Earth Imagery Segmentation on Terrain Surface with Limited Training Labels: A Semi-supervised Approach based on Physics-Guided Graph Co-Training

WENCHONG HE, Department of Computer & Information Science & Engineering, The University of Florida, USA
ARPAN MAN SAINJU, Department of Computer Science, Middle Tennessee State University, USA
ZHE JIANG, Department of Computer & Information Science & Engineering, The University of Florida, USA
DA YAN, Department of Computer Science, University of Alabama, Birmingham, USA
YANG ZHOU, Department of Computer Science and Software Engineering, Auburn University, USA

Given earth imagery with spectral features on a terrain surface, this paper studies surface segmentation based on both explanatory features and surface topology. The problem is important in many spatial and spatiotemporal applications such as flood extent mapping in hydrology. The problem is uniquely challenging for several reasons: first, the size of earth imagery on a terrain surface is often much larger than the input of popular deep convolutional neural networks; second, there exists topological structure dependency between pixel classes on the surface, and such dependency can follow an unknown and non-linear distribution; third, there are often limited training labels. Existing methods for earth imagery segmentation often divide the imagery into patches and consider the elevation as an additional feature channel. These methods do not fully incorporate the spatial topological structural constraint within and across surface patches and thus often show poor results, especially when training labels are limited. Existing methods on semi-supervised and unsupervised learning for earth imagery often focus on learning representation without explicitly incorporating surface topology. In contrast, we propose a novel framework that explicitly models the topological skeleton of a terrain surface with a contour tree from computational topology, which is guided by the physical constraint (e.g., water flow direction on terrains). Our framework consists of two neural networks: a **convolutional neural network (CNN)** to learn spatial contextual features on a 2D image grid, and a **graph neural network (GNN)** to learn the statistical distribution of physics-guided spatial topological dependency on the contour tree. The two models are co-trained via variational EM. Evaluations on the real-world flood mapping datasets show that the proposed models outperform baseline methods in classification accuracy, especially when training labels are limited.

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. IIS-1850546, IIS-2008973, CNS-1951974, OAC-2152085, and the National Oceanic and Atmospheric Administration (NOAA), Microsoft AI for Earth Grant and the Extreme Science and Engineering Discovery Environment (XSEDE).
Authors' addresses: W. He and Z. Jiang (corresponding author), Department of Computer & Information Science & Engineering, The University of Florida, P.O. Box 116120, Gainesville, Florida, USA, 32611; emails: {whe2, zhe.jiang}@ufl.edu; A. M. Sainju, Department of Computer Science, Middle Tennessee State University, Murfreesboro, TN, USA, 37132; email: asainju@mtsu.edu; D. Yan, Department of Computer Science, University of Alabama, Birmingham, Birmingham, AL, USA, 35294; email: yanda@uab.edu; Y. Zhou, Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA, 36849; email: yangzhou@auburn.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2157-6904/2022/01-ART26 $15.00
https://doi.org/10.1145/3481043

CCS Concepts: • **Information systems** → *Geographic information systems*; *Data mining*; • **Computing methodologies** → *Machine learning*; • **Applied computing** → *Earth and atmospheric sciences;*

Additional Key Words and Phrases: Earth image segmentation, terrain surface, contour tree, physic-guided, semi-supervised

ACM Reference format:
Wenchong He, Arpan Man Sainju, Zhe Jiang, Da Yan, and Yang Zhou. 2022. Earth Imagery Segmentation on Terrain Surface with Limited Training Labels: A Semi-supervised Approach based on Physics-Guided Graph Co-Training. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 26 (January 2022), 22 pages.
https://doi.org/10.1145/3481043

1 INTRODUCTION

Given earth imagery with spectral features on a terrain surface (defined by an elevation function over a 2D grid), this paper studies the problem of surface segmentation based on both explanatory features and surface topology. The problem is important for many spatial and spatiotemporal applications, such as mapping flood surface extent in hydrology and identifying pocket structures on protein elevation surface in biochemistry [12, 13, 19, 23, 48, 51]. Figure 1 provides a real-world example in earth science and hydrology. Given remote sensing images with spectral features (Figure 1(a)) and geographic terrains based on the digital elevation (Figure 1(b)), the problem aims to classify pixels into flood and dry classes (Figure 1(c)) based on not only spectral features but also spatial surface topography. Specifically, the spatial extent of the flood area in Figure 1(c) follows the geospatial topological structural dependency on the elevation surface in Figure 1(b). Mapping flood extent on the Earth's surface can not only improve the situational awareness for disaster response agencies, but also enhance the flood forecasting capabilities at the NOAA National Water Center [35].

The problem is uniquely challenging for several reasons. First, the size of a terrain surface is often very large beyond the input shape of common deep convolutional neural networks. For example, the elevation surface in Figure 1(b) contains around 10.6 million pixels. Second, there exists a global topological structure dependency between class locations on the surface, and such dependency can follow an unknown and non-linear distribution. Consider the same example in Figure 1. The floodwater locations on the surface are constrained by the surface topography due to gravity. Third, there are often limited training labels due to the slow and expensive process of collecting ground truth [26].

