CGAP: A Hybrid Contrastive and Graph-based Active Learning Pipeline to Detect Water and Sediment in Multispectral Images

Bohan Chen*, Kevin Miller, Andrea L. Bertozzi *Member, IEEE*, Jon Schwenk

***Abstract*—We develop a contrastive graph-based active learning pipeline (CGAP) to identify surface water and near-water sediment pixels in multispectral images. CGAP enhances the graph-based active learning pipeline (GAP) (10.1109/IGARSS52108.2023.10282009), which outperforms methods such as CNN-Unet, support vector machine (SVM), and random forest (RF), while requiring less training data. Active learning plays an important role for training data reduction, resulting in an order of magintude less training data compared with conventional methods and three or more orders of magnitude less compared with CNN-Unet. Our improvements focus on boosting both the pipeline's robustness and efficiency by integrating a feature-embedding neural network prior to graph construction. This neural network, trained using contrastive learning, performs effective data dimension reduction by projecting high-dimensional raw features into a lower-dimensional space, thereby facilitating more efficient graph learning. The training process incorporates specialized augmentations to bolster the embedded features' resilience to geometric transformations, varying resolutions, and light cloud cover. Moreover, we develop a Python-based demo, GraphRiverClassifier (GRC), that uses the Google Earth Engine and our enhanced pipeline to provide a user-friendly tool for rapid and accurate surface water and sediment analyses and rapid testing of algorithm performances.**

***Index Terms*—Remote Sensing, Surface Water Detection, Graph Learning, Active Learning, Contrastive Learning**

I. Introduction

MAPPING surface water dynamics is crucial for a host of environmental, engineering, and management problems, including flood monitoring and mitigation, freshwater resource management, water quality analyses, and earth science research [1]–[4]. New technologies are needed to take advantage of the rise of global, remotely sensed surface water observations [5]–[7]. Rivers provide freshwater and ecologic resources that support human development, agriculture, and transportation worldwide. Many rivers are highly dynamic to environmental conditions [8], [9], and inferring these dynamics from remotely-sensed images has been a major focus in Earth Sciences over the previous decade [10]. The importance of automated surface water detection from remotely-sensed images is highlighted by significant efforts that have published global datasets or pre-trained models of surface water [6], [11], [12].

Global surface water datasets and models are effective at capturing the majority of surface water, yet often lack the local precision required for specific applications such as measuring river widths [13], [14] or estimating river migration rates [10], [15], where the targeted features may be as narrow as 1-2 pixels along the river boundary. To address this, researchers typically develop local models that involve labor-intensive manual data labeling and lack generalizability. For some river studies, an additional class representing in-channel sediment or highly-turbid water may be desired [10], [16]. In our case, we aim to identify rivers at their so-called "bankfull" state [17], which we define as the union of water and active (unvegetated) in-channel sediment bars [10].

Our work builds upon the RiverPIXELS dataset [18], a high-quality, hand-labeled collection of water and in-river, unvegetated sediment from Landsat multispectral images. Although RiverPIXELS covers various river types and environmental settings, it represents only a small fraction of global surface water. Therefore, our goal is to develop a robust, accurate, and efficient machine learning algorithm that can leverage the RiverPIXELS dataset to detect surface water and sediment pixels in multispectral satellite images on a global scale.

A. Related Work and Motivation

Several datasets and models exist for surface water detection. The Global Surface Water dataset [6] uses a Support Vector Machine (SVM) model [19] with post-processing to identify water pixels in Landsat images, accounting for confounding features like ice and snow. The USGS's Dynamic Surface Water Extent product [20] employs spectral mixing methods for Landsat imagery. Among pre-trained models, Deepwatermap [11] stands out, utilizing a convolutional neural network (CNN) U-net architecture [21] trained on extensive public datasets. While these approaches effectively capture

Bohan Chen is currently with the computing and mathematical sciences department at the California Institute of Technology. This work was done while he was at the University of California, Los Angeles, in the Department of Mathematics, and in residence at Los Alamos National Laboratory. Kevin Miller is with the Oden Institute for Computational Engineering and Sciences at the University of Texas, Austin and the Department of Mathematics at Brigham Young University, Provo. Andrea L. Bertozzi is with the Department of Mathematics, University of California, Los Angeles. Jon Schwenk is with Los Alamos National Laboratory.

BC was supported by the UC-National Lab In- Residence Graduate Fellowship Grant L21GF3606. KM is supported by the Peter J. O'Donnell Jr. Postdoctoral Fellowship. ALB is funded by NSF grants CCF-2345256 and DMS-2152717. JS is supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project numbers 20170668PRD1 and 20210213ER. ALB and JS are also supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under award DE-SC0025589 and under Triad National Security, LLC ('Triad') contract grant 89233218CNA000001 [FWP: LANLE2A2].

* Corresponding author (email: bhchen@caltech.edu).

overall surface water dynamics, they often lack the precision required for specific applications (particularly in labeling in- and near-channel sediment) and necessitate large amounts of training data, especially for state-of-the-art neural network models.

The identification of surface water from satellite images can be treated as an image segmentation problem, wherein labels are given to each pixel in an image such that pixels with the same label share certain characteristics. Image segmentation has been approached from many angles. There are partial differential equation (PDE)-based methods [22]–[24] for unsupervised segmentation, and deep learning methods like U-Net [21] for supervised segmentation on extensive annotated training datasets.

The RiverPIXELS dataset comprising 104 patches of 256 x 256 pixels, motivates us to explore semi-supervised methods capable of high performance with small training datasets. In this paper, we consider graph learning, a quintessential semi-supervised learning approach. This approach has proven successful in noisy image recovery [25]–[27], image or video segmentation [28]–[30], studies using remotely-sensed images to combine LIDAR and optical images [31], hyperspectral unmixing [32], [33], and SAR imagery classification [34], [35]. Additionally, graph convolutional networks (GCN) have been utilized for wetland classification, outperforming CNN models [36].

A general graph learning approach for image segmentation is based on a similarity graph generated from pixel features, with each pixel's feature vector serving as a node and the edge weights representing the similarity between nodes. To enhance the efficiency of graph learning, we previously introduced a graph-based active learning pipeline (GAP) [12], which does not require constructing a graph on the millions of pixels corresponding to the 104 patches in the RiverPIXELS dataset—a process that would be computationally inefficient. Instead, it employs an active learning approach [37], [38] to select representative samples from the training set. Our previous GAP method has outperformed SVM [19], Random Forest (RF) [39], and the CNN-based model DeepWaterMap [11].

B. Our Contributions

In this paper, we enhance our previously developed GAP by pre-training a feature-embedding neural network to improve both the method's performance and efficiency. Instead of using raw neighborhood patches as feature vectors for graph construction, we now preprocess them through this network. The feature embedding neural network is trained using the contrastive learning approach [40], [41], which has previously shown impressive results in SAR image classification when combined with graph learning [34]. Our proposed method is called the contrastive graph-based active learning pipeline (CGAP), with both a basic (B-CGAP) and adaptive (A-CGAP) version. The flowchart of the basic version (B-CGAP) is illustrated in Figure 1

Our main contributions are as follows:

1) We introduce a new CGAP method that combines our previous GAP approach with contrastive learning. Compared to previous methods, CGAP demonstrates better performance and higher efficiency and exhibits robustness to different resolutions and potential cloud cover.
2) We propose two versions of the CGAP method: basic (B-CGAP) and adaptive (A-CGAP). B-CGAP is based on a fully labeled training dataset, whereas A-CGAP allows us to train without any ground-truth label information initially. The A-CGAP process includes using active learning to guide a human-in-the-loop labeling process, requiring labels for only about 0.1% of the total pixels.
3) We provide a Python-based tool, GraphRiverClassifier (GRC), that leverages Google Earth Engine and Colab to deploy our pre-trained models to detect surface water and sediment within any Landsat image. This tool is highly flexible, mostly automated, and user-friendly.

The experimental results presented in this paper may be reproduced using the code available in our GitHub repository[1]. For a quick start with our tool, visit the GRC GitHub repository [2].

II. Background Information

This section reviews the methods integral to our pipeline (Section III-C), including contrastive learning, similarity graph construction, graph-based Laplace learning, and active learning.

A. Contrastive Learning

Contrastive Learning is a powerful technique for training deep neural networks, particularly for learning low-dimensional representations without extensive labeled datasets. At its core, contrastive learning aims to learn embeddings by maximizing the similarity between augmented views of the same data point while minimizing the similarity between embeddings of different data points. Chen et al. introduce a framework for contrastive learning of visual representations (SimCLR), which simplifies the contrastive learning paradigm by eliminating the need for specialized architectures or a memory bank [40]. This framework is further expanded in [42], demonstrating the effectiveness of large-scale self-supervised learning for improving semi-supervised learning performance. For a neural network f and a minibatch of m data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, each $\mathbf{x}_k$ is augmented into pairs $\tilde{\mathbf{x}}_{2k-1}, \tilde{\mathbf{x}}_{2k}$ which are processed through the neural network as $\mathbf{z}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1}), \mathbf{z}_{2k} = f(\tilde{\mathbf{x}}_{2k})$. The SimCLR loss is

$$\begin{aligned} \mathcal{L}_{\text{SimCLR}} &= \frac{1}{2m}\sum_{k=1}^{m}[\ell(2k, 2k-1) + \ell(2k-1, 2k)], \\ \ell(i,j) &= -\log \frac{\exp(g(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1, k\neq i}^{2m} \exp(g(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \end{aligned} \tag{1}$$

where τ is a constant parameter, and $g(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$ is the angular (cosine) similarity.

[1]https://github.com/wispcarey/CGAP-SurfaceWaterDetection
[2]https://github.com/wispcarey/GraphRiverClassifier

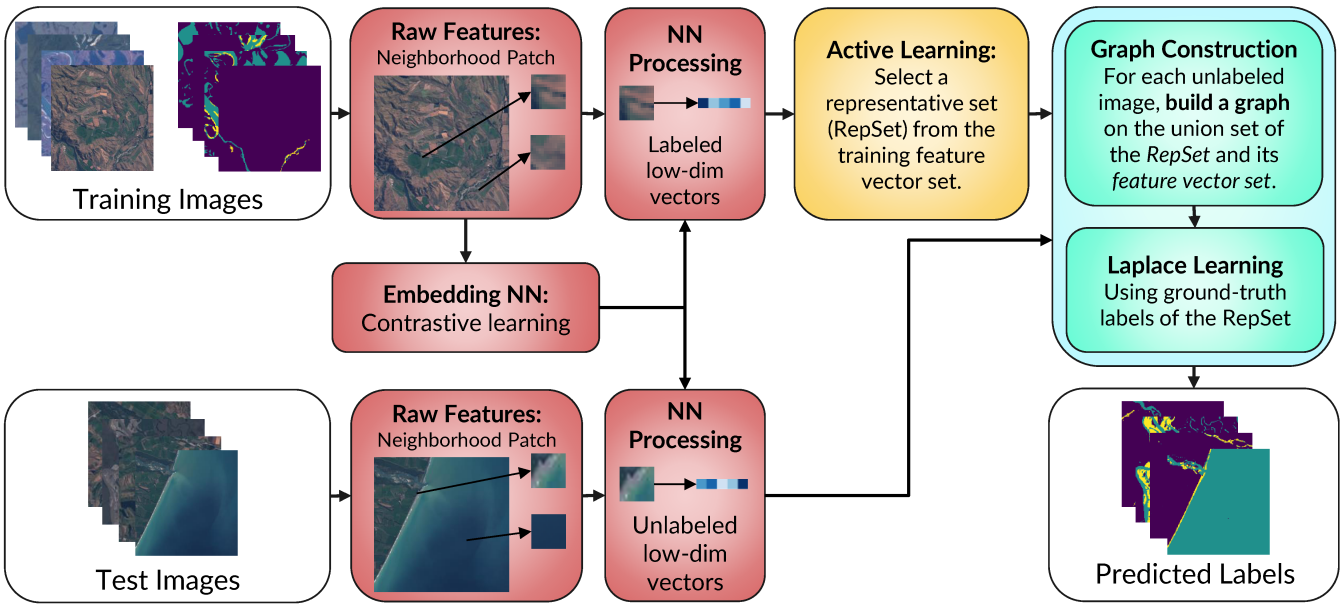


Fig. 1. The flowchart of our basic contrastive graph-based active learning pipeline (CGAP): 1. (Red Boxes) A neural network is trained with contrastive learning to convert raw images into feature vectors. 2. (Yellow Box) Condense the labeled feature vector set into a smaller representative set (RepSet) using active learning approaches. 3. (Cyan Box) Build a graph based on the union of the RepSet and the unlabeled feature set. Then, apply graph learning approaches to predict labels for unlabeled features.

