set of all similar patches thus discovered constitutes p's similarity neighborhood.

Subsequently, patches and their neighborhoods are converted into a similarity graph by testing within each neighborhood for patches having similar foreground spectral signatures with respect to the associate patch. The output of this exercise is represented by a similarity graph in which a pair of two vertices, with each vertex corresponding to an image patch,

share an edge if they are similar both with respect to the background and the foreground contexts. Although the worst-case time complexity of this algorithm is given by $O\left(|V|^2\right)$, where V is the number of patches, the worst case happens only when the entire similarity graph is a clique (for which the number of edges is quadratic on the number of vertices; that is $|E| = O\left(|V|^2\right)$).

F. Population Compression

We represent a pixel along with the image patch that provides its surrounding context by a vertex in the similarity graph mentioned previously. When a particular vertex is selected for inclusion in the concise dataset, all other vertices that are similar to it are subsequently marked as redundant in the similarity neighborhood of the selected vertex. The task of data reduction is then to find a minimal set of vertices that maximally cover the overall redundant set of vertices. This optimization problem can be formulated as the "Set Cover problem", which is a well-known NP-Complete problem [27]. The Set Cover problem is also closely related to the Dominating Set problem in graph theory. An approximated solution using greedy algorithm, as described in the subsection that follows, can produce a solution that is guaranteed to be within $O\left(\log|V|\right)$ factor of the optimal solution where V is the set of vertices in the graph. In terms of algorithmic complexity, the greedy algorithm runs in $O\left(|E|\right)$ time, where E is the set of edges in the graph.

G. Creating a Concise-set Representation of the Population

In the previous section, we mentioned using a greedy algorithm to find an approximated solution to the Set Cover problem. We now describe the algorithm in detail and show how it returns a concise-set representation for the target population.

Algorithm 1 Create a Concise-set Representation

Input: U = Set of all items. (eg., image patches)
Output: A concise-set representation of U

1: $S \leftarrow \{\{\text{LSH-GetItemsSimilarTo}(e)\}, \forall e \in U\}$
   // S = Set of candidate similar-item sets (clusters).
2: $R \leftarrow$ [ ] // Array of cluster representatives.
3: $W \leftarrow$ [ ] // Array of redundancy weights.
4: $C \leftarrow$ [ ] // Array of similar-item sets (clusters).
5: **while** SomeItemsNotCovered(U, C) == True **do**
6: $(c^*, r^*) \leftarrow$ GetMaxCluster(S) // Max cluster, c^*, and its representative, r^*.
7: $S \leftarrow$ RemoveAndUpdateClusters (S, c^*)
8: $R \leftarrow$ Append (R, r^*)
9: $C \leftarrow$ Append (C, c^*)
10: $W \leftarrow$ Append $(W, |c^*|)$
11: **end while**
12: **return** (R, C, W)

Algorithm 1 takes as input a set of indices representing image patches IDs in the population. The algorithm outputs a triplet, (R, C, W), as the concise-set representation of the population, where:

- R is the *concise dataset* that is an array of cluster representatives, with one representative for each cluster of vertices in the approximated similarity graph.
- C is an array of clusters. That is, for each i, $C[i]$ is a cluster represented by a set of vertices. Each vertex in $C[i]$ corresponds to an image patch that is similar to the cluster representative, $R[i]$.
- W is an array of cluster sizes. That is, $W[i]$ is the size of cluster $C[i]$ for which $R[i]$ is the corresponding cluster representative.

H. Annotating the Concise Dataset

After we have created a concise-set representation of the image patch population, we proceed to annotate the center pixels (the foreground pixels) of the image patches retained in the concise dataset. That is, a human annotator looks at image patches associated in the R array, as returned by Algorithm 1, and assign ground-truth labels to their center pixels. The human annotator does not assign labels to the surrounding pixels in the image patch. In keeping with our earlier discussion, an image patch is modeled as containing contextual pixels (ie., the background) surrounding the center pixel (ie., the foreground) for which we want the human annotator to supply a class label.

I. On Extending the Concise-Set Representative Label to the Other Members of the Same Set

The most straightforward way to extend the human-supplied annotation label for a cluster representative in the R array is to simply assign the same label to all the other members in same cluster.

However, it is possible to conceive of alternatives to the obvious mentioned above that have ramifications regarding the size of the overall representation created for a large dataset involving hundreds of satellite images. One could, for example, argue that since — *seemingly* — all the other members in a cluster are redundant vis-a-vis the cluster representative for constructing or evaluating a classifier, why not just retain *only* the cluster representatives and discard the rest of the data. The problem with that logic is that such a data reduction could significantly impact the class probabilities associated with different land types in a geographic regions and, consequently, result in erroneous classification performance results (regardless of the choice of the classifier).

To get around this difficulty, and, at the same time, to benefit from the compression made possible by the R array, we could associate the size of each cluster with each cluster representative in R. This is indeed one of the options made available by our concise-set framework when we generate the final representation for the satellite data. We refer to this as the "The Weighted Representative Method (WRM)" for creating the final representation.

When not using the weighted representative method, the system simply extends the human-supplied annotations for each cluster representative in R to the rest of the rest of the

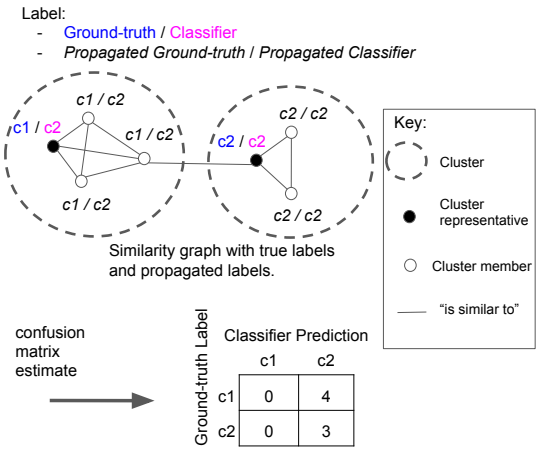

Fig. 3. An example of the "weighted-representative" method: The similarity graph shown here has two clusters depicted by the dashed circles. Each cluster has a representative shown as a black dot. The classifier is applied to only the cluster representatives and the classifier generated labels for the representatives propagated to the rest of the cluster. Each vertex is shown with two labels, one for the ground-truth and the other for classifier-generated, and, in each case, they are both propagated from the cluster representative. In this example, there are four vertices labeled "c1/c2" and therefore the corresponding "c1/c2" entry in the estimated confusion matrix is 4. Similarly for the "c2/c2" entry.