There exists extensive research in deep learning on image segmentation over the last decade [32]. The most popular technique is to learn a convolutional neural network [2, 30, 44] to extract high-level semantic features together with deconvolution (or upsampling) layers to combine features at multiple scales for detailed segmentation [7, 8, 38]. Promising results have been achieved on traditional camera photos and medical images. When applied to a large high-resolution topological surface defined by an elevation (or depth) function [52], these methods often require partitioning the image into smaller patches (e.g., 224 by 224) and learning a convolutional neural network for each patch with the depth channel as an additional feature [14, 31, 50]. Thus, these methods are limited in capturing topological structures due to the rigid shape patterns in convolution kernels. Some works aim to resolve this issue by adding spatial transformation in convolution kernels through a depth or Gaussian term, e.g., bilateral filters [3] and depth-aware CNN [52], but they still do not fully capture the topological structure for the entire surface. Also, these methods often require large amounts of labeled training data to ensure performance, which limits their application in the real world. Other methods incorporate topological constraints into image segmentation in

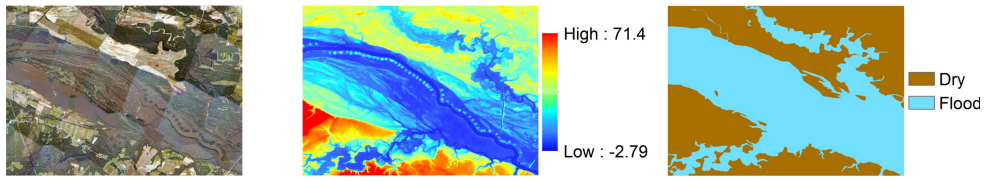

Fig. 1. A real-world example in the flood mapping application.

two ways: enforcing MRF or CRF-based topological constraints in the inference step [1, 6, 39, 49], which is unable to fully utilize a topological prior to training a model; or using a topology-aware loss function to train the neural network by leveraging persistent homology to define a topological loss on the predicted class image [18, 33]. One example is retinal layer segmentation in a medical application [17, 28], which only focuses on learning continuous boundaries between layers. But these methods only focus on simple topology constraints (e.g., the number of connected components or holes). In addition, due to the slow and expensive process of collecting a large number of segment labels in neural network training, people also study semi-supervised image segmentation. For example, some works use unlabeled data by enforcing consistency between model predictions over multiple perturbations on the hidden layer representation [41]. Others use entropy regularization to transfer the information from labeled images to unlabeled images [25]. However, these methods still do not incorporate the explicit topological structural constraints on the surface. There are also works for high-resolution image segmentation that integrates both global images and local patches to capture semantic features of different granularity [10, 54], but these works also ignore the topological structure in a terrain surface. Recently, a family of hidden Markov tree models has been proposed to incorporate physics-guided topological structural constraint on terrains for flood mapping [15, 16, 20–22, 24, 45–47, 53], but these models still use per-pixel explanatory features without learning spatial contextual features in an end-to-end manner.

In contrast, we propose a novel framework that explicitly models the topological skeleton of a terrain surface with a contour tree from computational topology, guided by the physical constraint (e.g., water flow directions on terrains). Our framework consists of two neural networks: a **convolutional neural network (CNN)** to learn spatial contextual features on a 2D image grid, and a **graph neural network (GNN)** to learn the statistical distribution of topological dependency on the contour tree. The two models are co-trained via variational EM. Evaluations on the real-world flood mapping datasets show that the proposed models outperform baseline methods in classification accuracy, especially when the number of training labels is small.

2 PROBLEM FORMULATION

2.1 Preliminaries

We now define some basic concepts. A list of symbols with description is in Table 1. **3D Spatial Topological Surface:** Given a 2D grid with N pixels, a 3D topological surface is defined as an elevation (or depth) function over the grid. We denote the elevation function by an array $\mathbf{E} \in \mathbb{R}^N$, the m explanatory features (e.g., RGB feature) on the surface by an array $\mathbf{X} \in \mathbb{R}^{N \times m}$ and the target classes by an array $\mathbf{Y} \in \mathbb{R}^N$. We denote a sample (pixel) on the surface by $\mathbf{s}_n = (\mathbf{x}_n, e_n, y_n)$, where $\mathbf{x}_n$, e_n, and y_n represent the m explanatory features, elevation value, and the class of the pixel, respectively. For example, in flood mapping, the surface elevation is collected from 3D Lidar point cloud, the explanatory features are the spectral bands from remote sensing imagery, and the target