Khosla et al. [41] extend SimCLR from self-supervised learning to supervised contrastive learning (SupCon) by leveraging label information. SupCon enhances the discriminative power of the learned embeddings by encouraging embeddings from the same class to cluster together, significantly outperforming traditional cross-entropy loss for supervised learning in many cases. In the supervised setting, we have the ground-truth labels $\{y_1, y_2, \ldots, y_{2m}\}$, $y_{2k-1} = y_{2k}$ for $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{2m}\}$. Based on (1), the supervised contrastive loss is defined by

$$\begin{aligned}\mathcal{L}_{\text{SupCon}} &= \sum_{i=1}^{2m} \frac{-1}{|P(i)|} \sum_{j \in P(i)} \log \frac{\exp(g(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1, k\neq i}^{2m} \exp(g(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \\ &= \sum_{i=1}^{2m} \frac{1}{|P(i)|} \sum_{j \in P(i)} \ell(i,j),\end{aligned} \tag{2}$$

where $P(i) = \{k \neq i : y_k = y_i\}$ is the set of indices with the same label of $\mathbf{z}_i$, and ℓ is defined in (1). The SimCLR loss (1) only considers augmented pairs $\mathbf{z}_{2k-1}, \mathbf{z}_{2k}$ while the SupCon loss (2) considers all pairs with the same label in the minibatch. In this work, we set $\tau = 0.5$ for both the SimCLR loss (1) and the SupCon loss (2) in the preprocessing step of our proposed pipelines (Section III-A) and our experiments (Section IV).

B. Graph Construction

We construct a graph $G = (X, W)$ to model the relationships within a dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$ of d-dimensional feature vectors. We consider X to be the set of vertices and construct a non-negative edge weight matrix $W \in \mathbb{R}^{N \times N}$. An edge between vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ $(i \neq j)$ exists if and only if the corresponding weight is non-zero (i.e., $W_{ij} > 0$). $W_{ij} > 0$ measures the similarity measure between two feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$W_{ij} = \exp\left(-\frac{\angle(\mathbf{x}_i, \mathbf{x}_j)^2}{\sqrt{\kappa_i \kappa_j}}\right), \tag{3}$$

where $\angle(\mathbf{x}_i, \mathbf{x}_j)$ represents the angular distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, computed by $\arccos\left(\frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)$. The normalization constant κ_i for each vertex $\mathbf{x}_i$ is determined by its angular distance to the Kth nearest neighbor, $\mathbf{x}_{i_K}$.

To facilitate efficient computations over the graph, we employ a sparse representation for W. Only connections between each vertex $\mathbf{x}_i$ and its K-nearest neighbors are considered. This is achieved using an approximate nearest neighbor search algorithm [43]. The K-nearest neighbors of $\mathbf{x}_i$, denoted as $\mathbf{x}_{i_k}$ for $k = 1, 2, \ldots, K$ (excluding $\mathbf{x}_i$ itself), are determined according to the corresponding angular distances. Consequently, a sparse version of W, denoted $\bar{W}_{ij}$, is defined by:

$$\bar{W}_{ij} = \begin{cases} W_{ij}, & \text{if } j = i_1, i_2, \ldots, i_K, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

It is important to select a large enough K to ensure the graph G is connected; in this paper, $K = 30$ is sufficient for our experiments. Furthermore, $\bar{W}$ is symmetrized as $W_{ij} := (\bar{W}_{ij} + \bar{W}_{ji})/2$ to guaranteeing that W is sparse and symmetric, possessing non-negative edge weights ($W_{ij} \geq 0$). In both the contrastive learning approach discussed in Section II-A and the graph construction methodology presented in this section, we utilize the angular distance as a similarity measure. This deliberate choice facilitates the natural application of the embedding features produced by contrastive learning in the graph-based Laplace learning classifier.

C. Graph Learning

With a graph $G = (X, W)$ constructed as described in the previous section, we now describe a graph-based approach for semi-supervised learning and present previous work in this field. Assume we have the ground-truth labels on a subset of vertices $X_l \subset X$. Let $y_i \in \{1, 2, \ldots, n_c\}$ be the ground-truth label of $\mathbf{x}_i \in X_l$, with corresponding one-hot vector $\mathbf{y}_i \in \mathbb{R}^{n_c}$. Important geometric information about the dataset X is encoded in graph Laplacian matrices [44], [45] of the graph G. Define $d_j = \sum_{\mathbf{x}_k \in X} W_{jk}$ to be the degree of node j and let D be the diagonal matrix with diagonal entries $d_1, d_2, \ldots, d_N$. While there are various scalings for graph Laplacians [45], we use the symmetric normalized graph Laplacian $L_{\text{sym}} := D^{-1/2}(D-W)D^{-1/2}$. Compared with the unnormalized graph Laplacian $L = D - W$, it is less sensitive to the size of the graph and has all eigenvalues in $[0, 2]$.

The inferred classification of unlabeled vertices comes from thresholding a continuous-valued node function $\mathbf{u} : X \to \mathbb{R}^{n_c}$. Specifically, the predicted label of $x_i \in X$ is $y_i = \arg\max\{\mathbf{u}_1(i), \mathbf{u}_2(i), \ldots, \mathbf{u}_{n_c}(i)\}$, where $\mathbf{u}_k(i)$ is the kth entry of $\mathbf{u}(i)$. Consider a $N \times n_c$ matrix U, whose i^{th} row is $\mathbf{u}(i)$; that is, each node function $\mathbf{u}$ can be identified by a matrix U whose i^{th} represents the output of $\mathbf{u}$ at node i. Our graph-based semi-supervised learning (SSL) model obtains the matrix U^* (and the corresponding node function $\hat{\mathbf{u}}$) by solving an optimization problem of the form:

$$\begin{aligned} U^* &= \underset{U \in \mathbb{R}^{N \times n_c}}{\arg\min} \frac{1}{2} \langle U, L_{\text{sym}} U \rangle_F, \\ &\text{s.t. } \mathbf{u}(x_i) = \mathbf{y}_i, \ \forall x_i \in X_l, \end{aligned} \tag{5}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product for matrices. This hard-constraint $\mathbf{u}(x_i) = \mathbf{y}_i, \ \forall x_i \in X_l$ forces the minimizer U^* to be the same as the ground-truth $\mathbf{y}$ on the labeled node set $X_l \subset X$. This SSL scheme was introduced in [46] and we refer to it as Laplace learning. We can reorder the vertices to write $U = \begin{bmatrix} U_l \\ U_u \end{bmatrix}$, where U_l corresponds to the submatrix of U whose rows correspond to the labeled set X_l and U_u similarly corresponds to the unlabeled set $X \setminus X_l$. Likewise, we can split the graph Laplacian matrix L_{sym} into labeled and unlabeled submatrices as

$$L_{\text{sym}} = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}. \tag{6}$$

As a result of the hard-constraint labeling of Laplace learning, the labeled part U_l^* of U^* is fixed as the one-hot encodings of the ground-truth labels on X_l; that is

$$U_l^* = \begin{bmatrix} \mathbf{y}_{i_1} \\ \mathbf{y}_{i_2} \\ \vdots \\ \mathbf{y}_{i_{|X_l|}} \end{bmatrix}, \quad i_1, i_2, \ldots, i_{|X_l|} \in X_l.$$

From the harmonic property of the optimized node function $\mathbf{u}^*$ (or U^*) of (5), U_u^* of Laplace learning can be calculated explicitly as

$$U_u^* = -L_{uu}^{-1} L_{ul} U_l^*. \tag{7}$$

D. Active Learning based on Uncertainty

Active learning is a technique to enhance the performance of underlying semi-supervised learning (SSL) techniques by strategically selecting which unlabeled data points to label, utilizing an oracle or human-in-the-loop approach. As mentioned above, we adopt Laplace learning [46] as the underlying semi-supervised classifier. When the labeled set size is fixed (i.e., $|X_l| = N_L$), the goal of active learning is to identify which $X_l \in X$ would be most beneficial for improving the classifier's performance. Alternatively, active learning can be viewed as a method for choosing a subset $X_l \subset X$ that (in some sense) optimally represents the geometric structure of the entire dataset X [35], [47].

Active learning implements an iterative approach to selecting a labeled set X_l from X. Starting with an initial labeled subset $X_l \subset X$ and a target final size N_L, the method proceeds by iteratively selecting a query set $\mathcal{Q} \subset X \setminus X_l$ and updating the labeled set as $X_l \leftarrow X_l \cup \mathcal{Q}$ in each round. The query set $\mathcal{Q}$ is selected according to an *acquisition function* $\mathcal{A} : X \setminus X_l \to \mathbb{R}$, which quantitatively evaluates how informative each unlabeled data point is to the learning model.

In this paper, we consider the smallest-margin uncertainty (UC) [48]–[50] acquisition function. Let $\mathbf{u}^*$ denote the current Laplace learning node function computed with the currently labeled set X_l. Then, the UC acquisition function is defined by:

$$\mathcal{A}_{\text{UC}}(\mathbf{x}_i) = 1 - \left(\mathbf{u}_{k_0}(\mathbf{x}_i) - \max_{k=1,2,\ldots,n_c; k \neq k_0} \mathbf{u}_k(\mathbf{x}_i) \right), \tag{8}$$

where $\mathbf{u}_k(\mathbf{x}_i)$ denotes the kth element of $\mathbf{u}(\mathbf{x}_i)$, and $k_0 = \arg\max_{j=1,2,\ldots,n_c} \mathbf{u}_k(\mathbf{x}_i)$.

For a given acquisition function, we consider a method for selecting a batch of query points (i.e., $|\mathcal{Q}| > 1$) from the set of acquisition function values on the unlabeled set $(\{\mathcal{A}_{UC}(\mathbf{x}_i)\}_{i \in X \setminus X_l})$ in each iteration of the active learning cycle. In contrast to most active learning methods, which sequentially selects $\mathcal{Q} = \{\mathbf{x}_k\}$ where $\mathbf{x}_k = \arg\max_{\mathbf{x}_i} \mathcal{A}(\mathbf{x}_i)$, we implement the LocalMax [35], [51] batch active learning strategy for selecting multiple query points per iteration. It selects a query set $\mathcal{Q} \subset X \setminus X_l$ comprising multiple unlabeled nodes that meet a local maximum criteria of the acquisition function $\mathcal{A}$ within the graph $G = (X, W)$. This batch approach accelerates the active learning sampling rate in proportion to the batch size, without significantly compromising model performance, as assessed by test accuracy in each iteration.

III. Surface water and sediment detection pipeline

This section introduces our contrastive graph-based active learning pipeline (CGAP). There are two versions of the CGAP method, namely the basic CGAP (B-CGAP), for training based on the fully labeled RiverPIXELS dataset [18], and the adaptive CGAP (A-CGAP) for additional unlabeled data to be labeled in a human-in-the-loop process using active learning. Consider the training image set $\mathcal{I} = \{I_1, I_2, \ldots, I_{n_l}\}$ and the test image set $\tilde{\mathcal{I}} = \{\tilde{I}_1, \tilde{I}_2, \ldots, \tilde{I}_{n_u}\}$. In training via the B-CGAP on $\mathcal{I}$, it is assumed that all ground-truth labels of pixels in $\mathcal{I}$ are available. In contrast, in training

via the A-CGAP, a priori knowledge of labels in $\mathcal{I}$ is not required. In both pipelines, we first preprocess the pixels in each image into feature vectors by training a feature-embedding neural network by contrastive learning. We then sample a training feature vector set, termed the representative set (RepSet) $\mathcal{R}$, as a subset of the entire preprocessed feature set $X = \{x_1, x_2, \ldots, x_N\}$. Each feature vector in $\mathcal{R}$ has its ground-truth label available (for A-CGAP, it could be obtained by human experts).

Our pipeline's core idea is to use this RepSet for classifying the pixels in each test set image in $\tilde{\mathcal{I}}$. For each test set image $\tilde{I}_j \in \tilde{\mathcal{I}}$ with N_0 pixels and corresponding unlabeled feature vector set $\tilde{X}_j = \{\tilde{\mathbf{x}}_1^j, \tilde{\mathbf{x}}_2^j, \ldots, \tilde{\mathbf{x}}_{N_0}^j\}$, we generate a weighted graph $G = (\mathcal{R} \cup \tilde{X}, W)$ according to section II-B and then apply Laplace learning (Section II-C) to classify the unlabeled feature vectors in $\tilde{X}_j$.

A. Contrastive Learning for Feature Preprocessing

For every image I within the set $\mathcal{I} \cup \tilde{\mathcal{I}}$, we extract a patch neighborhood centered around each pixel to form the raw feature vectors, considering all N_0 pixels in the image. The raw feature cube for each pixel is of the size $9 \times 9 \times 6$. Denote the set of raw feature cubes of I by $\{\mathbf{x}_1^{\text{raw}}(I), \mathbf{x}_2^{\text{raw}}(I), \ldots, \mathbf{x}_{N_0}^{\text{raw}}(I)\} \in \mathbb{R}^{9\times9\times6}$. Before training the feature embedding neural network, we design some transformations for those neighborhood patches of the size $k \times k$:

1) **Horizontal Flip**: Horizontally flip the patch.
2) **Vertical Flip**: Vertically flip the patch.
3) **Rotation**: Select a random angle and rotate the patch clockwise. Then crop the output into $k \times k$. This might introduce some zero values in the output patch.
4) **RandomPixelAugmentation**: Randomly select some pixels and set their values to 1 for all six channels.
5) **CenterCropResize**: Crop the image from the center based on a random scale and then resize the image to $k \times k$.
6) **Gaussian Reweight**: Reweight the patch by a $k \times k$ Gaussian kernel matrix.