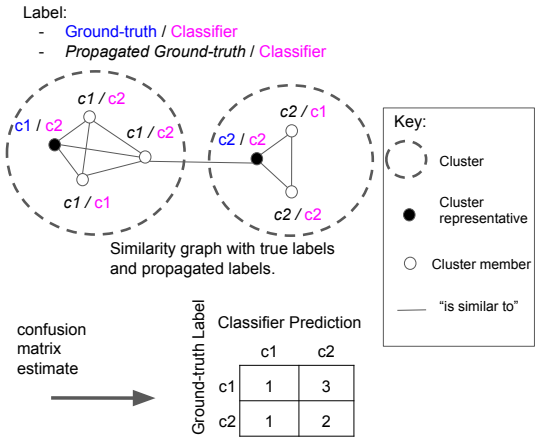

Fig. 4. An example of the "whole-cluster method": The depiction here parallels the one shown in Fig. 3 except for the fact that the classifier is applied to every member of each cluster. For each vertex, the first label is the ground-truth label as propagated from the cluster representative and the second label is as produced by the classifier. In the example shown, there are three vertices labeled "c1/c2" and therefore the corresponding "c1/c2" entry in the estimated confusion matrix is 3. Similarly for the other entries in the confusion matrix.

cluster members. In order to make a distinction with WRM, we refer to this method as "The Whole Cluster Method (WCM)".

Fig. 3 illustrates an example of estimating the confusion matrix using the "weighted-representative" method, and Fig. 4 illustrates an example of estimating the confusion matrix using the "whole-cluster" method. As the reader would expect, the "weighted-representative" method is simple and fast, but it tends to either overestimate or underestimate the classifier's true performance. On the other hand, the whole-cluster method produces a better performance estimate, although at the cost of doing more work. Note that the annotation effort is the same for both methods.

J. A Quality Coefficient for Choosing the Best Value for the LSH Similarity Threshold

It should be obvious to the reader that the validity of the confusion matrices as produced by the two methods presented in the previous subsection depends significantly on the similarity distance threshold used in the LSH algorithm. A similarity distance threshold that is too large would degrade the quality of the concise dataset with regard to the following two considerations: (1) We will have increased tendency of the data samples from disparate classes to populate the same clusters; and (2) The dataset may end up with fewer land-type classes than there actually are in the satellite data.

At the same time, a similarity distance threshold that is too small would generate too many small clusters, which would increase the human burden associated with supplying the ground-truth label for the representative of each cluster. That leads to the question of whether there is any automatic way to determine a good value to use for the similarity distance threshold. As we discuss below, the answer to the above question is yes.

Our answer presented in this section is based on the following observation: Since the similarity neighborhoods returned by the LSH algorithm consist of the vertices that are hashed into the same bucket, it is possible for a similarity neighborhood to intersect multiple clusters as returned by LSH (see Fig. 5). When a vertex lies simultaneously in multiple similarity neighborhoods, it *may* acquire a set of different propagated *class* labels. As to the reason for *may*, first note that LSH will form multiple distinct clusters for the same ground-truth class label. LSH forms a cluster on the basis of the approximate similarity of vertices. Subsequently, the human annotator labels one cluster representative and then that label is propagated to all the other members. For an example of there being multiple clusters for the same ground-truth class label, think of the pixels corresponding to the label "road". Since roads, in general, are made from different materials — concrete, asphalt, gravel, or just plain dirt — any automatic clusterer is likely to place the road image pixels in different clusters that may or may not be overlapping. Such different clusters for the same class label are NOT the source of inconsistency we are talking about. For the sort of inconsistencies we are talking about, consider the image pixels that, through propagation from the cluster representatives, simultaneously acquire two different labels such as "roof" and "road". This can easily happen since in many parts of the world we have roofs, especially flat roofs, that are made from the same materials that go into road construction. So, human-annotated "road" pixels may get hashed into an LSH bucket that also contains "roof" pixels and vice versa. Such a vertex contributes to inconsistencies in the labeling of the data. We claim that when all the propagated class labels are consistent, we have chosen a good value for the similarity distance threshold. So, if we can find a way to estimate the number of the vertices with inconsistently propagated class labels, we can assess the appropriateness of the value chosen for the similarity distance threshold.

Let T be the total number of vertices and let I be the

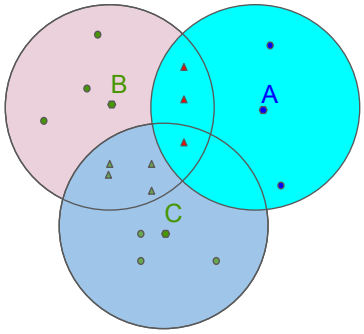

Fig. 5. In this example for illustrating the notion of consistency, while we have three overlapping clusters in some feature space, two of the clusters, represented by the cluster representatives B and C, carry the same propagated ground-truth label. On the other hand, the cluster represented by A carries a different propagated label. Note that true ground-truth labels are provided only for the cluster representatives. We have a total of 18 vertices in the three clusters. In the figure, small circular dots represent vertices that belong to only one cluster while small triangles are vertices that simultaneously belong to two or more clusters. We see that 7 of the 18 vertices have two or more cluster memberships. However, on account of the equivalency of the class labels for B and C, only three vertices have different class labels. Therefore, the ground-truth consistency (See Eq. 3) is $1 - \frac{3}{18} = 0.833$.

number of vertices with inconsistent class labels in different clusters, we define *ground-truth consistency* of the concise-set representation as:

$$\text{ground-truth consistency} = 1 - \frac{I}{T} \tag{3}$$

Note that the ground-truth consistency is only meaningful when there are overlapping clusters.

V. Setting the Experimental Parameters for WorldView2 Imagery

Recall that at the heart of our approach for constructing a concise-set representation of the data is the construction of an approximate similarity graph with the help of LSH. As mentioned in Section IV-F, each vertex of this similarity graph represents a pixel along with the image patch that provide geographical context for the pixel. When the vertices are tested for similarity, it is done on the basis of both the spectral signatures at the pixels themselves and the attributes of the associated image patches. As mentioned earlier, each image patch is represented by a histogram of the pixel "colors" in the patch.