Table 1. List of Major Symbols and Descriptions

Symbols	Domain	Descriptions
$\mathbf{x}_n$	$\mathbb{R}^m$	m explanatory features of pixel n
e_n	$\mathbb{R}$	elevation value of pixel n
y_n	$\{0, 1\}$	class of pixel n
s_n		data sample of pixel n
$\mathbf{X}$	$\mathbb{R}^{N\times m}$	features matrix of image with N pixels
$\mathbf{E}$	$\mathbb{R}^N$	elevation array of image with N pixels
$\mathbf{Y}$	$\mathbb{R}^N$	class array of image with N pixels
$\mathbf{D}_L$	$\mathbb{R}^{N_L\times m}$	dataset for labeled image
$\mathbf{D}_U$	$\mathbb{R}^{N_U\times m}$	dataset for unlabeled image
$\mathcal{V}$		The set of nodes in contour tree
$\mathcal{E}$		The set of edges between nodes in contour tree
$\mathbf{X}^{\mathcal{G}}$	$\mathbb{R}^{M\times m}$	Features matrix of contour tree
$\mathbf{Y}^{\mathcal{G}}$	$\mathbb{R}^M$	Class array of contour tree with M nodes
ϕ	$\mathbb{R}$	GNN Model parameters
θ	$\mathbb{R}$	CNN Model parameters

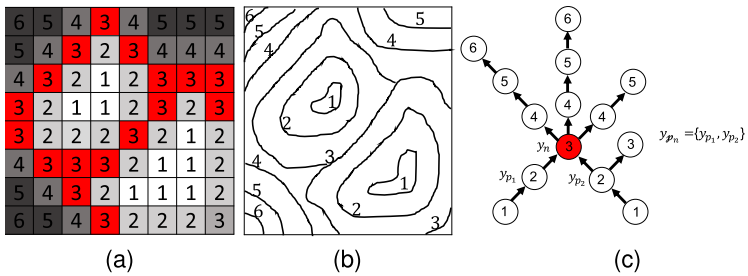

Fig. 2. (a): An example of an elevation surface defined over a 2D image, each grid represents one pixel, the number is its elevation level. (b): Contours of the surface, each circle is one contour, which represents a set of connected pixels with the same elevation level (level set). (c): Contour tree of the surface, each node represents one contour in (b). Two nodes are connected only if the two contours are adjacent.

classes are flood and dry categories. The above notations of the surface are based on a 2D grid view with each pixel as a sample unit.

Geospatial Contour Tree: In computational topology, an elevation surface can also be characterized by its level sets. Formally, a level set is a set of pixels with an equal elevation, i.e., $l(e_0) = \{\mathbf{s}_n | e_n = e_0\}$, where e_0 is an elevation threshold. A level set consists of a number of connected components called *contours*, which are in the form of equal elevation circles on a 2D surface (see Figure 2(b)). As the elevation level rises, the topology of the corresponding level set and their contours also evolve. Such evolution can be represented by a *contour tree*. A contour tree is a directed polytree whose nodes represent contour components and whose edges represent the topological order (based on elevation gradient) between two adjacent contour components [5, 42]. This paper focuses on a geospatial contour tree whose elevation function is defined on a 2D spatial grid (e.g., elevation of the Earth's surface). Figure 2(c) provides an example of the contour tree corresponding to the elevation surface in Figure 2(a). Assume that the areas where elevation level is not greater than 1 are flooded in Figure 2(a). Then in Figure 2(c) the two leaf nodes correspond to the appearance of two separate lakes as water starts to accumulate to the elevation level of 1. The contour tree skeleton captures how the surface class segments (e.g., flood extent) evolve with different

elevation levels. Consider the contour tree node with elevation 3 as y_n (the red node in Figure 2(c)), then its parent nodes $y_{\mathcal{P}_n}$ are the set of adjacent nodes with elevation level 2, $y_{\mathcal{P}_n} = \{y_{\mathrm{p}_1}, y_{\mathrm{p}_2}\}$ (as shown in Figure 2(c)). Note that each contour tree node corresponds to a set of pixels on the corresponding contour segment. For example, the red node in the contour tree with elevation 3 corresponds to all red pixels in the elevation map on the left. In a contour tree representation, each contour component (node) is considered as a sample unit. The contour tree structure can be represented with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges between the nodes. Each node v_i corresponds to a level set $l(e_i)$ in the elevation surface. Thus, the feature of node v_i is the aggregation of the feature of those pixels in the level set $l(e_i)$. We denote the feature matrix of all graph notes as $\mathbf{X}^{\mathcal{G}}$, where each node feature $x_i^{\mathcal{G}} = Agg(l(e_i)), i \in \mathcal{V}$, and $l(e_i)$ represent those samples corresponding to i_{th} contour component in surface image. For the aggregation function, we choose to use the average function.

Specifically, we have two separate topological surfaces. One is for training with known class labels, denoted by surface channels $(\mathbf{X}_L, \mathbf{E}_L, \mathbf{Y}_L)$ or samples $\mathcal{D}_L$. The other is for testing with unknown (hidden) class labels, denoted by surface channels $(\mathbf{X}_U, \mathbf{E}_U, \mathbf{Y}_U)$ or samples $\mathcal{D}_U$. We denote $\mathbf{X} = \mathbf{X}_L \cup \mathbf{X}_U$ and $\mathbf{Y} = \mathbf{Y}_L \cup \mathbf{Y}_U$. We denote the samples in entire dataset as $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$.