Each of these six transformations is applied simultaneously to each channel of the patch. The final random augmentation design for contrastive learning consists of a sequential combination of these six transformations, where the first five are applied with a 50% probability of being executed, while the last transformation, Gaussian Reweight, is guaranteed to be applied. The first three transformations, Horizontal Flip, Vertical Flip, and Rotation, are standard goemetric image transformations. The fourth one, RandomPixelAugmentation, is designed for potential cloud coverage. The fifth one, CenterCropResize, is designed for the robustness of different (higher) resolutions. The last one, Gaussian Reweight, emphasizes pixels near the center and is inspired by the Non-local Means [52] model in image processing.

We use a shallow convolutional neural network for feature embedding. The network architecture is shown in Figure 2.

The loss function $\mathcal{L}(\theta)$ is the SupCon loss(2) for B-CGAP and is the SimCLR loss(1) for A-CGAP, since we don't have the ground-truth label information in the feature preprocessing stage of the A-CGAP. The respective training processes are described in Section II-A. Denote the trained parameters by $\hat{\theta}$. Finally, let the set of preprocessed feature vectors corresponding to image I be written as $\{\mathbf{x}_i(I) = f(\mathbf{x}_i^{\text{raw}}(I), \hat{\theta}) \mid i = 1, 2, \ldots, N_0\}$.

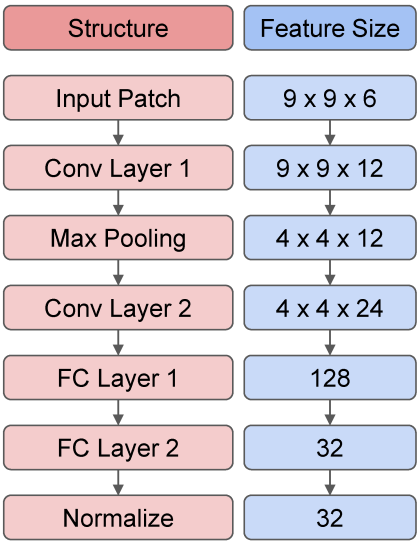


Fig. 2. The architecture of our feature embedding neural network which embeds $9 \times 9 \times 6$ neighborhood cubes into 32-dimensional feature vectors with 57836 trainable parameters. The **Conv Layer** denotes the convolutional layer and **FC layer** denotes the fully connected layer. There is a ReLU layer after each of **Conv Layer 1**, **Conv Layer 2**, and **FC Layer 1**. **Conv Layer 1** has a kernel size of 5 and padding 2 while **Conv Layer 2** has the a kernel size of 3 and padding 1. The final layer normalizes the L2 norm of the output feature vector to one, which makes it easier for the angular similarity in the loss function (1), (2) of contrastive learning.

B. Create the Representative Set

Including a massive set of all training feature vectors can be extremely inefficient in graph-based learning. For example, the label images of the Kolyma, Yana, Waitaki, and Colville Rivers in the RiverPIXELS dataset are comprised of 42 multispectral images of size $256 \times 256 \times 6$, consisting of over 2.7 million pixels. Generating a graph based on this extensive set of feature vectors would be prohibitively costly.

Denote X_i as the preprocessed feature vector set of the training image $I_i \in \mathcal{I}$. We downselect from $X = \cup_{i=1}^{n_l} X_i$ to create a much smaller RepSet $\mathcal{R} \subset X$ to be used to classify the pixels in the unlabeled images. Such a selection process removes redundant feature vectors while keeping significant feature vectors. We find a RepSet R_i for each image $I_i \in \mathcal{I}$ and take the union to obtain $\mathcal{R} = \cup_{i=1}^{n_l} R_i$. In the case that ground-truth labels are not available when generating the RepSet $\mathcal{R}$ (i.e. the A-CGAP case), pixels need to be labeled by human experts in a **human-in-the-loop** process. To determine the RepSet for a certain image $I_i \in \mathcal{I}$, we follow 3 steps: **Initialization**, **Active learning loop**, and **Termination**. Algorithm 1 provides an overview of such a process.

We now present the details of these 3 steps:

1) Initialization: Initialize the RepSet R_i^0.

There are two methods for initialization, **random initialization** and **core-set initialization**.

Algorithm 1 Create the Representative Set (RepSet) for Image I_i

INPUT: A set of labeled feature vectors X_i of Image I_i
OUTPUT: The RepSet with corresponding labels.
GRAPH CONSTRUCTION (Section II-B): Graph $G_i = (X_i, W_i)$.
INITIALIZATION: Initialize $R_i = R_i^0$ with corresponding labels.
WHILE Terminal Conditions are not triggered:
  Apply Laplace learning [46] on G_i with labels of R_i to classify the remaining feature vectors in $X_i \setminus R_i$.
  Calculate the Uncertainty acquisition function [48]–[50] value for each feature vector in $X_i \setminus R_i$.
  Sample a query set $Q \subset X_i \setminus R_i$ via the LocalMax [35] batch active learning approach.
  Obtain the labels of the query set Q.
  Update the RepSet $R_i \leftarrow R_i \cup Q$
END WHILE
RETURN: The RepSet $R_i \subset X_i$ and corresponding labels.

In the **random initialization**, feature vectors in the initial RepSet R_i^0 are randomly chosen from the set X_i. If all ground-truth labels of feature vectors in X_i are available, we can randomly sample the same number of feature vectors within each class to achieve a class-balanced initialization. If all ground-truth labels for the current image are *not available*, then the randomly sampled initial set will be highly imbalanced across labels. In many images in our dataset, land pixels account for over 80% of the image while sediment pixels account for less than 5%. Random initialization likely yields little to no sediment pixels, exacerbating class imbalances in the subsequent **active learning loop**.

The **coreset initialization** select a core-set that follows the geometric distribution of the feature vector set X_0. Here we use the Dijkstra Annulus Core-Set (DAC) [35], which provides a good initialization for the graph-based active learning process. Compared with the **random initialization**, it is not efficient since it needs to construct a graph structure on X_i and down-sample according to this graph.

2) Active learning loop: Following the guidance of active learning approaches II-D, add feature vectors from X_i and corresponding labels one by one to the RepSet.

The **Active learning loop** is illustrated within the WHILE section of Algorithm 1: Construct a graph G_i on X_i and initialize the RepSet to be R_i^0. Apply Laplace learning on G_i with the initially labeled set R^0 to make predictions for feature vectors in $X_i \setminus R^0$. Then based on the predicted labels, calculate the acquisition function $\mathcal{A}(\mathbf{x})$ for $\mathbf{x} \in X_i \setminus R^0$ according to (8). Then select a query set $\mathcal{Q}$ with the given batch size $|\mathcal{Q}| = B$ according to the LocalMax batch sampling approach [35]. Update the RepSet as $R_i^1 = R_i^0 \cup \mathcal{Q}$. Repeat this process for each iteration t until reaching a certain terminal condition, to be explained in the next bullet entitled **Termination**. The final R_i^t then is the RepSet of image I_i, denoted as R_i.

3) Termination: Stop the active learning loop when a certain terminal condition is satisfied. When ground-truth labels of all feature vectors are available, the **accuracy-based terminal condition** can be applied. Otherwise, the **label-change terminal condition** is applied.

In iteration t of the **active learning loop** step for image I_i, let X_i be the preprocessed feature set of pixels in I_i and R_i^t be the RepSet in the current iteration. We apply Laplace learning on the graph built with nodes X_i with labels on R_i^t to make predictions on $X_i \setminus R^t$. Let $\mathcal{Q}$ be the query set to obtain labels by active learning and $R_i^{t+1} = R_i^t \cup \mathcal{Q}$. The labels are either (1) already available if the training set images are fully labeled or (2) hand-labeled by a human in the loop. Denote the Laplace learning prediction label on $\mathbf{x} \in X_i$ at iteration t by $y^t(\mathbf{x})$.

With a hyperparameter $K_{\max}, \epsilon$, these terminal conditions are based on the predicted labels on $X \setminus R_i^t, X \setminus R_i^{t+1}$ at iterations $t, t+1$. Define the δ-function:

$$\delta(x, y) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{9}$$

We present two kinds of terminal conditions to check if we need to terminate the process at iteration $t+1$:

1) **Accuracy-based terminal condition:** If the ground-truth label $y(\mathbf{x})$ is *available* for feature vector $\mathbf{x} \in X_i$, we can terminate the active learning loop according to the change in prediction accuracy. The accuracy at iteration t is calculated by:

$$a_t = \frac{\sum_{\mathbf{x} \in X_i \setminus R_i^t} \delta(y^t(\mathbf{x}), y(\mathbf{x}))}{|X_i \setminus R_i^t|}. \tag{10}$$

Terminate the active learning loop if:

$$|a_t - a_{t+1}| < \epsilon \text{ or } t > K_{\max}. \tag{11}$$

Practically, we further penalize low accuracy by applying a lower ϵ when a_{t+1} is relatively small. Given a fixed parameter γ, we use the terminal condition:

$$|a_t - a_{t+1}| < \epsilon \exp\left(-\frac{100(1 - a_{t+1})}{\gamma}\right) \text{ or } t > K_{\max}. \tag{12}$$

2) **Label-change terminal condition:** If the ground-truth labels for feature vectors are *not available*, we can terminate the active learning loop according to the change of predicted labels. At iteration t, define the label-change value:

$$c_t = \frac{\sum_{\mathbf{x} \in X_i \setminus R_i^t} \delta(y^t(\mathbf{x}), y^{t-1}(\mathbf{x}))}{|X_i \setminus R_i^t|}. \tag{13}$$

Terminate the active learning loop if:

$$c_{t+1} < \epsilon \text{ or } t > K_{\max}. \tag{14}$$

Algorithm 2 Contrastive Graph-based Active Learning Pipelines (B-CGAP & A-CGAP)

INPUT: Training image set $\mathcal{I}$, Test image set $\tilde{\mathcal{I}}$
OUTPUT: Predicted labels for all pixels in $\tilde{\mathcal{I}}$
PREPROCESSING & ACTIVE LEARNING:
  If all ground-truth labels of the $\mathcal{I}$ are available (Use ***B-CGAP*** – Condensing Task):
    Train feature embedding neural network using ***supervised contrastive loss*** (2) (Sections II-A, III-A)
    Extract feature vectors X_i for each $I_i \in \mathcal{I}$ and $\tilde{X}_j$ for each $\tilde{I}_j \in \tilde{\mathcal{I}}$ (Section III-A)
    For each $I_i \in \mathcal{I}$:
      (Section III-B)
      Condense RepSet R_i from X_i using ***class-balanced random initialization*** and ***accuracy-based terminal condition***

    Construct RepSet $\mathcal{R} = \cup_{i=1}^{n_l} R_i$
  Else (Use ***A-CGAP*** – Human-in-the-loop Labeling):
    Train feature embedding neural network using ***SimCLR contrastive loss*** (1) (Sections II-A, III-A)
    Extract feature vectors X_i for each $I_i \in \mathcal{I}$ and $\tilde{X}_j$ for each $\tilde{I}_j \in \tilde{\mathcal{I}}$ (Section III-A)
    For each $I_i \in \mathcal{I}$:
      (Section III-B)
      Condense RepSet R_i from X_i using ***coreset initialization*** and ***label-change terminal condition***
    Construct RepSet $\mathcal{R} = \cup_{i=1}^{n_l} R_i$
TESTING:
  For each $\tilde{I}_j \in \tilde{\mathcal{I}}$:
    Extract feature vectors $\tilde{X}_j$ for $\tilde{I}_j$ using the pretrained feature embedding neural network (Section III-A)
    Construct graph $G_j = (\mathcal{R} \cup \tilde{X}_j, W_j)$ (Section II-B)
    Apply Laplace learning on G_j to get classifier matrix U_j and corresponding node function $\mathbf{u}_j$ (Section II-C)
    For each $\tilde{x}_k^j \in \tilde{X}_j$:
      Predict label $\tilde{y}_k^j = \arg\max \mathbf{u}_j(\tilde{x}_k^j)$
RETURN: Predicted labels for all pixels in $\tilde{\mathcal{I}}$

The accuracy-based terminal condition provides a clearer measure of how well a RepSet R_i^t is for the task of classifying the pixels in image I_i but requires the ground-truth labels for all pixels.

On the other hand, the label-change terminal condition does not require ground-truth labels so that one can create the RepSet from scratch. This terminal condition is met when the addition of labeled points to the current image's RepSet R_i^t changes the predicted labels of the unlabeled points $X \setminus R_i^t$ from Laplace learning at a very slow rate. In a sense, this gives a measure of when there is no significant marginal gain for adding more labeled points to the RepSet. Such a process requires manually labeling only a few feature vectors to construct a RepSet.

The effectiveness of our representative set is closely tied to the active learning process which uses graph Laplace learning as a semi-supervised classifier. The active learning process selects a RepSet that represents well both the clustering structure and the difficult decision boundary regions [35], [37], [53] in each training image. By selecting a RepSet of limited size that performs well on each of the training set images, we directly verify the effectiveness of the RepSet for the pixel classification task. Therefore, as long as the testing set of images is not too dissimilar to the training set, our proposed framework ensures efficient and accurate results. In addition, for our B-CGAP method, the active learning process reduces redundancy in the set of pixels that the human in the loop must label to maximize information gain with minimal labeling effort. In practice, we can adjust the parameters in the terminal condition to balance the effectiveness and efficiency of the active learning process.

Our experiments in the next section suggest that the size of the resulting RepSet is usually less than 1% of the original set of pixels in the training image set. Moreover, our pipeline consistently outperforms a range of fully-supervised methods in multiple experiments, despite such a small labeled dataset.