Therefore, as the reader can imagine, how many bins to use for the patch histograms and what similarity thresholds to use when comparing the histogram-based feature vectors, etc., are some of the critical experimental parameters in our approach. The goal of this section is to discuss how we choose values for these parameters. Our parameter selection strategy is conservative in the sense that we aim to minimize the computation overhead while keeping the final concise dataset small. Using these parameters, we will present our experiments and the results in Section VI.

In this section, we will describe how we set the parameters mentioned above for WorldView2 satellite imagery.

Fig. 6. A typical region in the Chile ROI from our WorldView2 dataset. Satellite image acquisition time: 2011-10-09 at 14:56:34 UTC

The WorldView2 data we use have four pan-sharpened spectral bands that are ToA corrected. ToA stands for Top-of-Atmosphere. This correction is a standard procedure for normalizing satellite data taken at different angles and distances relative to the earth's surface being imaged. The four spectral bands are Red (630-690 nm), Blue (510-580 nm), Green (450-510 nm), and Near Infrared (770-895 nm). And the pan-sharpened data have ground resolution of 0.5m per pixel. At this level of resolution, it is possible for a human annotator to identify typical narrow one-lane roads and alleys. In order to apply our framework to this data, we must first set the size for the image patches. We have experimentally determined that patches of size 101×101 pixels give a good balance between competing requirements. This size is large enough to provide a reliable context for the labeling of the center pixel and small enough that it does not include too much diversity within a single patch. The preprocessing step involves cropping image patches centered at all pixel locations in the satellite image. This step results in a dataset consisting of image patches totaling the number of pixels in the satellite images. We use the sliding window approach to optionally control the size of the dataset by allowing the sliding window to have the option of skipping every few pixels, in addition to skipping the invalid and border pixels.

For the purpose of exploring the best choices to make for the parameters, we manually annotated a dataset from the Chile ROI [1]. Fig. 6 shows a typical area in this ROI. The visually recognizable different regions in this area that are homogeneous in terms of the class labels are demarcated with a graphical tool. Five major land-types are used in this exercise: building, road, tree, water, and soil. The output of this exercise is a set of pixels along with their image patches, with each pixel along with its associated image patch serving as a vertex in the similarity graph. The important thing to remember is that every vertex created in this manner has a human-supplied class label. Subsequently, we use this data to investigate the power of our proposed formalism for creating concise-set representations for the different choices for the parameters of the framework.

We will show in Section VI that the parameters set in this

manner from the data collected over Chile work effectively in another part of the world that is significantly different — Australia. That fact provided further validation for our framework.

A. Color Spaces and Histogram Quantization

As mentioned earlier, we refer to the center pixel in an image patch as the foreground and the rest of the pixels in the patch as the background. And, as mentioned in Section IV-B, we represent the background of an image patch by its color histogram. In our experiments, we investigated both the RGB and CIELAB color spaces for the histogram. The RGB space results in b^3 bins. Subsequently, the choice of b controls the size of the generated histogram (ie., $b = 4$ results in $4^3 = 64$ bins and, therefore, a 64-dimensional feature vector). To reduce the number of bins in the histogram, we examined the CIELAB color space and remove the invalid bins – those bins that are not occupied by any color in the RGB space. We showed the obtained histogram sizes for both RGB and CIELAB in Table I. There is also another reason for using CIELAB color space over other color spaces that are not perceptually uniform. In our case, we want the image patches to be grouped together according to human perception of color similarity.

To determine the best value to use for b, we examined a wide range of values between $b = 4$ to $b = 64$. Note that there is no need to examine $b \geq 66$ as it has been shown to be the sufficient for CIELAB color quantization [28].

For each value of b, we constructed a similarity graph as described in Section IV-E using the Chile dataset. Then, we calculated the cardinality of each cluster and found average cardinality over all the clusters.

Fig. 7 illustrates the effect of histogram size on the average cardinality of the clusters for different similarity thresholds. We observe a decrease in the average values between $b = 4$ and $b = 8$ followed by a gradual increase after $b = 8$. It seems as the histogram size increases, more and more data become similar to one another. We note that this phenomenon does not always happen in general and can be attributed to data distribution and quantization effect.

TABLE I
COLOR HISTOGRAM SIZES FOR RGB AND CIE-LAB COLOR SPACES.

b bins per axis	RGB histogram size	LAB histogram size
4	64	45
8	512	245
12	1728	652
16	4096	1388
20	8000	2490
24	13824	4080
28	21952	6228
32	32768	9024
64	262144	64508

B. Calculating the Best Value to Use for the Similarity Threshold

In this section, we investigate the best choice to make for the similarity threshold that is needed by the LSH algorithm.

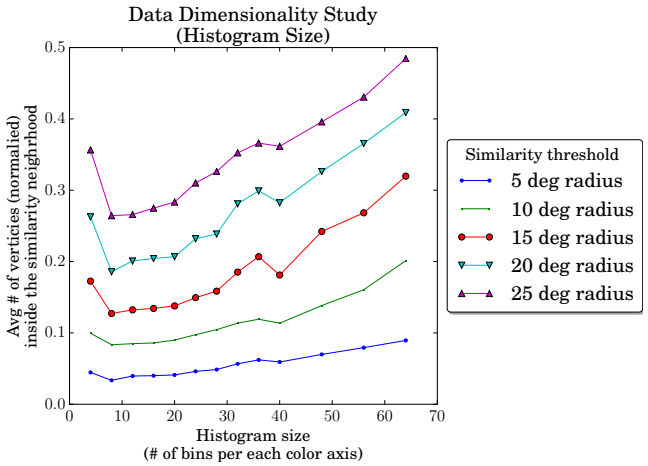

Fig. 7. Proportion of the data found inside the similarity neighborhoods (clusters).