Based on the elevation feature, we can construct the corresponding contour tree for the entire training and test topological surface as one graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (connected graph if training and test region are adjacent). We denote the nodes with known class labels as $\mathcal{V}_L$, and the nodes with unknown class labels as $\mathcal{V}_U$. The correspondence between the graph nodes and pixels in the surface is represented by matrix $\mathbf{H} \in \mathbb{R}^{M \times N}$, where M is the total number of vertices in the contour tree, and N is the total number of pixels in the surface image. $\mathbf{H}_{ij} = 1$ if the i_{th} contour tree node contains pixels s_j in surface image and 0 otherwise. The feature and class label matrix of all nodes is denoted by $\mathbf{X}^{\mathcal{G}}$ and $\mathbf{Y}^{\mathcal{G}}$, which are obtained by averaging the corresponding pixels feature and label in the image. In the following, for simplicity, we omit the superscripts $\mathcal{G}$ for the graph representation. In other words, $\mathbf{X}$ and $\mathbf{Y}$ can be the class and features of the surface image or contour tree based on its context. $\mathbf{x}_n$ and $\mathbf{y}_n$ represent the feature and class label of node n in contour tree if $n \in \mathcal{V}$, and pixel n in surface image if $n \in \mathcal{D}$.

Construction of Contour Tree: A contour tree can be constructed by the algorithm in [5]. The algorithm sorts surface pixels from low to high elevation values, creating a joint tree and a split tree by scanning the pixels in different orders of elevation values. We need to customize the original algorithm in [5] for our terrain surface. First, we assume each pixel center as a vertex in the mesh surface format. Second, we add perturbation to surface elevation values to enforce a total order on pixels with an equal elevation value (this is required by the algorithm). After the two customizations, we can run the algorithm in [5] and then collapse the nodes on the same contour segment into a single node [21]. The construction algorithm takes $O(n \log n)$ time where n is the number of surface pixels.

2.2 Problem Definition

Given a training surface with labeled samples $(\mathbf{X}_L, \mathbf{E}_L, \mathbf{Y}_L)$ and a test surface with unlabeled samples $(\mathbf{X}_U, \mathbf{E}_U, \mathbf{Y}_U)$, the problem of topological surface segmentation aims to learn a model to predict the classes of test samples, i.e., $\mathbf{Y}_U = f(\mathbf{X}_U, \mathbf{E}_U)$. We assume the test surface is very large with millions of pixels. In addition, we also assume that the pixel classes on the surface follow a contour tree topological structure. From the perspective of the contour tree structure, this problem can be formulated as a structured prediction problem that aims to model the distribution of nodes class conditioned on node features and contour tree topology, i.e., $p(Y^{\mathcal{G}}|X^{\mathcal{G}}, \mathcal{E})$. Specifically, we focus on semi-supervised transductive learning, that is, to learn a model by utilizing not only the labeled

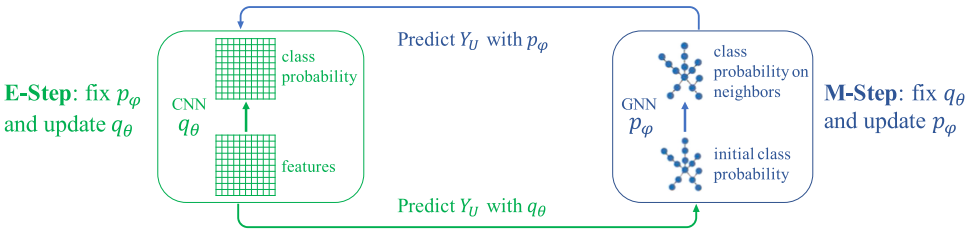

Fig. 3. Overview of the co-training framework.

training surface but also the unlabeled test surface as well. The training and test surfaces may also belong to a single larger surface that is partially labeled.

3 THE PROPOSED APPROACH

3.1 Overview of the Framework

Figure 3 provides an overview of our framework. Our idea is to use two deep neural networks: a ***convolutional neural network*** **(CNN)** [30, 44] on an image grid and a ***graph neural network*** **(GNN)** [11, 27] on a contour tree skeleton. The GNN model captures the topological structure between contour classes but ignores the spatial contextual dependency among pixel features. The CNN model captures the spatial context of features but ignores the topological structure between classes. The backbone network for GNN we use is **Graph Convolutional Network (GCN)** [27], for its simplicity and capability to aggregate neighborhood features. In our implementation, we used 3 GCN layers to learn higher-order dependency between neighbors. The backbone network for CNN is U-Net, because it can capture both global and local spatial contextual features for segmentation. U-Net model consists of an encoder-decoder structure, where the encoder has six double-convolution and five max-pooling operations, and the decoder uses transposed convolution to upsample the feature map. More details of the U-Net and GCN models are provided in Section 4.1. The two models are co-trained iteratively via variational EM [37]. The intuition is to consider the CNN and GNN models as two alternative ways to formulate the statistical distribution of all sample locations and iteratively train the two models by approximating one with the other [43]. Specifically, we can first use a pre-trained CNN to predict class probabilities of pixels on the surface. The predicted class probabilities will be aggregated into contour nodes and fed into a GNN to learn topological structure dependency. The output classes from the GNN that enforce topological structure dependency will in turn be used to re-train the CNN model. The iterations continue until the two models are converged. The converged models combine the advantages of both sides, being able to both learn spatial contextual features and enforce the topological structure between classes. Next, we will introduce the specific statistical formulation and the detailed learning algorithms under the variational EM framework.