C. Pipeline Structure

This subsection presents an overview of our proposed B-CGAP and A-CGAP approaches. Both pipelines are designed to classify pixels in a test image set $\tilde{\mathcal{I}}$ based on training conducted on a training image set $\mathcal{I}$. Algorithm 2 describes these pipelines, highlighting that the primary differences between B-CGAP and A-CGAP lie in the preprocessing and active learning stages, while the testing phase remains identical. The key distinctions are emphasized in ***bold italic*** within the algorithm.

At the outset of both pipelines, we preprocess each image in $\mathcal{I}$ and $\tilde{\mathcal{I}}$ into sets of feature vectors, as detailed in Section III-A. For any image $I_i \in \mathcal{I}$, we denote its extracted feature set as $X_i = x_1^i, x_2^i, \ldots, x_{N_0}^i$. Similarly, for each test image $\tilde{I}_j \in \tilde{\mathcal{I}}$, we represent its feature set as $\tilde{X}j = \tilde{x}_1^j, \tilde{x}_2^j, \ldots, \tilde{x}^j N_0$. Here, $N_0 = 256^2 = 65536$ is the number of pixels in each image, constant across all images in both sets. The following paragraphs provide more details about both pipelines.

The B-CGAP requires all ground-truth labels in $\mathcal{I}$ in the training process. Figure 1 presents the flowchart of the B-CGAP. We train the feature embedding neural network with the **supervised contrastive loss** (2). For each $I_i \in \mathcal{I}$, since all ground-truth labels of feature vectors in its feature set X_i are given, the RepSet R_i can be condensed from X_i with the class-balanced **random initialization** and the **accuracy-based terminal condition** according to III-B. The RepSet of $\mathcal{I}$ is $\mathcal{R} = \cup_{i=1}^{n_l} R_i$. For each test image $\tilde{I}_j \in \tilde{\mathcal{I}}$ with feature set $\tilde{X}_j$, construct a graph $G_j = (\mathcal{R} \cup \tilde{X}_j, W_j)$, where $\mathcal{R} \cup \tilde{X}_j$ is the vertex set of size $N_j = |\mathcal{R} \cup \tilde{X}_j|$ and W_j is the weight matrix generated from feature vectors in $\mathcal{R} \cup \tilde{X}_j$ according to Section (II-B). Apply Laplace learning on G_j to get the classifier matrix $U_j \in \mathbb{R}^{N_j \times n_c}$ according to Section II-C. Let $\mathbf{u}_j$ be the corresponding node function of the classifier matrix U_j. The predicted label $\tilde{y}_k^j$ of each unlabeled feature vector $\tilde{x}_k^j \in \tilde{X}_j$ is given by

$$\tilde{y}_k^j = \arg\max \mathbf{u}_j(\tilde{x}_k^j),\ k = 1, 2, \ldots, N_0, \tag{15}$$

where the $\arg\max$ of a vector $\mathbf{u}$ is the (first) index of $\mathbf{u}$'s largest element.

In addition, we include an extension of the B-CGAP, called the adaptive CGAP (A-CGAP). The training process of the A-CGAP does *not* require a priori access to the ground-truth labels for all pixels in each training image $I_i \in \mathcal{I}$. We apply the **SimCLR contrastive loss** (1) to train the feature embedding network. The creation of RepSet requires the **coreset initialization** and the **label-change terminal condition** according to III-B to sample RepSets R_i for $I_i \in \mathcal{I}$. With the aid of a human in the loop during this process, feature vectors in R_i are manually labeled in the active learning process. Let $\mathcal{R} = \cup_{i=1}^{n_l} R_i$. *The rest steps of the A-CGAP are the same as the B-CGAP.* We construct a graph on $\mathcal{R} \cup \tilde{X}_j$ and use graph Laplace learning to classify feature vectors in $\tilde{X}_j$.

Note that the A-CGAP provides flexibility for applications where there is no predefined training image set $\mathcal{I}$, but rather just a test image set $\tilde{\mathcal{I}}$. In such a case, the A-CGAP extracts a RepSet $\tilde{\mathcal{R}}$ of $\tilde{\mathcal{I}}$ from scratch and applies Laplace learning with the human-in-the-loop labels for $\tilde{\mathcal{R}}$ to classify the other pixels in $\tilde{\mathcal{I}}$. Furthermore, A-CGAP could be applied to reinforce the B-CGAP. Suppose one has access to all ground-truth labels for pixels in $\mathcal{I}$. In that case, we can apply the B-CGAP, and the A-CGAP to respectively extract the RepSets $\mathcal{R}$ and $\tilde{\mathcal{R}}$ of $\mathcal{I}$ and $\tilde{\mathcal{I}}$. Let $\mathcal{R}_{\text{new}} = \mathcal{R} \cup \tilde{\mathcal{R}}$ be the new RepSet. For an image $\tilde{I}_j \in \tilde{\mathcal{I}}$, applying Laplace learning on the graph with vertices $\mathcal{R}_{\text{new}} \cup \tilde{X}$ with labeled set $\mathcal{R}_{\text{new}}$ to classify feature vectors in $\tilde{X} \setminus \mathcal{R}_{\text{new}}$. Such a process allows us to expend limited human-in-the-loop effort to expand our labeled feature set (RepSet) by including some feature vectors from the test image set.

IV. Experiments and Results

This section includes experiments of both methods on the RiverPIXELS dataset. We choose five rivers from the dataset: the Kolyma, Yana, Waitaki, Colville, and Ucayali Rivers. We train on images chosen from the first four rivers while the algorithms are tested on all five rivers. There are 42 images belonging to the first four river regions and the Ucayali River includes 54 images. For each region, we randomly sample 75% of the labeled data as the training set and use the remaining 25% as the test set. The training set $\mathcal{I}$ has 32 labeled images while the test set $\tilde{\mathcal{I}}$ has 10 labeled images, which are considered unlabeled in our experiments. In Section IV-B, the test set image set $\tilde{\mathcal{I}}_{\text{ex}}$ is formed by 54 images of the Ucayali river.

Various metrics are provided to evaluate the performance of different methods and schemes. Each unlabeled image $\tilde{I}_j \in \tilde{\mathcal{I}}$ has M^2 pixels $\{p_{k,l}^j\}_{k,l=1}^M$ with ground-truth labels $\{y_{k,l}^j\}_{k,l=1}^M$ and predicted labels $\{\bar{y}_{k,l}^j\}_{k,l=1}^M$, where $M = 256$. We define d_b to be the distance to the boundary for each pixel $p_{k,l}^j$ of coordinate k, l of image $\tilde{I}_j$ by

$$d_b(p_{k,l}^j) = \min_{\{(\hat{k},\hat{l}): y_{k,l}^j \neq y_{\hat{k},\hat{l}}^j\}} \sqrt{(k-\hat{k})^2 + (l-\hat{l})^2}, \tag{16}$$

which is the Euclidean distance to the nearest pixel with a different ground-truth label.

For the test set $\tilde{\mathcal{I}}$ (or $\tilde{\mathcal{I}}_{\text{ex}}$) with the size $|\tilde{\mathcal{I}}| = n_u$, we define following metrics:

1) **Overall Accuracy:** The overall accuracy is the average accuracy of all pixels.

$$\text{Overall} = \frac{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} \delta(y_{k,l}^j, \bar{y}_{k,l}^j)}{n_u M^2} \tag{17}$$

2) **Class Accuracies:** We consider the true positive rate (TPR), false positive rate (FPR), and the normalized false positive rate (NFPR) of each class. For class index c:

$$\text{TPR}(c) = \frac{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} \delta(y_{k,l}^j, c)\delta(\bar{y}_{k,l}^j, c)}{\sum_{j=1}^{n_t} \sum_{k,l=1}^{M} \delta(y_{k,l}^j, c)} \tag{18}$$

$$\text{FPR}(c) = \frac{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} \delta(\bar{y}_{k,l}^j, c)(1-\delta(y_{k,l}^j, c))}{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} (1-\delta(y_{k,l}^j, c))}. \tag{19}$$

3) **Boundary Accuracy** The boundary accuracy of distance d is the average accuracy of pixels whose distance to the boundary is less or equal to d.

$$\text{BA}(d) = \frac{\sum_{j=1}^{n_u} \sum_{\{(k,l): d_b(p_{k,l}) \leq d\}} \delta(y_{k,l}^j, \bar{y}_{k,l}^j)}{\sum_{j=1}^{n_u} |\{(k,l) : d_b(p_{k,l}) \leq d\}|} \tag{20}$$

The land, water, and sediment classes in both our selected images and in general are imbalanced, with land pixels accounting for 70% to 90% of each image. Therefore, a naive classifier that tends to simply classify pixels primarily as land may still report an excellent overall accuracy. Furthermore, a method will rarely have both the best TPR and FPR for every single class – i.e. the best performance in each of the class accuracies metrics. We suggest that the boundary accuracy metric is the most indicative of model performance.

We compare B-CGAP and A-CGAP to various other methods, such as our previous graph-based active learning pipeline (GAP) [12], without contrastive learning, and DeepWaterMap (DWM) [11], support vector machine (SVM) [19] and random forest (RF) [39]. After feature preprocessing (Section III-A),

TABLE I
INFORMATION ON TRAINING AND TEST DATASETS

Method	***Original Set***		***Network Embedding***	***Sampled Training Set***		***Test Set***	
	Dataset	*Num Images*		*Dataset*	*Num Pixels*	*Dataset*	*Num Images*
B-CGAP (Ours)	$\mathcal{I}$	$\lvert\mathcal{I}\rvert = 32$ (2.1M Pixels)	SupCon [41]	RepSet	3.71K	$\tilde{\mathcal{I}}$ for Section IV-A	$\lvert\tilde{\mathcal{I}}\rvert = 10$ (650K Pixels)
A-CGAP (Ours)			SimCLR [40]	RepSet	2.99K		
GAP [12]			No	RepSet	3.27K		
SVM [19]			No	$\mathcal{P}$	42.6K	$\tilde{\mathcal{I}}_{\text{ex}}$ for Section IV-B	$\lvert\tilde{\mathcal{I}}_{\text{ex}}\rvert = 54$ (3.5M Pixels)
SVM-E [19]			SupCon [41]	$\mathcal{P}$	42.6K		
RF [39]			No	$\mathcal{P}$	42.6K		
RF-E [39]			SupCon [41]	$\mathcal{P}$	42.6K		
DWM-R [11]			No	$\mathcal{I}$	2.1M	No test set for Sections IV-C, IV-D	
DWM-O [11]	$\mathcal{I}_{\text{DWM}}$	100K	No	$\mathcal{I}_{\text{DWM}}$	6.55B		

Information of training and test datasets of different methods implemented in Section IV. Methods implemented here are our B-CGAP, A-CGAP, previously proposed GAP, SVM, RF, retained DWM (DWM-R), and original DWM (DWM-O). The suffix "-E" of SVM and RF corresponds to using the neural network embedding features. The training set of DWM-O is the original training set of DeepWaterMap while other methods use training sets sampled from RiverPIXELS. K, M, and B refer to a thousand, million, and billion respectively.

TABLE II
COMPARISON APPROACH: RETRAIN DEEPWATERMAP
ALL VALUES IN PERCENTAGE (%)

Importance	***3rd***						***1st***		***2nd***
Method	***Land***		***Water***		***Sediment***		***Boundary***		***Overall***
	TPR	*FPR*	*TPR*	*FPR*	*TPR*	*FPR*	*BA(3)*	*BA(10)*	*OA*
B-CGAP (ours)	98.57	5.63	92.29	1.59	78.49	**0.75**	**88.85**	**92.29**	**96.61**
A-CGAP (ours)	**99.02**	11.25	86.93	**1.36**	64.28	**0.77**	84.51	90.35	95.35
GAP [12]	98.27	8.43	89.89	1.98	65.72	0.86	81.90	90.41	95.50
SVM [19]	95.09	6.20	89.74	4.10	82.89	1.75	79.28	88.05	93.55
SVM-E [19]	91.78	**0.63**	93.84	6.61	**96.70**	2.42	78.52	85.26	92.38
RF [39]	93.72	2.65	92.26	5.10	90.73	2.14	77.40	86.84	93.31
RF-E [39]	91.81	0.95	**94.27**	6.94	92.49	2.06	80.23	86.32	92.38
DWM-R [11]	97.38	14.53	83.99	3.12	41.70	1.08	72.56	84.56	92.86

The comparison among different methods trained as 3-class classifiers of the land, water, and sediment. This table compares our B-CGAP, A-CGAP, previously proposed GAP, SVM, RF, and retrained DWM (DWM-R). SVM and RF include both the non-local means feature vectors and the neural network embedding feature vectors (-E). Accuracy metrics include the true positive rate (TPR), false positive rate (FPR) of each class, the boundary accuracy of distances 3 and 10 (BA(3), BA(10)), and the overall accuracy (OA). The first row "importance" indicates the important ranking of three different types of accuracy metrics. The best one of each accuracy metric (each column) is bolded. Our B-CGAP performs the best on boundary accuracies and the overall accuracy.