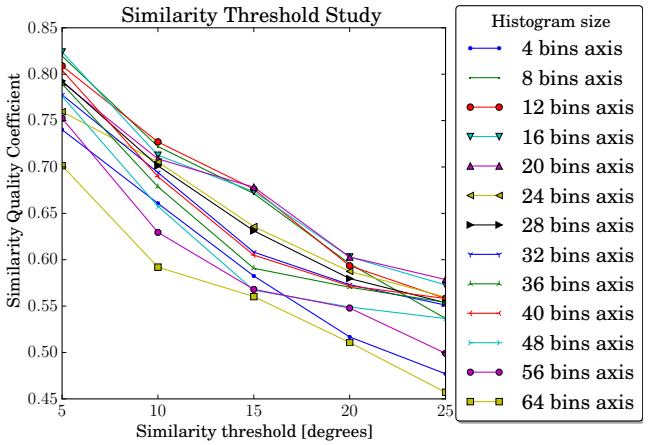

Fig. 8. Plots of background Similarity Quality Coefficient (SQC) as a function of similarity threshold.

Recall that we are talking about similarity between a pair of vertices in the similarity graph, with vertex standing for a pixel along with the associated image patch. In the previous section, we concluded that even with the same similarity threshold, how the data gets clustered changes when the data dimensionality changes.

We use the following logic to evaluate the quality of a similarity threshold: We examine each cluster and count the number of vertices in the cluster whose human-supplied class labels are the same as the human-supplied label for the cluster representative. This count is normalized by the cardinality of the cluster. The average of this ratio over all the clusters is a quality coefficient for a given similarity threshold. We refer to this coefficient as the "Similarity Quality Coefficient (SQC)". In Fig. 8, we show the dependence of SQC on different values for the similarity threshold. Note, each plot corresponds to a different histogram size.

We see in Fig. 8 that SQC values are greater than 50% for similarity thresholds less than 20 degrees, regardless of the histogram size. Therefore, in our experiment, we conservatively set the similarity threshold at 10 degrees so that at least 60% of the image patches found within a cluster can be

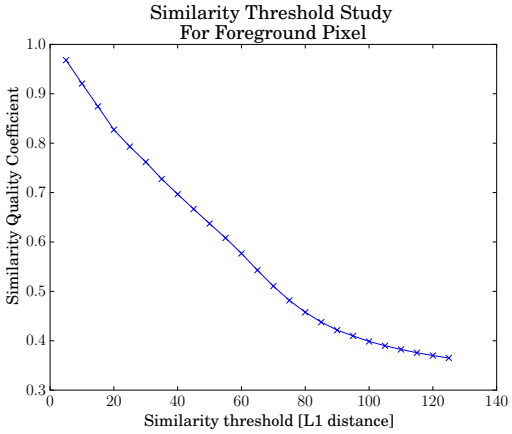

Fig. 9. Plot of foreground Similarity Quality Coefficient (SQC) as a function of similarity threshold.

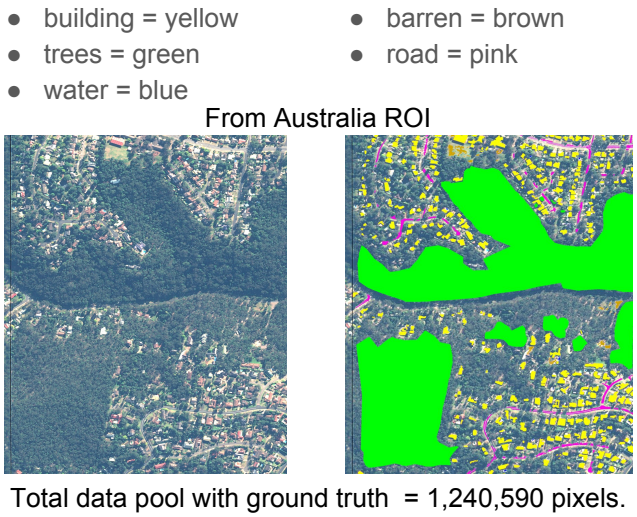

Fig. 10. Annotated data within a 1km by 1km region in the Australia ROI.

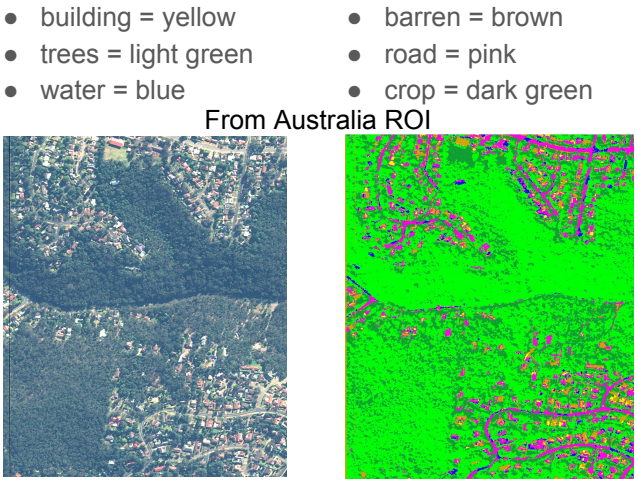

Fig. 11. Classification map of a cropped 1km by 1km region in the Australia ROI. The provided classifier from [1] is treated as a blackbox.

expected to have the same ground-truth label as the cluster representative. We could have picked any values smaller than 20, but a tighter similarity threshold increases the hashing time in LSH. We pick 10 over 15 because we would like to have higher SQC values while still keeping the hashing computation low. As for the histogram size, we picked $b = 32$ (9024 dimensions) for our experiments (See Table I). We noted from the study in [28] that b does not need to be more than 66 and thus $b = 32$ seemed like a good trade-off between color fidelity and computational efficiency. We obtained the same result when we repeated the similarity threshold study on the lower-dimensional data described in Section IV-D

Fig. 9 shows the SQC plot for the image foreground (ie., the spectral signature of the center pixel). Using this plot, we set the similarity threshold at 50 so that about 60% (ie., slightly over a majority) of the image patches found within this threshold can be expected to have the same ground-truth label as the cluster representative.

VI. Validation

A most important aspect of the comparative results we report in this section is that the datasets we used for these results are drawn from a part of the world that is different from what was used in Section V for parameter estimation. We refer to these datasets as our "validation datasets". *We however use the same parameters that we estimated in Section V to create a concise-set representation of the validation datasets.* What that implies is that the parameters estimated in Section V possess some measure of generality. The extent of this generality is yet to be investigated. Whereas the dataset used in Section V came from Chile, the validation datasets are from Australia. The left half of Fig. 10 shows an example of the area from which the validation datasets were drawn. The acquisition time for the corresponding WorldView2 satellite image is 2013-08-06 at 00:25:40 UTC.