3.2 Model Co-Training in Variational EM

Given a partially labeled surface (training surface and test surface), we assume the class labels and features of all datasets follow the conditional distribution in Equation (1), where y_n and $\mathbf{y}_{\mathcal{P}_n}$ are the class of the n-th contour tree node and the classes its parent contour nodes, respectively, $\mathbf{x}_n$ is the aggregated features for pixels on the contour, and ϕ is model parameters. In this formulation, we make conditional independence assumptions between class nodes based on the topological structure represented by a contour tree, i.e., a node's class only depends on its parents' classes and its own feature. For example, in hydrology, the flood extent boundary on the elevation surface only gradually spreads across different contours. The above distribution can be considered as a

graphical model based on a contour tree. It captures the topological dependency between classes of contour components on a surface, ignoring the spatial context of surface features (assuming $\mathbf{x}_n$ to be independent). In order to learn the function of $p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{x}_n)$ in an unknown and nonlinear manner, we use a graph neural network to model the probability distribution of p_ϕ. A graph neural network generalizes the traditional deep convolutional neural networks from images to graphs by designing convolutional operations on a node's neighbors [11, 27]. If we feed the initial class probability $p(y_i)$ of each contour tree node into the input "feature" layer of the GNN, the output layer will generate class probabilities of each node after considering all neighbors' classes through graph convolution operations. The neural network architecture is able to learn more complex functions beyond a simple class transition matrix. In addition, with specific configurations of the hyper-parameters such as the number of neighbor hops and the number of graph convolution layers, we can learn higher-order dependency beyond a node and its parents.

$$p_\phi(\mathbf{Y}|\mathbf{X}) = p_\phi(\mathbf{Y}_L, \mathbf{Y}_U|\mathbf{X}) = \prod_{n\in\mathcal{V}} p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{x}_n) \tag{1}$$

Model parameters ϕ can be learned by maximizing the log-likelihood of observed node labels $\log p_\phi(\mathbf{Y}_L|\mathbf{X})$. However, directly optimizing it is intractable because of the complicated structural dependency between contours and a large number of unobserved contour classes $\mathbf{Y}_\mathbf{U}$. Alternatively, we maximize the evidence lower bound of the log-likelihood function of the observed contour classes, which is given in Equation (2), where $q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})$ is a variational distribution over the unlabeled pixels $\mathbf{Y}_\mathbf{U}$. The equation holds when $q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})$ is equal to the true posterior distribution $p_\phi(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X})$. Because the true posterior distribution of unlabeled pixels is intractable, we use the variational distribution $q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})$ to approximate it. The optimal $q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})$ is the one that maximizes the evidence lower bound $\mathcal{L}(q_\theta, p_\phi)$.

$$\log p_\phi(\mathbf{Y}_L|\mathbf{X}) \geq \mathcal{L}(q_\theta, p_\phi) = \mathbb{E}_{q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})}[\log p_\phi(\mathbf{Y}_L, \mathbf{Y}_U|\mathbf{X}) - \log q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})] \tag{2}$$

Specifically, we apply the mean-field theory in the approximation [40], assuming pixel classes are conditionally independent given features. The variational distribution $q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X})$ can be factorized as shown in Equation (3) below, where each factorized distribution $q_\theta(y_n|\mathbf{X})$ is a categorical distribution of a pixel class based on complete surface features. To learn the complex unknown and non-linear spatial contextual features that determine pixel classes, we use a convolutional neural network (e.g., U-Net) to model variational distribution $q_\theta(y_n|\mathbf{X})$. The network takes the explanatory features of pixels on a surface as inputs and produces the class probabilities of each pixel as output. Note that if the surface is much larger than the input shape of a normal CNN, we can partition the surface into smaller patches and apply the CNN model to each patch to predict pixel class probabilities.

$$q_\theta(\mathbf{Y}_\mathbf{U}|\mathbf{X}) = \prod_{n\in\mathcal{D}_U} q_\theta(y_n|\mathbf{X}) \tag{3}$$

Note that we *slightly abuse the notation* to use the same element index n for both a contour node (a set of pixels) in $p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{x}_n)$ and a pixel in $q_\theta(y_n|\mathbf{X})$ for simplicity. In other words, en element y_n as a node in p_ϕ corresponds to a set of y_n as pixels in q_θ. The model learning process involves iterations between an E-step and an M-step. In the E-step, we fix the parameters ϕ in GNN and optimize q_θ (i.e., re-training the CNN). In the M-step, we fix the parameter θ and optimize p_ϕ (i.e., re-training the GNN). The iterations continue until the two models converge. The specific details are discussed below.