TABLE III
COMPARISON APPROACH: MODIFY LABELS (SED → LAND)
ALL VALUES IN PERCENTAGE (%)

Importance	***3rd***						***1st***		***2nd***
Method	***Land***		***Water***		***Sediment***		***Boundary***		***Overall***
	TPR	*FPR*	*TPR*	*FPR*	*TPR*	*FPR*	*BA(3)*	*BA(10)*	*OA*
B-CGAP(ours)	98.58	8.27	91.73	1.42	N/A	N/A	**89.73**	**93.66**	**97.03**
A-CGAP(ours)	**99.04**	14.80	85.20	**0.96**	N/A	N/A	85.46	91.17	95.90
GAP [12]	98.58	11.47	88.53	1.42	N/A	N/A	83.75	91.66	96.30
SVM [19]	97.73	12.49	87.51	2.27	N/A	N/A	82.77	91.08	95.41
SVM-E [19]	94.09	6.10	93.90	5.91	N/A	N/A	81.67	87.80	94.04
RF [39]	96.37	9.37	90.63	3.63	N/A	N/A	81.50	89.63	95.07
RF-E [39]	93.44	**5.93**	**94.07**	6.56	N/A	N/A	82.67	88.18	93.58
DWM-O [11]	97.85	14.26	85.74	2.15	N/A	N/A	78.02	88.81	95.11

The comparison among different methods trained as 2-class classifiers of the land, and water. To compare with the original DeepWaterMap (DWM-O), we changed all ground-truth labels of sediment into land. This table compares our B-CGAP, A-CGAP, previously proposed GAP, SVM, RF, and retrained DWM (DWM-R). Accuracy metrics include the true positive rate (TPR) and false positive rate (FPR) of each class, the boundary accuracy of distances 3 and 10 (BA(3), BA(10)), and the overall accuracy (OA). The first row "importance" indicates the important ranking of three different types of accuracy metrics. The best one of each accuracy metric (each column) is bolded. Our B-CGAP performs the best on boundary accuracies and overall accuracy. *It is a coincidence that the B-CGAP and GAP have the same land TPR and Water FPR.*

the original extracted feature vector set X has over 2 million feature vectors from the training set consisting of 32 labeled images. Each method has a different amount of training data, chosen to represent the best performance of each method. The training and test datasets information for each method are listed in Table I.

We now present the **training set details** for each method. For SVM and RF, the training performances are almost the same when the number of training feature vectors is relatively large. To balance the training performance and the computational cost of model fitting, we randomly select a subset of all labeled pixels to train SVM and RF. For each labeled image in the training set $\mathcal{I}$, we randomly sample N_s pixels from each class (if a class has less than 500 pixels, sample all) to form a pixel set $\mathcal{P}$. The pixel set $\mathcal{P}$ includes 42634 pixels consisting of 16000, 14558, and 12076 pixels for land, water, and sediment respectively.

For GAP, B-CGAP, or A-CGAP, the training set for the graph Laplace learning is the RepSet $\mathcal{R}$ extracted from X using LocalMax batch active with batch size $B = 15$. For B-CGAP and GAP, the accuracy terminal condition (10)(12) is applied with the $\epsilon = 10^{-4}$, $K_{\max} = 3000$ (for each training image), and $\gamma = 5$ (for B-CGAP only). For A-CGAP, the label-change terminal condition (13)(14) is applied with $\epsilon = 5 \times 10^{-4}$, $K_{\max} = 3000$ (for each training image). To clarify, the active learning process is only applied to our graph-based methods to reduce graph size in B-CGAP and the previous method GAP, and to reduce human labeling requirements in A-CGAP. The other methods for comparison utilize much larger training sets (Table I).

DeepWaterMap [11] only provides the classification of water and land pixels, while RiverPIXELS patches include water, bare sediment, and land. Here the "land" does not follow a strict definition, which can include complicated ground textures like buildings, mountains, and forests. In binary classification results, such as DWM, "land" refers to non-water pixels. In 3-class classification results including the sediment, such as our pipeline, "land" refers to non-water and non-sediment pixels. As a result, we cannot directly compare our three-class model with DWM. Here, we provide two approaches to compare the performance of the other methods and DWM.

The first approach is to *retrain DeepWaterMap* (DWM-R). We train a new neural network with the same structure of DWM on our training set with 32 labeled images and labels of water, sediment, and land. The resulting CNN thus provides a classification of water, sediment, and land pixels. The second approach is to *modify labels*. Inspection of the original DeepWaterMap (DWM-O) training set shows that nearly all sediment pixels are labeled as land. We modify the labels of our training set and the ground-truth labels of our test set by changing sediment labels to land labels. Based on this modification, we train all methods as classifiers for water and land and compare them with the original DeepWaterMap.

In light of these details regarding DWM, we consider two types of training sets for DWM comparisons. DWM-R is trained on the training image set $\mathcal{I}$ that all of our other comparison methods are trained on. DWM-O is trained on the vast training set of the original DeepWaterMap [3]. Furthermore, note that the pixel-wise feature vectors for different methods might differ. B-CGAP uses the supervised contrastive learning (SupCon (2)) neural network embedding feature vectors while A-CGAP uses the unsupervised feature vectors (SimCLR (1)), according to Section II-A, III-A. We also provide experiments with SVM and RF results, marked by the suffix "-E", on the SupCon feature vectors. The GAP, SVM, and RF are based on the 294-dimensional Non-local means feature vectors [12], [52] generated by 7×7 neighborhood patches centering at each pixel. For DWM, the inputs are images rather than feature vectors since DWM is a CNN-based method that takes the whole image as input.

A. Comparison between different methods

We compare the classification performance of both our B-CGAP and A-CGAP to GAP [12], DWM [11], SVM [19] and RF [39] models. Information regarding training and test sets is shown in Table I. For this subsection (Subsection IV-A), the test set is denoted as $\tilde{\mathcal{I}}$ and contains 10 images. It is worth noting that our methods, B-CGAP, A-CGAP, and GAP, use **significantly less** training data than the other methods considered. The results are presented in two tables: Table II shows the comparison to the retrained DWM (DWM-R) as a 3-class classifier, while Table III compares to the original DWM (DWM-O) when the sediment class in the RiverPIXELS dataset is modified to be classified as land.

Figures 3, 4 and 5 are sampled images and experiment results of the Waitaki, Colville, and Yana Rivers, respectively.

In summary, according to Tables II, III, our approach B-CGAP has the best performance measured by BA(3), BA(10), and OA. Our A-CGAP and the GAP we proposed in our previous paper [12] perform similarly and have the second good result. According to Table I, compared with other methods, all three methods, B-CGAP, A-CGAP, and GAP, are trained on a much smaller training set, which only takes around 0.15% pixels of the training set $\mathcal{I}$. Although A-CGAP performs similarly to our previous pipeline, GAP, A-CGAP does not require any ground-truth information at the beginning of the training process.

Here we provide an analysis of the overall strengths of the methods.

1) Comparison with DWM: In Table II, the retrained DWM (DWM-R) has the worst performance in both BA(3) and BA(10). This implies that our training set $\mathcal{I}$ with 32 images is not sufficient to train a good CNN-UNet. In Table III, the original DWM (DWM-O) has a similar performance to the SVM and RF. However, it cannot provide the prediction of the sediment class.

2) The neural network embedding: The main difference between our B-CGAP and the previous GAP is the network embedding. According to Tables II, III, there is a big improvement from GAP to B-CGAP. However, the SVM-E and RF-E perform worse than SVM and RF, respectively, based on the same SupCon embedding used by B-CGAP. There are significant increases in the water and sediment TPR and FPR

[3]https://github.com/isikdogan/deepwatermap

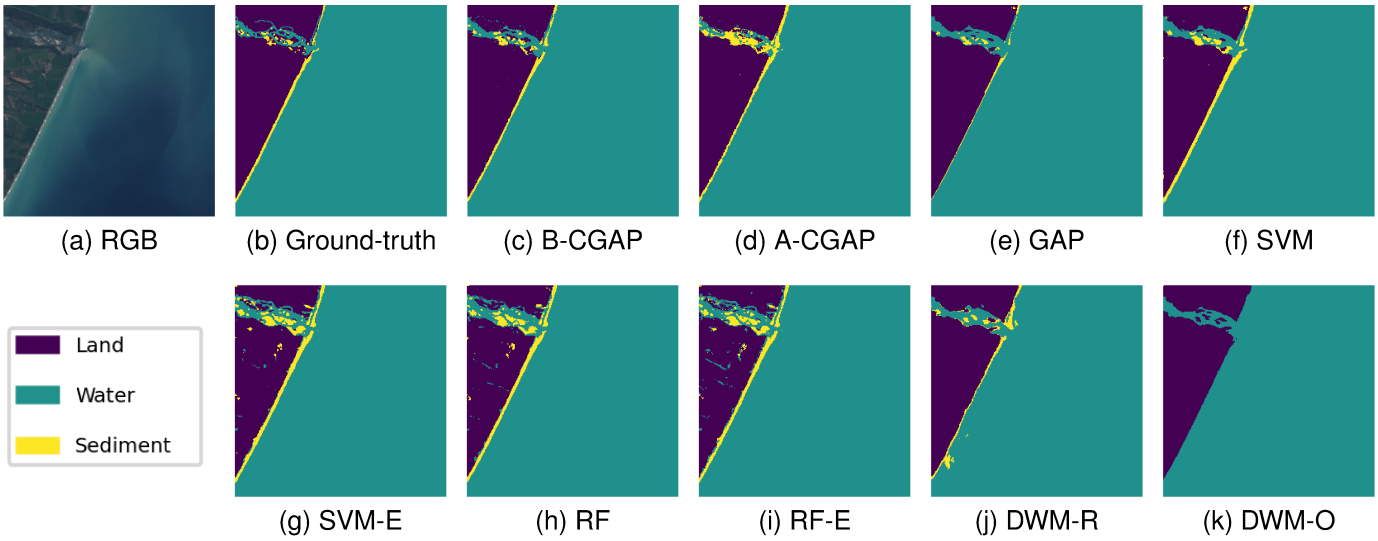


Fig. 3. Results for a Patch of Waitaki river. Original Patch name: *Waitaki_River_1 2019-03-02 074 091 L8 413 landsat*. This Patch contains an estuary and a coastline. Panel (k) is the original DWM prediction for water and non-water pixels while other panels (b)-(j) are 3-class results of land, water, and sediment.

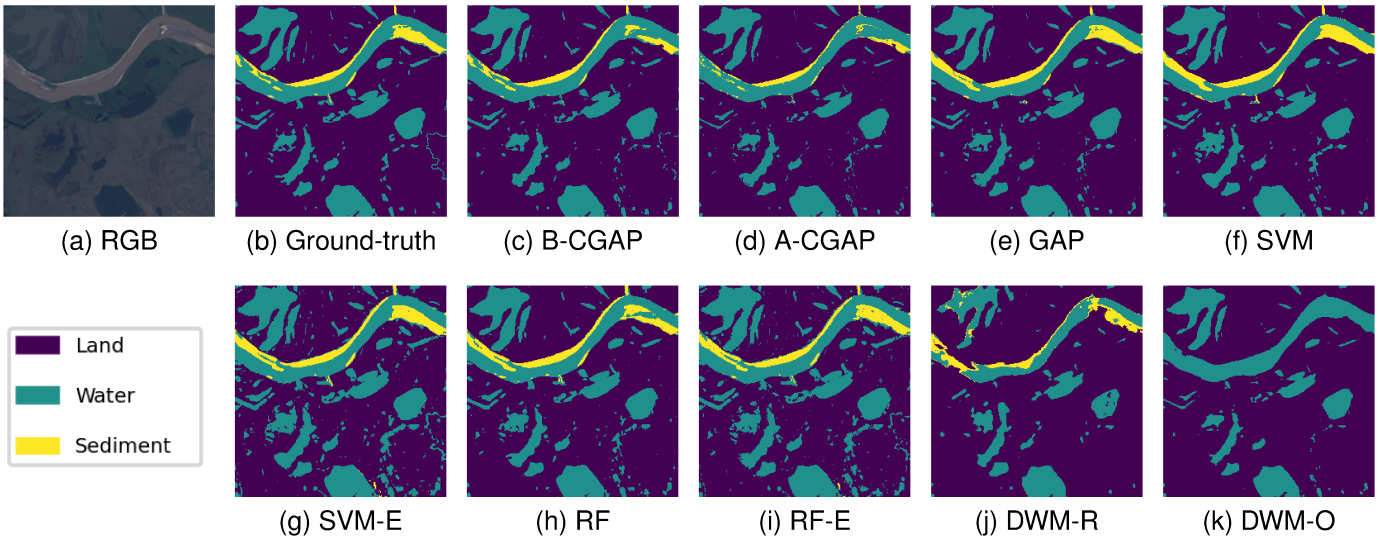


Fig. 4. Results for a Patch of the Colville River. Original Patch name: *Colville_River_2 2015-07-11 076 011 L8 125 landsat*. This Patch contains a complex network of water, including a mainstream, some lakes and small tributaries. Panel (k) is the original DWM prediction for water and non-water pixels while other panels (b)-(j) are 3-class results of land, water, and sediment.

when using the SupCon network embedding feature vectors. A possible reason for this observation is that the graph learning classifier based on the angular similarity aligns better with the similarity metric in the contrastive loss functions (1), (2).