As we mention later here, the true human-supplied class identity is known for every element in the validation datasets. This allows us to calculate the "true" confusion matrix on a validation dataset for any given classifier. We have chosen a Support Vector Machine (SVM) classifier created using the approach in [1] for this validation study since SVM-based classifiers are commonly used today for land-cover classification and require fewer training data [29]. Fig. 11 shows the classification map obtained by applying the classifier on a 1km by 1km region in the Australia ROI.

In this section, we will show that the confusion matrix calculated from the concise-set representation of the validation dataset is significantly closer to the true confusion matrix as compared to the confusion matrix calculated by the traditional random sampling method. In order to convince the reader that the above stated result is not a chance result — that is, a result that is specific to one particular random sampling of the data — we repeat the traditional random sampling method multiple times.

As we did for the parameter estimation dataset in Section V, for the validation datasets used in this section, we manually supply the class label for $1,240,590$ pixels within a 1 km $\times$1 km area in the Australia ROI. Hence, the largest validation dataset we can draw has $1,240,590$ units. Note that each unit in the dataset is represented by a 101×101 image

patch centered at the annotated pixel. The manual annotations were supplied with the graphical tools that we mentioned in Section V. As mentioned there, these tools allow us to quickly demarcate large sections of a satellite image that are homogeneous with respect to a class label. This allows the system to quickly assign a class label to large set of pixels, all at the same time. Fig. 10 shows the demarcated regions and their assigned class labels.

The following considerations go into calculating and comparing the confusion matrices produced by the concise-set representation method and the traditional random sampling method:

- To estimate the true confusion matrix from the concise-set representation, we use the "whole-cluster" method described in Section IV-I.
- To directly compare a pair of confusion matrices, we first normalize the confusion matrices and then compute the sum-of-squared-difference (SSD) between the two confusion matrices being compared. A small SSD value indicates a close match between the two confusion matrices.

A. Concise-set Representation versus Random Sampling

We repeat SVM based classification using the traditional random sampling approach 100 times and, for each trial, compare the classifier performance individually with what is obtained through the concise-set representation approach. The plot in Fig. 12 shows the SSD values for the 100 random trials. Here the SSD value quantifies the performance error relative to the true performance obtained from the entire validation dataset. The particular validation dataset used in this experiment has 1000 units and is randomly drawn from the $1,240,590$ annotated data pool mentioned in the previous section. We show the true performance, represented as a normalized confusion matrix, in Table II.

TABLE II
TRUE PERFORMANCE OF THE CLASSIFIER UNDER TEST IS OBTAINED BY TESTING THE CLASSIFIER ON THE ENTIRE VALIDATION DATASET AND NORMALIZING THE RESULTING CONFUSION MATRIX BY THE SIZE OF THE VALIDATION DASETSET. FOR THIS EXPERIMENT, WE LIMIT THE SIZE OF THE VALIDATION DATASET TO 1000.

True \ Predicted	Crop	Barren	Tree	Water	Building	Road
Crop	0.0	0.0	0.0	0.0	0.0	0.0
Barren	0.005	0.002	0.0	0.0	0.0	0.001
Tree	0.125	0.0	0.705	0.0	0.0	0.0
Water	0.0	0.0	0.0	0.0	0.0	0.0
Building	0.001	0.031	0.013	0.005	0.018	0.074
Road	0.001	0.0	0.001	0.0	0.0	0.018

The size of the random sample in each trial is set to 126 to match the size of the concise dataset. Note that we purposely keep the validation dataset small for this experiment so that we can run multiple trials within a short time. We study the effect of larger validation dataset in Experiment VI-D. Tables III and IV show the normalized confusion matrices obtained using our proposed approach and the transitional random-sampling approach. Comparing to Table IV, we see that the normalized confusion matrices shown in Table II and Table III are more similar to each other as the SSD value between them is smaller.

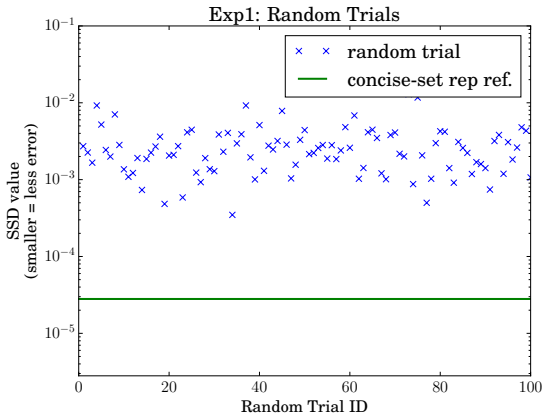

Fig. 12. None of the SSD values from the 100 random trials gives a better performance estimate than the SSD value obtained using the concise-set representation approach. Validation dataset size = 1000. Concise dataset size = 126. Random dataset size used for each trial = 126. Ground-truth consistency = 0.997.

TABLE III
THE ESTIMATED CONFUSION MATRIX USING OUR CONCISE-SET APPROACH. THE SIZE OF THE CONCISE DATASET IS 126. THE SSD VALUE BETWEEN THIS ESTIMATE AND THE TRUE CONFUSION MATRIX SHOWN IN TABLE II IS 2.8×10^{-5}.

True \ Predicted	Crop	Barren	Tree	Water	Building	Road
Crop	0.0	0.0	0.0	0.0	0.0	0.0
Barren	0.005	0.002	0.0	0.0	0.0	0.001
Tree	0.123	0.0	0.707	0.0	0.0	0.002
Water	0.0	0.0	0.0	0.0	0.0	0.0
Building	0.004	0.031	0.011	0.005	0.018	0.073
Road	0.0	0.0	0.001	0.0	0.0	0.017

TABLE IV
AN ESTIMATED CONFUSION MATRIX FROM ONE RANDOM TRIAL USING THE TRADITIONAL RANDOM SAMPLING APPROACH. THE SIZE OF THE RANDOM DATASET IS KEPT THE SAME AS IN TABLE III. THE SSD VALUE BETWEEN THIS ESTIMATE AND THE TRUE CONFUSION MATRIX SHOWN IN TABLE II IS 2.38×10^{-3}.