3.2.1 E-step: Fix p_ϕ and Update q_θ. In E-step, we fix p_ϕ and optimize q_θ through maximizing the evidence lower bound in Equation (2) (making the lower bound as tight as possible). Recall that

we use the mean field theory to express q_θ in a factorized distribution $q_\theta(y_n|\mathbf{X})$ in Equation (3). Such an optimal marginal variational posterior distribution $q_\theta(y_n|\mathbf{X})$ can be derived by the fixed-point condition [4, 43] in Equation (4), where θ_0 in the expectation is the old parameters of θ from the previous iteration (the detailed proof is provided in Theorem 3.1). The intuition behind this fix-point condition is that the logarithm of the optimal variational distribution on one factor y_n can be expressed as the posterior expectation of logarithm of p_ϕ over remaining factors $y_{\mathcal{P}_n}$.

$$\log q_\theta(y_n|\mathbf{X}) = \mathbb{E}_{q_{\theta_0}(\mathbf{y}_{\mathcal{P}_n}|\mathbf{X})}[\log p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{X})] + \text{const} \tag{4}$$

The optimal solution involves the expectation over the variational posterior distribution of $\mathbf{y}_{\mathcal{P}_n}$. The expectation may involve a large number of terms since the expectation ranges over all possible parent nodes classes. To address this challenge and simplify the condition, we estimate the expectation by drawing a sample from $q_{\theta_0}(\mathbf{y}_{\mathcal{P}_n}|\mathbf{X})$, denoted by $\hat{\mathbf{y}}_{\mathcal{P}_n}$. In this way, the expectation over q_θ in Equation (4) can be simplified as $\mathbb{E}_{q_{\theta_0}(\mathbf{y}_{\mathcal{P}_n}|\mathbf{X})}[\log p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{X})] \approx \log p_\phi(y_n|\hat{\mathbf{y}}_{\mathcal{P}_n}, \mathbf{X})$. Thus, the optimal solution of marginal variational posterior satisfies the equation below.

$$q_\theta(y_n|\mathbf{X}) = p_\phi(y_n|\hat{\mathbf{y}}_{\mathcal{P}_n}, \mathbf{X}) \tag{5}$$

Based on the above analysis, in the E-step, we can fix parameter ϕ and use old parameter θ_0 to sample classes by distribution q_{θ_0}. The sampled classes are fed into the GNN model p_ϕ to produce $p_\phi(y_n|\hat{\mathbf{y}}_{\mathcal{P}_n}, \mathbf{X})$. After this, we can fix $p_\phi(y_n|\hat{\mathbf{y}}_{\mathcal{P}_n}, \mathbf{X})$ as a target to update parameter in q_θ. According to Equation (5), the optimal variational distribution q_θ can be updated by minimizing the reverse KL divergence between $q_\theta(y_n|\mathbf{X})$ and the target $p_\phi(y_n|\hat{\mathbf{y}}_{\mathcal{P}_n}, \mathbf{X})$, which gives the following loss function in Equation (6), which is similar to categorical cross entropy loss with a soft ground true class probabilities.

$$\mathcal{L}_{\theta, \mathcal{D}_U} = \sum_{n \in \mathcal{D}_U} \mathbb{E}_{p_\phi(y_n|\hat{\mathbf{y}}_{\mathcal{P}_n}, \mathbf{X})}[\log q_\theta(y_n|\mathbf{X})]. \tag{6}$$

Additionally, the labeled training pixels can also be utilized to enhance the inference network by predicting the labels for the labeled pixels. We add a supervised learning loss function:

$$\mathcal{L}_{\theta, \mathcal{D}_L} = \sum_{n \in \mathcal{D}_L} \log q_\theta(y_n|\mathbf{X}), \tag{7}$$

where y_n is the ground-truth label of n. Therefore, the overall loss function is as follows:

$$\mathcal{L}_\theta = \mathcal{L}_{\theta, \mathcal{D}_U} + \lambda \mathcal{L}_{\theta, \mathcal{D}_L}, \tag{8}$$

where λ is a hyperparameter to control the relevant weight of supervised learning objective.

Theorem 3.1. *Assume $q_\theta(y_n|\mathbf{X})$ is the factorized variational distribution after assuming conditional independence between pixel classes in $q_\theta(\mathbf{Y_U}|\mathbf{X})$ (the mean field approximation). Assume that $p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{X})$ is the true conditional class distribution of the n-th contour tree node given its parent classes and the complete features. The optimal approximation of q_θ to p_ϕ that maximizes the evidence lower bound in Equation (2) is given by the following fixed point condition:*

$$\log q_\theta(y_n|\mathbf{X}) = \mathbb{E}_{q_{\theta_0}(\mathbf{y}_{\mathcal{P}_n}|\mathbf{X})}[\log p_\phi(y_n|\mathbf{y}_{\mathcal{P}_n}, \mathbf{X})] + \textit{const.}$$

Note that we slightly abuse the notation by defining y_n as the set of classes of all pixels on the n-th contour (node) in q_θ, though y_n in the original q_θ is factorized by each pixel. We also define y_n in p_ϕ as the class of the n-th contour (node) assuming that all pixels on the contour share the same class.