3) Comparison of B-CGAP v.s. A-CGAP: Our A-CGAP does not require any ground-truth information at the beginning of the training process (Section III-C). One major difference is the training process of the feature embedding neural network—A-CGAP uses the self-supervised loss SimCLR(1) while B-CGAP uses the supervised loss SupCon(2). In A-CGAP, only the 2.99K training feature vectors in the RepSet require ground-truth labeling, which can be processed by a human-in-the-loop process.

B. Performance on other regions

In the previous Section IV-A, all methods are trained on subsets of $\mathcal{I}$, and the performances are evaluated on the test set $\tilde{\mathcal{I}}$. We note that $\mathcal{I}$ and $\tilde{\mathcal{I}}$ are segmented from the same set, the Kolyma, Yana, Waitaki, and Colville rivers. For a certain image $\tilde{I}_j \in \tilde{\mathcal{I}}$, there is another image in $I_i \in \mathcal{I}$ that belongs to the same region as $\tilde{I}_j$, where the region refers to either the Arctic or New Zealand. Since DeepWaterMap works globally, we want to test if our models also retain accuracy in other regions that may have different landscapes or geographic features.

We consider images of the Ucayali River in our RiverPIXELS dataset. The Ucayali River is a tributary to the Amazon River and is mostly single-threaded with large in-channel bare sediment bars. We apply the same comparison strategies as

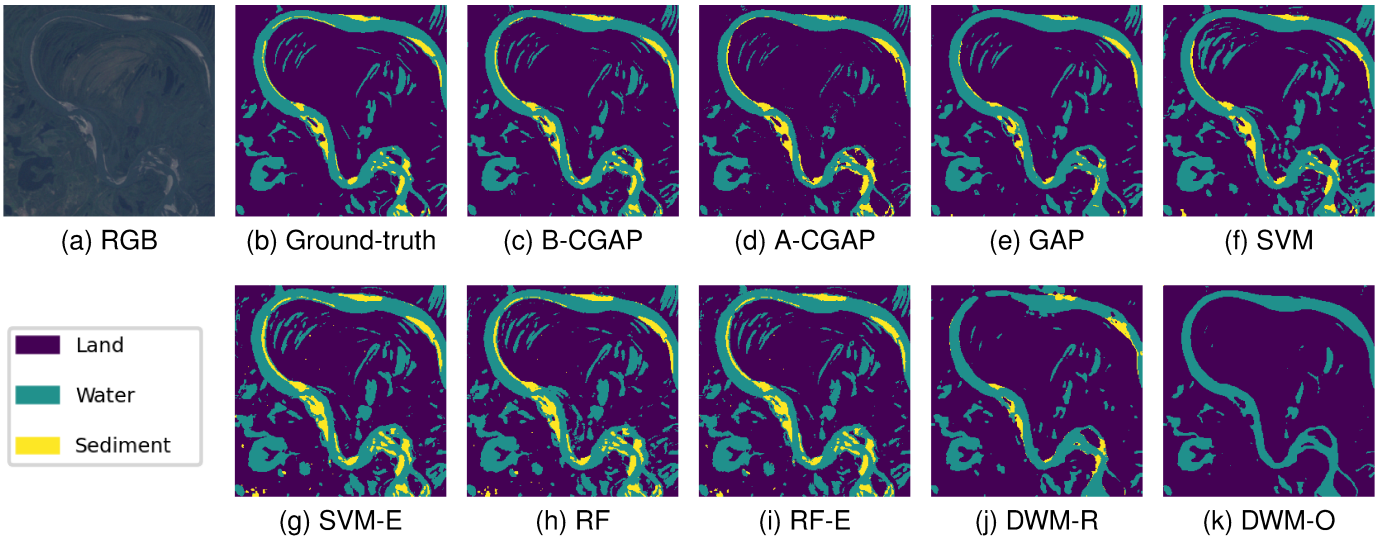


Fig. 5. Experiment on images of Yana river. Original image name: *Yana_River_1 1991-08-13 122 012 L5 511 landsat*. This Patch contains a complex network of water, including a mainstream, some lakes and small tributaries. Panel (k) is the original DWM prediction for water and non-water pixels while other panels (b)-(j) are 3-class results of land, water, and sediment.

in section IV-A. The training information of each method is the same as that in Section IV-A, Table I, while the test set now is the set of images of the Ucayali river, which includes 54 images. Table IV compares our B-CGAP and A-CGAP to GAP, SVM, RF, the retrained DWM (DWM-R) and original DWM (DWM-O) on the test set $\tilde{\mathcal{I}}_{\text{ex}}$. Unlike Tables II and III, we have not included comparisons with SVM-E, and RF-E methods here for brevity. This decision is based on previous detailed analyses in Section IV-A: SVM-E and RF-E underperformed compared to their direct counterparts due to the mismatch of the embedding neural network. Similarly to Section IV-A, we provided results on 3 classes of land, water, and sediment, and 2 classes of water and non-water by modifying the sediment labels into the land, to compare with both the DWM-R and DWM-O. According to Table IV, our B-CGAP has the best performance as we have the highest BA(3), BA(10), and overall accuracy in both comparisons. A-CGAP performs similarly to GAP while has a higher BA(3).

We sample two images of the Ucayali river and show those results in figures 6 and 7. According to these figures and tables, the random forest method (RF) completely fails, implying that it may be unstable when applied to a region different from its training set. In Figure 6, our B-CGAP detects the small tributaries better than other methods. In Figure 7 with light cloud haze, both our B-CGAP and previous GAP method (panel (c), (d)) match well to the ground-truth labels (panel (b)).

C. Computational Efficiency Analysis

From the previous section, we conclude that B-CGAP has the best performance followed by A-CGAP and GAP. Now we provide information on the time consumption of three methods, B-CGAP, A-CGAP, and GAP, in both training and deploying (see Table V). With the low-dimensional feature vectors preprocessed by the neural network (B-CGAP and A-CGAP), the active learning and model deploying processes are significantly accelerated. The model deploying is **more than 10 times faster** compared with our previous GAP method. Here we train both the B-CGAP and A-CGAP preprocessing network on the whole RiverPIXELS dataset of 104 images to make a fair comparison. In previous Sections IV-A,IV-B, B-CGAP is trained only on the training set $\mathcal{I}$ of 32 images since it is supervised. Since the whole dataset includes 6.8M pixels and the dataset $\mathcal{I}$ includes 2.1M pixels, we use part of them for the network training. Practically, we use all the sediment pixels, randomly sampled 20% water pixels, and randomly sampled 3% land pixels to train the neural network, which is a subset of 0.48M pixels. As for the hyperparameters, we choose the constant parameter $\tau = 0.5$ in the loss functions (1),(2), learning rate 0.02, and batch size 2048 to train both networks for 200 epochs.

D. Low-dimensional Visualization of Feature Vectors

Here we provide more details on how the feature embedding neural network changes the feature vectors. Figure 8 shows the low-dimensional visualization of different feature vectors using UMAP [54] and t-SNE [55] methods. The pixels shown in the panels are downsampled from the entire RiverPIXELS dataset, which originally contains millions of pixels with an imbalanced number of pixels per class. To ensure clarity, each panel only includes a low-dimensional visualization of 59,475 pixels' feature vectors, with approximately 20,000 pixels per class. In Figure 8, panels (a) and (d) are visualizations of raw neighborhood patches of pixels. We observe approximately three cluster structures in (a) and (d), but there is a clear mix and overlap at the center and between the boundaries of each pair of classes. For the embeddings produced by SimCLR (panels (b), (e)), the cluster structures are more defined, with clearer distinctions between boundaries. However, the clusters are a bit dispersed, for instance, the water class (cyan) is divided into three smaller clusters in both visualizations. The

TABLE IV
EXPERIMENTS ON IMAGES OF THE UCAYALI RIVER

Method	***Retrain DWM***(%)			***Sed→Land***(%)		
	BA(3)	*BA(10)*	*OA*	*BA(3)*	*BA(10)*	*OA*
B-CGAP (ours)	**90.63**	**95.29**	**98.31**	**91.57**	**95.40**	**98.60**
A-CGAP (ours)	86.47	93.11	97.60	88.35	93.97	98.06
GAP [12]	83.07	92.93	97.48	85.08	94.17	98.21
SVM [19]	79.08	90.29	96.26	77.49	90.96	97.23
RF [39]	26.41	26.79	12.52	76.01	89.09	96.21
DWM-R [11]	69.28	83.43	94.39	N/A	N/A	N/A
DWM-O [11]	N/A	N/A	N/A	79.42	91.14	97.29

This table shows the comparison of our B-CGAP and A-CGAP to GAP, SVM, RF, DWM-R, and DWM-O. Accuracy values in this table are based on the extra test set $\tilde{\mathcal{I}}_{\text{ex}}$ consisting of 54 images of the Ucayali river while methods in this table are trained on the training set $\mathcal{I}$ of Arctic and New Zealand. More details of the training set refer to Table I. Metrics are the boundary accuracy of distances 3 and 10 (BA(3), BA(10)), and the overall accuracy. The best one of each accuracy metric (each column) is bolded. To compare with the retrained DWM (DWM-R) and the original DWM (DWM-O), we provide results on 3 classes (columns "Retrain DWM") and 2 classes (columns Sed→Land). Our B-CGAP performs the best on boundary accuracies and overall accuracy.

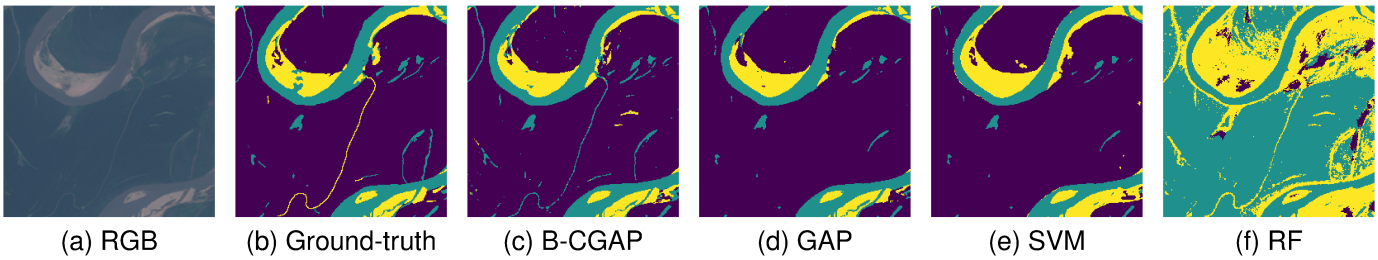


Fig. 6. Results for the Ucayali River. Original Patch name: *Ucayali_River_1 2018-09-11 006 066 L8 549 landsat*. This Patch includes two mainstreams and some small tributaries. Purple, cyan, and yellow represent land, water, and sediment respectively.

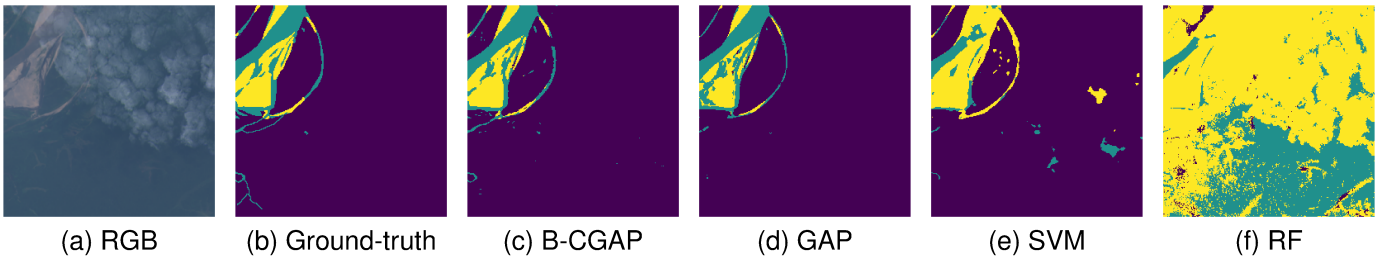


Fig. 7. Results from a Patch of the Ucayali River. Original Patch name: *Ucayali_River_1 2018-09-11 006 066 L8 316 landsat*. This patch includes some light clouds. Purple, cyan, and yellow represent land, water, and sediment respectively.

TABLE V
TIME CONSUMPTION

Method	***Info***		***Training***				***Deploying***		
	CPU	*GPU*	***Feature Embedding NN***		***Active Learning***		*Image Size*	*Preprocessing NN Embedding*	*Graph Learning*
			Num Epochs	*Total Time*	*Batch Size*	*Total Time*			
B-CGAP (ours)	Intel(R) i7-11800H	Nvidia RTX3070 Laptop	200	14.34h	15	2023.21s	256×256	0.448s	10.76s
A-CGAP (ours)			200	13.87h		1710.55s		0.448s	11.15s
GAP [12]			N/A	N/A		6881.73s		N/A	135.31s

This table shows the time consumption for our B-CGAP, A-CGAP, and previous GAP. The neural network training and deploying stages use the GPU, and all other processes are on the CPU. Although the neural network training takes a relatively long time, it reduces both the active learning time and model deploying time significantly.

low-dimensional visualization by the SupCon method is the best, with the clearest boundaries between classes and each class generally forming a cohesive cluster.