True \ Predicted	Crop	Barren	Tree	Water	Building	Road
Crop	0.0	0.0	0.0	0.0	0.0	0.0
Barren	0.0	0.0	0.0	0.0	0.0	0.0
Tree	0.15	0.0	0.75	0.0	0.0	0.0
Water	0.0	0.0	0.0	0.0	0.0	0.0
Building	0.0	0.023	0.0079	0.0	0.015	0.039
Road	0.0	0.0	0.0	0.0	0.0	0.0079

We observe from the plot shown in Fig. 12 that none of the SSD values from the random sampling approach is smaller than the SSD value obtained using the proposed approach. As for the ground-truth consistency estimate, the value in this case is 0.997 (See Section IV-J and Eq. 3).

B. Concise-set Representation versus the Average of the Results Obtained with Randomly Drawn Samples

In this experiment, we average the confusion matrices obtained by the traditional random sampling approach over many trials and then compare it with the the confusion matrix

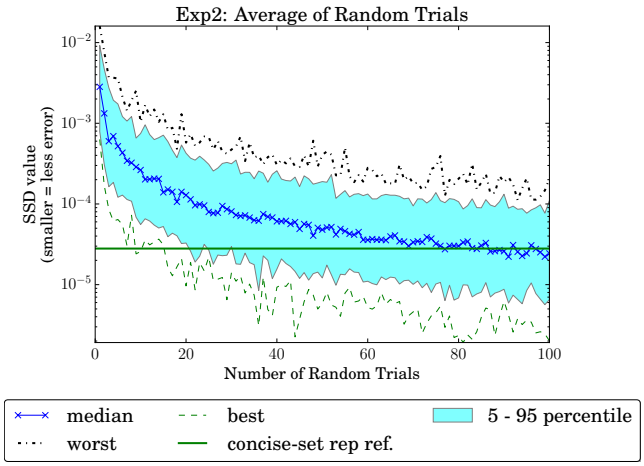

Fig. 13. From the "best" performance curve, we see that averaging over at least 10 random trials is needed for the random sampling approach to have any chance of outperforming the concise-set representation. Validation dataset size = 1000. Concise dataset size = 126. Random dataset size used for each trial = 126. Number of repeated experiments per trial = 100. Ground-truth consistency = 0.997.

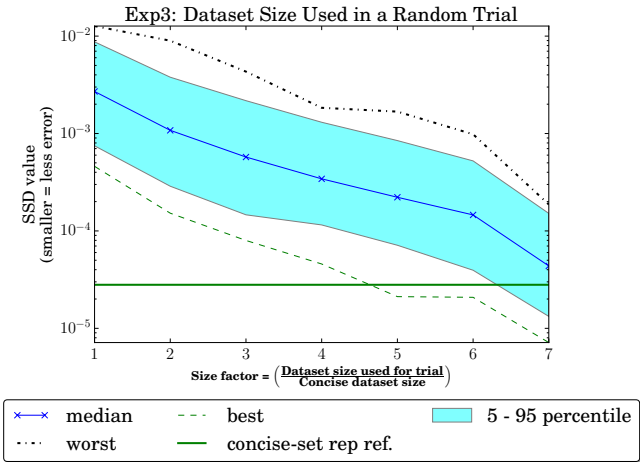

Fig. 14. Performance of randomly drawn datasets of different sizes. From the "best" performance curve, we see that a random dataset needs to be at least 4.7 times larger than the concise dataset in order to have any chance of outperforming the concise-set representation approach. Validation dataset size = 1000. Concise dataset size = 126. Number of trials with differently sized random datasets = 100. Ground-truth consistency = 0.997.

obtained through the concise-set representation approach. We then repeat the averaging experiment 100 times and plot the min, max, 5^{th}, 50^{th}, and 95^{th} percentiles curves. These percentile curves give us additional insights into the effectiveness of our concise-set representation approach.

As made evident by Fig. 13, the median (ie., 50^{th} percentile) curve in the figure indicates that taking the average of at least 80 random trials is needed to achieve a more accurate confusion matrix than the concise-set representation approach 50% of the time. We can draw similar conclusions regarding the best-case scenarios from the "best" curve. We see that in the best case, averaging of 10 trials are needed.

C. Concise-set Representation versus Random Datasets of Different Sizes

The goal of this experiment is to examine the effect of dataset size on the random sampling approach vis-a-vis the results obtained with our concise-set representation. In particular, we wish to investigate how large a randomly drawn dataset must be in order for it to outperform the concise-set representation. We repeat the experiment 100 times for each size and plot the values of the min, max, 5^{th}, 50^{th}, and 95^{th} percentiles as shown in Fig. 14. The figure indicates that for the random-sampling approach to have any chance of outperforming the concise-set representation, the size of a randomly drawn dataset should be at least 4.7 times larger than the concise dataset.

D. Larger Validation Dataset

In this experiment, we repeat Experiment VI-A but with a larger validation dataset. The result shown in Fig. 15 indicates that the concise-set representation approach still outperforms the traditional random sampling approach.

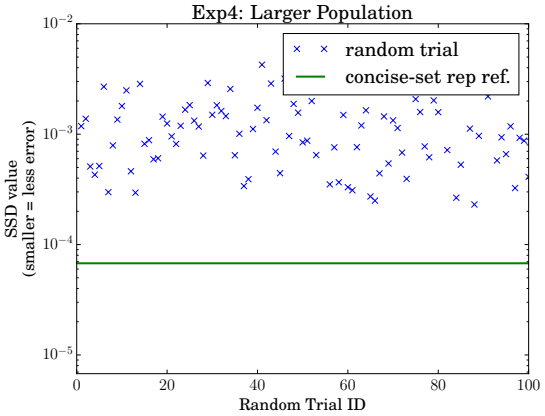

Fig. 15. With a larger validation dataset (10000 units instead of 1000), the concise-set representation approach continues to outperform the random sampling approach. Concise dataset size = 379. Random dataset size used for each of the 100 trials = 379. Ground-truth consistency = 0.9717.