Proof. The evidence lower bound of the log-likelihood function of the observed contour classes is:

$$
\begin{aligned}
&\mathbb{E}_{q_\theta(\mathbf{Y_U}|\mathbf{X})}[\log p_\phi(\mathbf{Y}_L, \mathbf{Y}_U|\mathbf{X}) - \log q_\theta(\mathbf{Y_U}|\mathbf{X})] \\
&= \sum_{\mathbf{Y_U}} \prod_n q_\theta(\mathbf{y_n}|\mathbf{X}) \left[\log p_\phi(\mathbf{Y_U}, \mathbf{Y_L}|\mathbf{X}) - \sum_n \log q_\theta(\mathbf{y_n}|\mathbf{X})\right] \\
&= \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \sum_{\mathbf{Y_U}\setminus\mathbf{y_{n_0}}} \prod_{n\neq n_0} q_\theta(\mathbf{y_n}|\mathbf{X}) \left[\log p_\phi(\mathbf{Y_U}, \mathbf{Y_L}|\mathbf{X}) - \sum_n \log q_\theta(\mathbf{y_n}|\mathbf{X})\right] \\
&= \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \sum_{\mathbf{Y_U}\setminus\mathbf{y_{n_0}}} \prod_{n\neq n_0} q_\theta(\mathbf{y_n}|\mathbf{X}) \log p_\phi(\mathbf{Y_U}, \mathbf{Y_L}|\mathbf{X}) \\
&- \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \sum_{\mathbf{Y_U}\setminus\mathbf{y_{n_0}}} \prod_{n\neq n_0} q_\theta(\mathbf{y_n}|\mathbf{X}) \left[\sum_{n\neq n_0} \log q_\theta(\mathbf{y_n}|\mathbf{X}) + \log q_\theta(\mathbf{y_{n_0}}|\mathbf{X})\right] \\
&= \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \mathbb{E}_{q_\theta(\mathbf{Y_U}\setminus n_0|\mathbf{X})}[\log p_\phi(\mathbf{Y_U}, \mathbf{Y_L}|\mathbf{X})] - \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \log q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) + \text{const.} \\
&= \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \mathbb{E}_{q_\theta(\mathbf{Y_U}\setminus n_0|\mathbf{X})}[\log p_\phi(\mathbf{y_{n_0}}|\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}, \mathbf{X})] - \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \log q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) + \text{const.} \\
&= \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \mathbb{E}_{q_\theta(\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}|\mathbf{X})}[\log p_\phi(\mathbf{y_{n_0}}|\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}, \mathbf{X})] - \sum_{\mathbf{y_{n_0}}} q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) \log q_\theta(\mathbf{y_{n_0}}|\mathbf{X}) + \text{const.} \\
&= -\mathrm{KL}(q_\theta(\mathbf{y_{n_0}}|\mathbf{X})||\mathbb{E}_{q_\theta(\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}|\mathbf{X})})[\log p_\phi(\mathbf{y_{n_0}}|\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}, \mathbf{X})]) + \text{const.}
\end{aligned}
$$

From the above Equation, the optimal $q_\theta(\mathbf{y_{n_0}}|\mathbf{X})$ is equal to $\mathbb{E}_{q_\theta(\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}|\mathbf{X})})[\log p_\phi(\mathbf{y_{n_0}}|\mathbf{y}_{\mathcal{P}_{\mathbf{n_0}}}, \mathbf{X})]$ □

3.2.2 M-step: Fix q_θ and Update p_ϕ. In M-step, we fix the parameter θ in the CNN model and update parameter ϕ in the GNN model. The objective is to maximize the evidence lower bound of log-likelihood in Equation (2), i.e., $\mathbb{E}_{q_\theta(\mathbf{Y_U}|\mathbf{X})}[\log p_\phi(\mathbf{Y}_L, \mathbf{Y}_U|\mathbf{X})]$. Similar to the E-step, directly calculating the expectation over the variational posterior $\mathbf{Y_U}$ (or y_n) is infeasible due to the lack of closed form expression in p_ϕ. We utilized the same idea to generate samples from distribution q_θ, i.e. $\hat{\mathbf{y}}_n \sim q_\theta(y_n|\mathbf{X})$ if n is an unlabeled pixel. Otherwise, if n is a labeled pixel (from the training surface), we set $\hat{\mathbf{y}}_n$ as its ground-truth. In this way, the model p_ϕ can be optimized by maximizing the loss function below.

$$\mathcal{L}_\phi = \sum_{n\in\mathcal{D}} \log p_\phi(\hat{\mathbf{y}}_n|\hat{\mathbf{y}}_{\mathcal{P}_\mathbf{n}}\mathbf{X}) \tag{9}$$

Practically, the process is as follows: we first use q_θ to predict the class probabilities of pixels on the unlabeled (test) surface, aggregate those pixels classes into contours, and then use contour classes to re-train the GNN model.