V. GRAPHRIVERCLASSIFIER: A GLOBAL CLASSIFIER FOR WATER AND SEDIMENT PIXELS IN SATELLITE IMAGES

We provide a Python-based demo, called GraphRiverClassifier (GRC), to classify any Landsat-5 image (GitHub repository[4]). For instructions on how to use the demo please refer

[4]https://github.com/wispcarey/GraphRiverClassifier

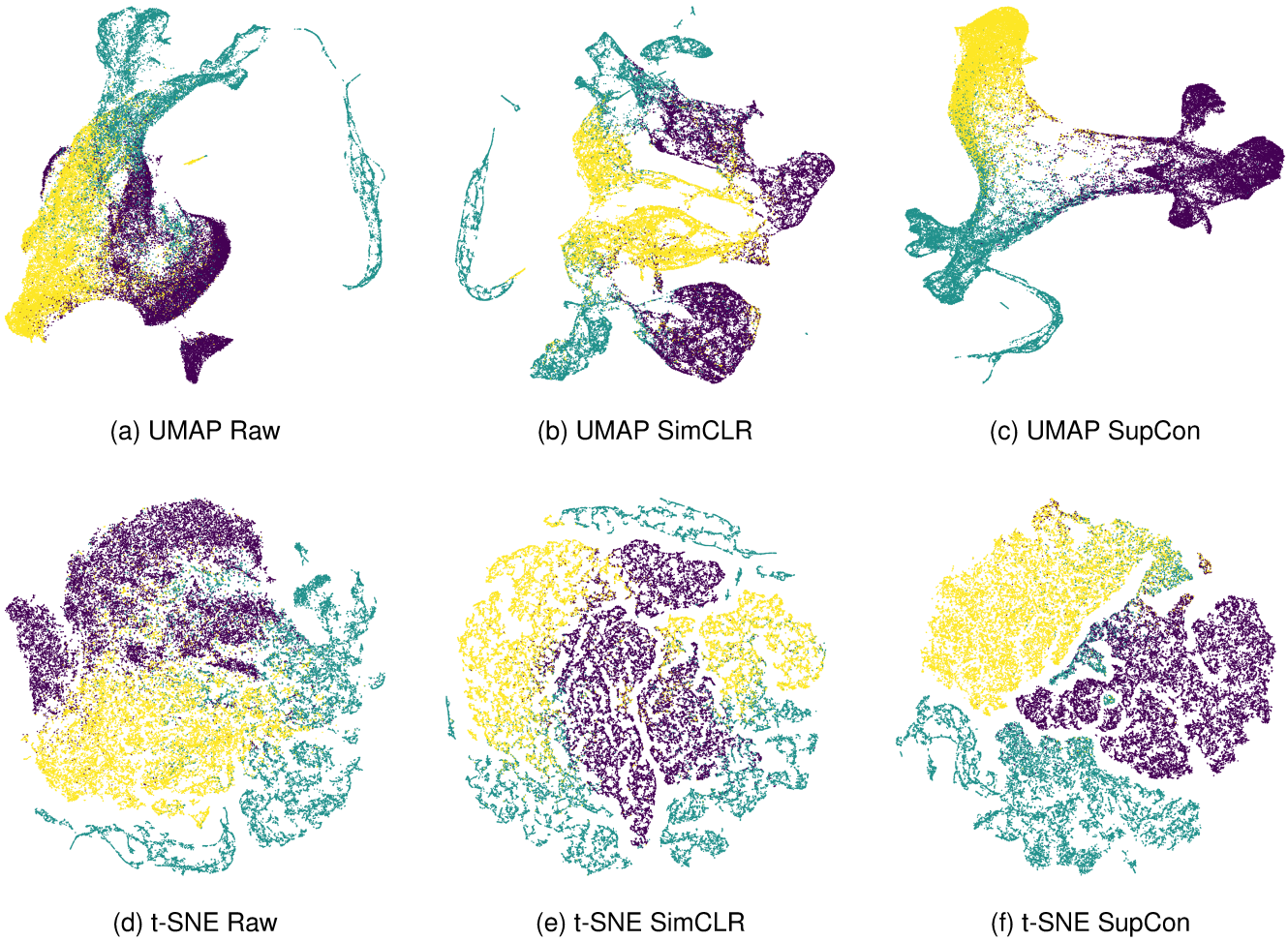


Fig. 8. Low-dimensional visualizations of feature vectors using UMAP [54] (panels (a)-(c)) and t-SNE [55] (panels (d)-(f)). We compare low-dimension representations of the raw neighborhood patch, SimCLR, and SupCon feature vectors of pixels in the whole RiverPIXELS dataset. Purple, cyan, and yellow represent land, water, and sediment respectively. Each panel is a low-dimensional visualization of feature vectors of 59,475 pixels that are downsampled from the entire RiverPIXELS dataset, with approximately 20,000 pixels per class.

to the `readme.md` file in the repository. In this section, we describe some features of this tool.

A. Google Platforms

The tool described is implemented in Python and utilizes Google Earth Engine (GEE) [56] [5] for identifying and downloading the requested Landsat scenes. Users can perform the image extraction by simply providing the coordinates of the center point along with the desired latitude and longitude range of the bounding box. The selection method for Landsat scenes (e.g. surface reflectance versus top of atmosphere) is fully consistent with the RiverPIXELS dataset [18], to which their readme can be referred for further information.

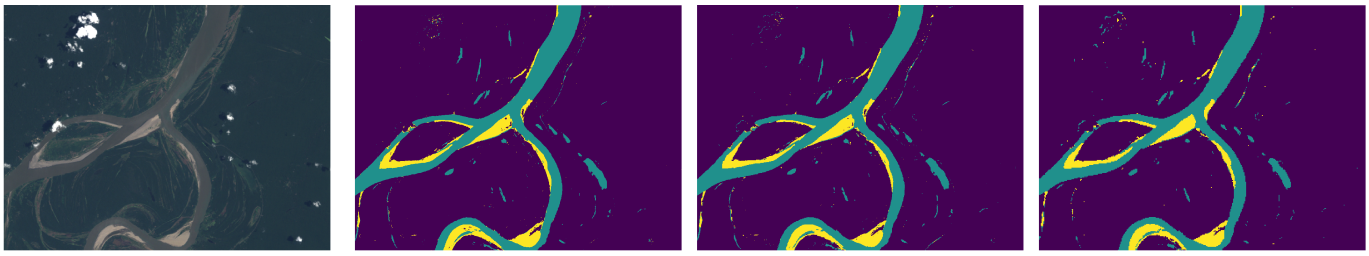
Once Landsat scenes are identified, the tool automatically employs a pre-trained feature embedding neural network to preprocess feature vectors. Subsequently, our B-CGAP method is deployed to classify the pixels within the image into land, water, and sediment. The neural network training and the selection of RepSet through graph-based active learning are based on the RiverPIXELS dataset.

It is recommended that the tool be run on Google Colab as it allows for a seamless connection with GEE and ensures the automation of the entire process. Running the tool via a Google Colab notebook also avoids large image downloads by reading directly from a GEE-connected Google Drive account. Users may run the tool locally, but Landsat scene identification and downloading must be done manually. The tool also offers an integration with ChatGPT that allows users to enter the name of a place or river to return a bounding box. However, this feature requires access to ChatGPT's API and may incur costs.

B. Global Classifier of Water and Sediment

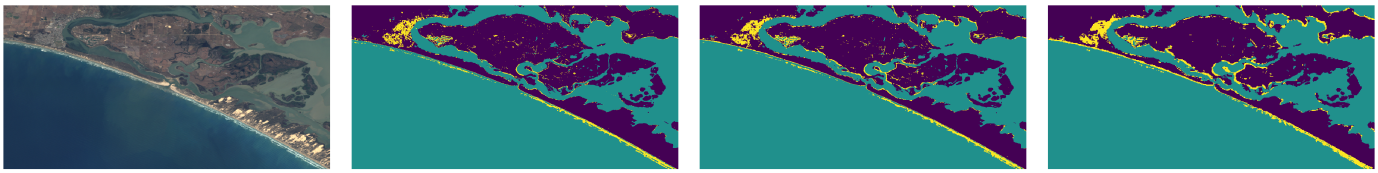
The GRC tool offers advanced classification solutions for water and sediment across the globe, leveraging the power of remote sensing and machine learning. This tool stands out for its exceptional flexibility, ease of use, and global scope, allowing users not only to define custom geographic areas but

[5] https://earthengine.google.com/



(a) RGB image (b) Our prediction - 15m (c) Our prediction - 30m (d) Our prediction - 60m

Fig. 9. Results of our GraphRiverClassifier tool from an image of the **Ucayali River, Peru** which is not included in the RiverPIXELS dataset. Region information: a rectangle centered at -73.4487, -4.45291 with a longitude range of 0.2 and a latitude range of 0.15. Panel (a) is the RGB visualization of the 30-meter resolution. Panels (b) - (d) are three predictions of the same region with different resolutions: (b) higher resolution (15 m), (c) native resolution (30 m), (d) coarser resolution (60 m). Purple, cyan, and yellow represent land, water, and sediment respectively. Our predictions are robust among various resolutions.



(a) RGB image (b) Our prediction - 15m (c) Our prediction - 30m (d) Our prediction - 60m

Fig. 10. Results of our GraphRiverClassifier tool from an image of the **Murray River, Australia** which is not included in the RiverPIXELS dataset. Region information: a rectangle centering at 138.88, -35.559 with a longitude range of 0.3 and a latitude range of 0.15. Panel (a) is the RGB visualization of the 30-meter resolution. Panels (b) - (d) are three predictions of the same region with different resolutions: (b) higher resolution (15 m), (c) native resolution (30 m), (d) coarser resolution (60 m). Purple, cyan, and yellow represent land, water, and sediment respectively. Our predictions are robust among various resolutions.

also to choose from a variety of Landsat datasets and temporal ranges. Such versatility ensures that users can conduct detailed and specific analyses tailored to their research or management needs. We note that the workflow of the tool also enables rapid development and testing of new classification algorithms.

In terms of processing speed, our GRC demonstrates impressive efficiency. For an image with one million pixels, the full process typically takes between 3 to 5 minutes. This includes data identification and download from GEE and storage to your Google Drive, which takes approximately 60 to 90 seconds, followed by feature embedding with a neural network and classification via graph learning, taking around 180 seconds in total. Figures 9, 10 show the visualization of two regions about the Ucayali and Murray Rivers that do not belong to the RiverPIXELS dataset.

C. Robustness to Different Resolutions

According to Section III-A, the augmentations we designed for contrastive learning allow the embedding feature vectors to be robust with different resolutions. The parameter ”scale” allows us to extract images of different resolutions on the same region. In Figures 9, 10, we present results of three resolutions, 15-meter, 30-meter, and 60-meter. The classification results are almost the same in panels (b)-(d), which verifies the robustness of our feature preprocessing approach.

VI. Conclusion

We develop a contrastive graph-based active learning pipeline (CGAP), which enhances the previously proposed GAP [12] by incorporating contrastive learning to train a shallow feature embedding neural network as a preprocessing step. The contrastive learning method is compatible with both supervised and unsupervised scenarios through the use of SupCon and SimCLR loss functions. It offers two significant advantages: first, based on our custom augmentations, the processed embedding feature vectors become more robust to different resolutions, cloud coverage, and geometric transformations; second, such a neural network significantly reduces the dimensionality of the feature vectors, greatly improving the efficiency of subsequent steps. Our methods' most significant benefit is their minimal data requirement, using around 3 thousand training vectors to surpass the performance of CNN neural networks trained with 2.1 million pixels, and also outperforming SVM and RF models trained with larger datasets. As shown here, the CGAP method has significant improvements in efficiency and accuracy compared to GAP. Furthermore, we introduced two versions, B-CGAP and A-CGAP, where the former is suitable for situations with complete ground-truth information, while the latter is useful for training from scratch with no ground-truth information. A-CGAP identifies the pixels most crucial to the graph-learning model and asks a human-in-the-loop to provide these labels. Experiments show that A-CGAP's performance is similar to our previously proposed GAP method, slightly inferior to B-CGAP.

We provide a Python-based tool, GraphRiverClassifier (GRC), to detect surface water and sediment globally. This tool utilizes Google Earth Engine to identify and download

Landsat scenes for a selected area and then applies our B-CGAP method for pixel classification. The tool is designed for ease of use and flexibility, allowing specifications of regions, place names, and/or selected time periods.

For future work, we suggest two main areas for development. The first area focuses on augmenting the datasets used for training. Currently, the scarcity of annotations for urban areas, such as cities and buildings, leads to misclassification of these regions as sediment rather than land. Expanding datasets like RiverPIXELS to include more comprehensive information about urban areas or employing automatic annotation methods to generate pseudo-labels could help strengthen model performance in these challenging scenarios. Furthermore, it is imperative to expand the applicability of the models to encompass a broader range of data modalities beyond the current reliance on Landsat data. Researchers need to develop techniques to align data from a diverse range of sensors, scales, and channel characteristics to ensure the effective application of the proposed methods.

The second area for future work concerns the development of improved algorithms for the feature learning and active learning components of the proposed pipeline. Firstly, one could consider incorporating transfer learning by leveraging newly acquired unlabeled images to refine our feature-embedding neural network using the SimCLR loss. Secondly, one could explore novel graph-based semi-supervised learning methods to improve the classifier performance within our pipeline. Methods such as p-Laplace learning [57], [58] or reweighting the graph before processing the Laplace learning [59], [60] could provide further improvement.

REFERENCES

[1] S. W. Cooley, J. C. Ryan, and L. C. Smith, "Human alteration of global surface water storage variability," *Nature*, vol. 591, no. 7848, pp. 78–81, 2021.