E. Evaluations with Multiple Validation Datasets

To show that our approach works on other validation datasets, we repeat the last experiment (Section VI-D) on 5 validation datasets, each containing 10000 random samples drawn from the annotated pool described in Section VI.

Fig. 16 shows a bar chart consisting of five different validation datasets. The performance of each dataset is summarized by three SSD ratios calculated from Eq. 4 for K $\in [0, 5, 50]$:

$$\text{SSD Ratio} = \frac{\text{K-percentile(100 random trials)}}{\text{SSD from Concise-set rep.}} \tag{4}$$

Note that depending on the region, the minimum ratio can be close to 1.0. For example, in a region where not much diversity exists, the traditional random-sampling approach will do just as well as the concise-set representation approach. For another example, if the region is highly diverse, then the concise dataset will be larger, as more clusters are formed. In that case, the best performance by the random-sampling

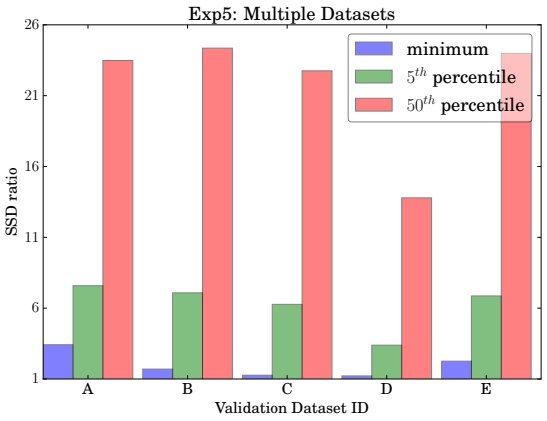

Fig. 16. Each group of three bars is a run of Exp4 (See Section VI-D) but on a different validation dataset. The SSD ratios are computed by Eq. 4. When the minimum ratio is above 1.0, it means that the concise-set representation approach is better than the random-sampling approach in all of the 100 random trials.

approach might improve due to the larger number of samples used in its random dataset, which has the same size as the concise dataset.

VII. Algorithm Complexity Analysis

To understand how well our system scales to Big Data, we must analyze all algorithms used in the system. In the next three sections, we will analyze the run-time complexity of three major algorithms we used in our system. These three sections will help us understand the current limitation and also provide potential future research directions.

A. Indexing the Dataset using Hyperplane LSH

In this section, we investigate the question pertaining to the cost of using Hyperplane LSH to index the dataset. Recall that the purpose of indexing the dataset is to support efficient neighborhood querying. Ideally, given an item, we could like to retrieve all of its neighbors in constant time. Two indexing methods that supports this fast retrieval are the brute-force method and LSH. The brute-force approach is not appropriate because the number of comparisons it takes to generate the index is a quadratic function of the size of the dataset. On the other hand, the number of comparisons for the LSH approach is linear in dataset size.

The cost of using LSH to index the dataset depends on the size of the dataset as well as the number of hashes needed to achieve the desired performance guarantee. When the number of hashes is large, due to desiring a high performance guarantee (See Section IV-E), the cost of indexing the dataset dominates the cost of data dimensionality reduction. Since the cost of indexing a dataset with Hyperplane LSH is linear in the number of data dimensions, a 100x reduction in data dimension translates to two order of magnitude in speedup.

To show it with complexity analysis, let d and k be the number of data dimensions before and after applying FastMap (See Section IV-D). Then, the time complexity of FastMap is $O(d\ k\ n)$, where n is the size of the dataset. On the other hand, the time complexity of indexing the dataset with Hyperplane LSH is $O(h\ d\ n)$, where h is the number of hashes per datum. The total running times for indexing a dataset with and without dimensional reduction is given in Eq. 7 and q. 5, respectively. The speedup effect due to the reduced data dimensionality is thus $\theta(\frac{d}{k})$. As a result, when we reduce the data dimensionality of the background color-histogram vectors from 9024 down to about 100, we can expect to get about two order of magnitude in speedup.

$$T_{\text{without dimensional reduction}} = O(h\ d\ n) \tag{5}$$

$$T_{\text{with dimensional reduction}} = O(d\ k\ n) + O(h\ k\ n) \tag{6}$$

$$= O(h\ k\ n) \quad \text{when } h \gg d \tag{7}$$

With regard to the the appropriateness of using LSH for large datasets, we note that when the dataset is small, indexing the dataset with LSH actually takes longer time than the brute-force indexing method. This is the case simply because $O(n)$ can grow faster than $O(n^2)$ for values of n below some threshold. For Hyperplane LSH, this threshold is $\theta(h)$. Therefore, in order to take advantage of Hyperplane LSH, the dataset must be bigger than h. Otherwise, we might as well use the brute-force indexing method. As an example, with the LSH parameters set to: $d_1 = 15$, $d_2 = 37.5$, desired $p_1 = 0.99$, and desired $p_2 = 0.001$, the value of h is 37278.

B. Sifting Through Neighbor Candidates in a LSH Bucket

In the previous section, we mentioned that LSH indexing has $O(h\ d\ n)$ time complexity. Here n is the size of the dataset, h is a function of the LSH performance guarantee, and d is the number of data dimensions. We stated that LSH indexing is linear in n because in our case d is typically less than 100 and h is a constant that does not change.

Because LSH is an approximated method, the collection of neighbor candidates inside the LSH bucket of a given input query may include false positives as well as duplicates. In the worst case, when every item is similar to all other items, it will take $O(n^2)$ time to remove false positives and duplicates from every neighborhood. This is the case because there are n neighborhoods, one per datum, and each neighborhood contains $O(n)$ neighbor candidates in the worst case.

C. Extracting the Dominating Clusters/Neighborhoods

As we mentioned in Section IV-F, the greedy solution for the Dominating Set problem runs in $O(|E|)$ time where E is the set of edges in the graph. In the worst case, when the graph is dense, $|E| = O(|V|^2)$ the greedy solution takes $O(|E|) = O\left(|V|^2\right)$ where V is the set of vertices in the graph. In our application, the number of vertices in the similarity graph is the size of the dataset. Therefore, extracting the dominating clusters will take $O\left(n^2\right)$ time to run in the worst case scenario. Here n is the size of the dataset.