[2] T. Chen, C. Song, L. Ke, J. Wang, K. Liu, and Q. Wu, "Estimating seasonal water budgets in global lakes by using multi-source remote sensing measurements," *Journal of Hydrology*, vol. 593, p. 125781, 2021.

[3] V. Kandekar, C. Pande, J. Rajesh, A. Atre, S. Gorantiwar, S. Kadam, B. Gavit *et al.*, "Surface water dynamics analysis based on sentinel imagery and Google Earth Engine platform: A case study of Jayakwadi dam," *Sustainable Water Resources Management*, vol. 7, no. 3, pp. 1–11, 2021.

[4] J. Schwenk and E. Foufoula-Georgiou, "Meander cutoffs nonlocally accelerate upstream and downstream migration and channel widening," *Geophysical Research Letters*, vol. 43, no. 24, pp. 12–437, 2016.

[5] R. C. Sharma, R. Tateishi, K. Hara, and L. V. Nguyen, "Developing superfine water index (swi) for global water cover mapping using modis data," *Remote Sensing*, vol. 7, no. 10, pp. 13 807–13 841, 2015.

[6] J.-F. Pekel, A. Cottam, N. Gorelick, and A. S. Belward, "High-resolution mapping of global surface water and its long-term changes," *Nature*, vol. 540, no. 7633, pp. 418–422, 2016.

[7] G. Donchyts, F. Baart, H. Winsemius, N. Gorelick, J. Kwadijk, and N. Van De Giesen, "Earth's surface water change over the past 30 years," *Nature Climate Change*, vol. 6, no. 9, pp. 810–813, 2016.

[8] A. Dewan, R. Corner, A. Saleem, M. M. Rahman, M. R. Haider, M. M. Rahman, and M. H. Sarker, "Assessing channel changes of the Ganges-Padma River system in Bangladesh using Landsat and hydrological data," *Geomorphology*, vol. 276, pp. 257–279, 2017.

[9] M. G. Rozo, A. C. Nogueira, and C. S. Castro, "Remote sensing-based analysis of the planform changes in the upper Amazon river over the period 1986–2006," *Journal of South American Earth Sciences*, vol. 51, pp. 28–44, 2014.

[10] J. Schwenk, A. Khandelwal, M. Fratkin, V. Kumar, and E. Foufoula-Georgiou, "High spatiotemporal resolution of river planform dynamics from landsat: The rivmap toolbox and results from the Ucayali river," *Earth and Space Science*, vol. 4, no. 2, pp. 46–75, 2017.

[11] L. F. Isikdogan, A. Bovik, and P. Passalacqua, "Seeing through the clouds with DeepWaterMap," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 10, pp. 1662–1666, 2019.

[12] B. Chen, K. Miller, A. L. Bertozzi, and J. Schwenk, "Graph-based active learning for surface water and sediment detection in multispectral images," in *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, 2023, pp. 5431–5434.

[13] K. M. Andreadis, G. J.-P. Schumann, and T. Pavelsky, "A simple global river bankfull width and depth database," *Water Resources Research*, vol. 49, no. 10, pp. 7164–7168, 2013.

[14] P. Lin, M. Pan, G. H. Allen, R. P. de Frasson, Z. Zeng, D. Yamazaki, and E. F. Wood, "Global estimates of reach-level bankfull river width leveraging big data geospatial analysis," *Geophysical Research Letters*, vol. 47, no. 7, p. e2019GL086405, 2020.

[15] J. C. Rowland, E. Shelef, P. A. Pope, J. Muss, C. Gangodagamage, S. P. Brumby, and C. J. Wilson, "A morphology independent methodology for quantifying planview river change and characteristics from remotely sensed imagery," *Remote Sensing of Environment*, vol. 184, pp. 212–228, 2016.

[16] Y. Li, Q. Wang, C. Wu, S. Zhao, X. Xu, Y. Wang, and C. Huang, "Estimation of chlorophyll a concentration using nir/red bands of meris and classification procedure in inland turbid water," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 3, pp. 988–997, 2011.

[17] D. M. Bjerklie, "Estimating the bankfull velocity and discharge for rivers using remotely sensed river morphology information," *Journal of hydrology*, vol. 341, no. 3-4, pp. 144–155, 2007.

[18] J. Schwenk and J. Rowland, "Riverpixels: paired landsat images and expert-labeled sediment and water pixels for a selection of rivers v1.0," 1 2022. [Online]. Available: https://www.osti.gov/biblio/1865732

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] J. W. Jones, "Improved automated detection of subpixel-scale inundation—revised dynamic surface water extent (DSWE) partial surface water tests," *Remote Sensing*, vol. 11, no. 4, p. 374, 2019.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[22] D. B. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Communications on pure and applied mathematics*, 1989.

[23] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[24] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on image processing*, vol. 10, no. 2, pp. 266–277, 2001.

[25] E. Merkurjev, T. Kostic, and A. L. Bertozzi, "An MBO scheme on graphs for classification and image processing," *SIAM Journal on Imaging Sciences*, vol. 6, no. 4, pp. 1903–1930, 2013.

[26] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 755–768, 2013.

[27] D. Valsesia, G. Fracastoro, and E. Magli, "Deep graph-convolutional image denoising," *IEEE Transactions on Image Processing*, vol. 29, pp. 8226–8237, 2020.

[28] H. Hu, J. Sunu, and A. L. Bertozzi, "Multi-class graph Mumford-Shah model for plume detection using the MBO scheme," *Proceedings of the EMMCVPR conference in Hong Kong 2015*, vol. 8932, pp. 209–222, 2015, x. -C. Tai et al (Eds), Springer Lecture Notes in Computer Science.

[29] Z. M. Boyd, E. Bae, X.-C. Tai, and A. L. Bertozzi, "Simplified energy landscape for modularity using total variation," *SIAM Journal on Applied Mathematics*, vol. 78, no. 5, pp. 2439–2464, 2018.

[30] Z. M. Boyd, M. A. Porter, and A. L. Bertozzi, "Stochastic block models are a discrete surface tension," *Journal of Nonlinear Science*, vol. 30, no. 5, pp. 2429–2462, 2020. [Online]. Available: https://doi.org/10.1007/s00332-019-09541-8

[31] G. Iyer, J. Chanussot, and A. L. Bertozzi, "A graph-based approach for data fusion and segmentation of multimodal images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 5, pp. 4419–4429, May 2021.

[32] J. Qin, H. Lee, J. T. Chi, L. Drumetz, J. Chanussot, Y. Lou, and A. L. Bertozzi, "Blind hyperspectral unmixing based on graph total variation

regularization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 4, pp. 3338–3351, 2021.

[33] B. Chen, Y. Lou, A. L. Bertozzi, and J. Chanussot, "Graph-based active learning for nearly blind hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[34] J. Brown, R. O'Neill, J. Calder, and A. L. Bertozzi, "Utilizing contrastive learning for graph-based active learning of sar data," in *Algorithms for Synthetic Aperture Radar Imagery XXX*, vol. 12520. SPIE, 2023, pp. 181–195.

[35] J. Chapman, B. Chen, Z. Tan, J. Calder, K. Miller, and A. L. Bertozzi, "Novel batch active learning approach and its application to synthetic aperture radar datasets," in *Algorithms for Synthetic Aperture Radar Imagery XXX*, E. Zelnio and F. D. Garber, Eds., vol. 12520, International Society for Optics and Photonics. SPIE, 2023, p. 125200B. [Online]. Available: https://doi.org/10.1117/12.2662393

[36] H. Jafarzadeh, M. Mahdianpari, and E. Gill, "Wet-gc: A novel multi-model graph convolutional approach for wetland classification using sentinel-1 and 2 imagery with limited training samples," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2022.

[37] B. Settles, *Active Learning*. Morgan & Claypool Publishers LLC, Jun. 2012, vol. 6, no. 1. [Online]. Available: https://doi.org/10.2200/s00429ed1v01y201207aim018

[38] K. Miller, J. Mauro, J. Setiadi, X. Baca, Z. Shi, J. Calder, and A. L. Bertozzi, "Graph-based active learning for semi-supervised classification of SAR data." Society of Photo-Optical Instrumentation Engineers (SPIE) Conference on Defense and Commercial Sensing, 2022, https://arxiv.org/abs/2204.00005.

[39] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[41] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[42] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[43] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[44] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." *Journal of machine learning research*, vol. 7, no. 11, 2006.

[45] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[46] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[47] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[48] A. L. Bertozzi, X. Luo, A. M. Stuart, and K. C. Zygalakis, "Uncertainty quantification in the classification of high dimensional data," *SIAM/ASA J. Uncertainty Quantification*, vol. 6, no. 2, pp. 568–595, 2018.

[49] K. Miller, H. Li, and A. L. Bertozzi, "Efficient graph-based active learning with probit likelihood via Gaussian approximations," in *ICML Workshop on Experimental Design and Active Learning*. International Conference on Machine Learning (ICML), Jul. 2020, arXiv: 2007.11126.

[50] Y. Qiao, C. Shi, C. Wang, H. Li, M. Haberland, X. Luo, A. M. Stuart, and A. L. Bertozzi, "Uncertainty quantification for semi-supervised multi-class classification in image processing and ego-motion analysis of body-worn videos," *Electronic Imaging*, vol. 2019, no. 11, pp. 264–1, 2019.

[51] B. Chen, K. Miller, A. L. Bertozzi, and J. Schwenk, "Batch active learning for multispectral and hyperspectral image segmentation using similarity graphs," *Communications on Applied Mathematics and Computation*, pp. 1–21, 2023.

[52] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 60–65.

[53] K. S. Miller and A. L. Bertozzi, "Model change active learning in graph-based semi-supervised learning," *Communications on Applied Mathematics and Computation*, pp. 1–29, 2024.

[54] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[55] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[56] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, "Google Earth Engine: Planetary-scale geospatial analysis for everyone," *Remote sensing of Environment*, vol. 202, pp. 18–27, 2017.

[57] J. Calder, "The game theoretic p-laplacian and semi-supervised learning with few labels," *Nonlinearity*, vol. 32, no. 1, p. 301, 2018.

[58] M. Flores, J. Calder, and G. Lerman, "Analysis and algorithms for ℓ_p-based semi-supervised learning on graphs," *Applied and Computational Harmonic Analysis*, vol. 60, pp. 77–122, 2022.

[59] J. Calder and D. Slepčev, "Properly-weighted graph laplacian for semi-supervised learning," *Applied mathematics & optimization*, vol. 82, pp. 1111–1159, 2020.

[60] K. Miller and J. Calder, "Poisson reweighted laplacian uncertainty sampling for graph-based active learning," *SIAM Journal on Mathematics of Data Science*, vol. 5, no. 4, pp. 1160–1190, 2023.

VII. Biography

Bohan Chen is a postdoctoral researcher at computing + mathematical sciences (CMS) at the California Institute of Technology. He received a B.S. degree in math from the School of Mathematical Sciences, Peking University and received a Ph.D. in Mathematics from the University of California, Los Angeles. He received the UC-National Laboratory In-Residence Graduate Fellowship in 2021 and works partially as a visiting scholar at Los Alamos National Laboratory (LANL). His research interests include data assimilation, image processing, graph semi-supervised learning, and active learning.

Kevin Miller is an Assistant Professor of Mathematics at Brigham Young University, Provo. Previously he was a Peter J. O'Donnell Postdoctoral Fellow at the Oden Institute for Computational Engineering and Sciences at the University of Texas, Austin. He received his Ph.D. in Mathematics from the University of California, Los Angeles where he was awarded the Department of Defense's National Defense Science and Engineering Graduate (ND-SEG) Fellowship. His research focuses on theory and applications for active learning and uncertainty quantification, specializing in graph-based methods for semi-supervised classification.

Andrea L. Bertozzi is Professor of Mathematics and Mechanical and Aerospace Engineering at the University of California Los Angeles. She holds the Betsy Wood Knapp Chair for Innovation and Creativity since 2012. She earned her Ph.D. degree in Mathematics from Princeton. She held faculty positions at the Univ. of Chicago and Duke University before moving to UCLA in 2003. During 1995-6 she was the Maria-Geoppert Mayer Distinguished Scholar at Argonne National Laboratory. Bertozzi is a member of the US National Academy of Sciences and a Fellow of the American Academy of Arts and Sciences. She was awarded SIAM's Kovalevsky Prize in 2009 and SIAM's Kleinman Prize in 2019. Her research focuses on graph-based machine learning, analysis, and numerical methods for partial differential equations, as well as complex systems, networks, and fluid dynamics.

Jon Schwenk is a research Scientist at Los Alamos National Laboratory in the Division of Analytics, Intelligence, and Technology. He earned a Ph.D. degree in Civil Engineering from the University of Minnesota, Twin Cities while at the Saint Anthony Falls Laboratory. He held a postdoc position at the University of California, Irvine before being awarded a Director's Postdoctoral Fellowship at Los Alamos National Lab.

Jon's research aims at applying big data and novel techniques to problems in fluvial geomorphology and hydrology. He is particularly interested in the use of remotely-sensed imaging to test and recreate fundamental concepts and theories in river morphodynamics and river network structure. He has authored multiple software packages that automate large-scale analysis of river channels and networks in an effort to exploit the wealth of available satellite images.