D. Discussion

When the entire dataset is too big to be processed by a single computing unit. Our approach is employing Map-Reduce processing paradigm to distribute the computation across multiple

computing units. With this approach, we mitigate the run-time complexity issues mentioned in the previous sections.

When a $O(n^2)$ algorithm can not be avoid, our strategy is to keep n small enough such that the algorithm can still finish within a short time. If we break the entire dataset into small chunks/partitions, then each chunk can be processed quickly even though the complexity of the processing algorithm is quadratic in the worst case. Furthermore, if we process all chucks in parallel across many computing units, then the overall processing time is the time it takes to process a single chunk plus the time it takes to execute the subsequent merging step.

Our current Python implementation can handle chunk size of $n = 250000$. Further improvement in execution speed requires using optimized programming language such as C++ and replacing any quadratic-time algorithms with linear-time equivalents, if possible.

VIII. Conclusion

It can be mentally exhausting for human annotators to generate the ground truth needed for evaluating land-cover classifiers meant for large geographic regions covered by hundreds of satellite images. Human annotators end up wasting time by not realizing that new annotations may not be adding any additional discriminatory information to those already supplied. The work we have presented in this paper seeks to alleviate the annotation burden by reducing redundancies in the data through fast, albeit approximate, clustering by the LSH algorithm. Subsequently, the human is asked to annotate only the cluster representatives, in other words, the concise dataset, with one representative per cluster. Given the fairly wide range of the attributes derived from the spectral signatures that are used in land-cover classification, LSH may represent each land-type by hundreds of clusters. Nonetheless, providing annotations for the cluster representatives takes far less work than for the individual pixels in the satellite images that cover a wide area.

What adds to the power of our proposed approach is our demonstration that the approach is not overly sensitive to the choice of the parameters for the LSH algorithm. In our demonstration, we estimated the good values to use for the parameters from the satellite images that cover Chile and then used them to create a concise-set representation of the data in Australia.

We validated our approach through a comparison with traditional random-sampling based methods that are typically used for large datasets. We showed that for the same annotation effort, the concise-set approach to data representation outperforms the traditional random sampling approach in estimating the true confusion matrix.

Acknowledgment

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-C-7214. The U.S Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

We thank Noha Elfiky and German Holguin for their helpful comments and edits in the earlier drafts of the manuscript.

References

[1] T. Chang, B. Comandur, J. Park, and A. C. Kak, "A variance-based Bayesian framework for improving Land-Cover classification through wide-area learning from large geographic regions," *Computer Vision and Image Understanding*, vol. 147, pp. 3–22, 2016.

[2] A. C. Tamhane and D. D. Dunlop, *Statistics and Data Analysis from Elementary to Intermediate*. Upper Saddle River, NJ, USA: Prentice Hall, 2000.

[3] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[4] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima, "The A-tree: An index structure for high-dimensional spaces using relative approximation," in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 516–526.

[5] D. M. Mount, "ANN Programming Manual," https://www.cs.umd.edu/~mount/ANN/Files/1.1.2/ANNmanual_1.1.pdf, 2010.

[6] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *VLDB*, vol. 98, 1998, pp. 194–205.

[7] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 97–104.

[8] M. Izbicki and C. Shelton, "Faster cover trees," in *Proceedings of the Thirty-Second International Conference on Machine Learning*, 2015, pp. 1162–1170.

[9] A. C. Kak, "A Python implementation of Locality Sensitive Hashing for finding nearest neighbors and clusters in multidimensional numerical data," https://pypi.org/project/LocalitySensitiveHashing/, 2018.

[10] H. Xie, X. Tong, W. Meng, D. Liang, Z. Wang, and W. Shi, "A multilevel stratified spatial sampling approach for the quality assessment of remote-sensing-derived products," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4699–4713, 2015.

[11] J. Wickham, S. Stehman, J. Smith, T. Wade, and L. Yang, "A priori evaluation of two-stage cluster sampling for accuracy assessment of large-area land-cover maps," *International Journal of Remote Sensing*, vol. 25, no. 6, pp. 1235–1252, 2004.

[12] R. S. De Fries, M. Hansen, J. R. G. Townshend, and R. Sohlberg, "Global land cover classifications at 8 km spatial resolution: The use of training data derived from Landsat imagery in decision tree classifiers," *International Journal of Remote Sensing*, vol. 19, no. 16, pp. 3141–3168, 1998.

[13] C. Homer, C. Huang, L. Yang, B. Wylie, and M. Coan, "Development of a 2001 national land-cover database for the united states," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 7, pp. 829–840, 2004.

[14] K. Karantzalos, D. Bliziotis, and A. Karmas, "A scalable geospatial web service for near real-time, high-resolution land cover mapping," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4665–4674, 2015.

[15] N. C. Codella, G. Hua, A. Natsev, and J. R. Smith, "Towards large scale land-cover recognition of satellite images," in *International Conference on Information, Communications and Signal Processing (ICICS)*. IEEE, 2011, pp. 1–5.

[16] C. Jacqueminet, S. Kermadi, K. Michel, D. Béal, M. Gagnage, F. Branger, S. Jankowfsky, and I. Braud, "Land cover mapping using aerial and VHR satellite images for distributed hydrological modelling of periurban catchments : Application to the Yzeron catchment (Lyon , France)," *Journal of Hydrology*, vol. 485, pp. 68–83, 2013.

[17] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice Hall, 2006.

[18] A. C. Kak, "Constructing Optimal Subspaces for Pattern Classification," https://engineering.purdue.edu/kak/Tutorials/OptimalSubspaces.pdf, 2018.

[19] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[20] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[21] C. Faloutsos and K.-I. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," *SIGMOD Rec.*, vol. 24, no. 2, pp. 163–174, May 1995.

[22] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. ACM, 1998, pp. 604–613.

[23] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 128–131, March 2008.

[24] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 518–529.

[25] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011.

[26] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: ACM, 2002, pp. 380–388.

[27] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[28] A. Pujol and L. Chen, "Color quantization for image processing using self information," in *International Conference on Information, Communications Signal Processing*. IEEE, Dec 2007, pp. 1–5.

[29] A. Mathur and G. M. Foody, "Land cover classification by support vector machine: Towards efficient training," in *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, vol. 2. IEEE, 2004, pp. 742–744.