4 EVALUATION

The goal of the evaluation is to compare our proposed method with baseline methods in classification performance on two real-world flood mapping datasets. We compared the methods on different numbers of training labels to evaluate the effectiveness of semi-supervised learning. We also conducted self-comparison studies to evaluate the effect of different hyperparameters in our model. Experiments were conducted on a workstation with four NVIDIA RTX 6000 GPUs (each with 24GB memory) installed with Keras and Tensorflow. The candidate methods

in our comparison are listed below. There are four categories: base models, base models with self-training, b models with the **conditional random field (CRF)** on the contour tree topological skeleton, and base models with graph co-training (our proposed approach). We considered U-Net and DeepLab as base models when evaluating the effect of existing self-training and our proposed graph (contour tree) co-training.

- **Base models:**
  - **U-Net:** We used the U-Net model with a 224 by 224 input shape implemented in Keras [44]. The U-Net model consists of an encoder-decoder structure. The encoder has six double-convolution layers and five max-pooling layers. The number of output filters in each convolution layer is 32, 64, 128, 256, 512, 1024. There is a batch normalization operation within each convolutional layer before non-linear activation based on **ReLU (rectified linear unit)**. The decoder of the model upsamples the encoded feature map to higher resolution with transposed convolution and concatenates upsampled features with corresponding feature maps from the encoder.
  - **DeepLabv3+:** We used the DeepLabv3+ model [9] with an input shape of 224 by 224. It was implemented in Keras.[1] The DeepLabv3+ model has an encoder-decoder structure. The encoder applies Atrous Spatial Pyramid Pooling with four different rates to detect multiple-scale features. The encoder feature output stride is 16. Then the encoder features are first bilinearly upsampled by a factor of 4 and then concatenated with the corresponding low-level features from backbone CNN. Then another bilinear upsampling by a factor of 4 is applied to obtain the original resolution. The network backbone we used is Xception model.
- **Base models with self-training:** Specifically, we used a pre-trained U-Net or DeepLabv3+ model to make predictions on unlabeled image patches. Those patches with high confidence predictions were added into the training set for the next iteration. The iteration continues until the performance on the validation set stops improving.
- **Base models with the conditional random field (CRF) on the contour tree topological skeleton:** We used the base segmentation model (e.g., U-Net, DeepLab) to infer per-pixel class probability and then fed the per-pixel class probabilities as unary energy into a CRF on the surface contour tree (the similar idea was proposed in [55]). The pair-wise energy is defined to encourage nearby similar pixels to have the same predicted label. In order to consider the topology dependency in our problem and do a fair comparison, we applied the CRF in the contour tree structure instead of a grid graph of the image. The CRF model was implemented in Matlab.[2]
- **Base models with graph co-training on the contour tree (our approach):** We co-trained a U-Net or DeepLab model and a GNN in variational EM. The codes were implemented in Tensorflow.

Dataset description: We used two real-world flood mapping datasets collected from North Carolina during Hurricane Matthew in 2016. Explanatory features were red, green, and blue bands in aerial imagery from the NOAA National Geodetic Survey [34]. The digital elevation imagery was downloaded from the University of North Carolina Libraries [36]. All data were resampled into a 2 meter by 2-meter resolution. For the U-Net model, we partitioned a surface into 224 by 224 patches. In the first dataset, the complete training surface contains 171 square patches, the validation surface contains 120 patches, and the test surface contains 132 patches. In the second dataset,

[1] https://github.com/rishizek/tensorflow-deeplab-v3-plus.
[2] http://www.cs.ubc.ca/~schmidtm/Software/UGM.html.

the complete training surface contains 416 patches, the validation surface contains 120 patches, and the test surface contains 208 patches.

The evaluation metrics used include precision, recall, F-score, and overall accuracy.

4.1 Details in Model Training

Model Architectures: For U-Net, the model consists of five double convolutional layers and max-pooling layers in the downsample path as well as five double convolutional layers and transposed convolutional layers in the upsample path. There is a batch normalization operation within each convolutional layer before non-linear activation based on **ReLU (rectified linear unit)**. For DeepLabv3+, we used Xception as the network backbone and atrous separable convolutions in encoder-decoder.

Hyperparameter: In model training, we used Adam optimizer and binary cross-entropy loss for all base segmentation models. The mini-batch size was 5. For U-Net, the learning rate was 10^{-4}, We trained the model for 300 epochs. For DeepLabv3+, the learning rate was 10^{-2}. For our approach, in the CNN part, we used the same model architecture and training methods as the baseline U-Net and DeepLabv3+ model. The weight λ to control the relevant importance of the supervised learning objective is set as 1, i.e., the weights of the labeled and unlabeled data are the same. In the graph neural network, we used the GCN model with the loss function based on sparse softmax cross-entropy with logits in Tensorflow. We used the MomentumOptimizer with a momentum of 0.9, a learning rate of 10^{-4}, a decaying rate of 0.99, and L_2 regularization with a weight of 10^{-3}. We also added a batch normalization layer before the non-linear activation in each graph convolution layer. There was a total of three GCN layers. Each GCN layer has 32 output filters. We trained the GCN model for 150 epochs. The mini-batch size was 1 since the graph topology of the input patches was not the same. The selection of hyper-parameters was based on data characteristics and the evaluation of validation data. For EM iterations, the convergence threshold is 0.001 (the iteration will stop if there is less than 0.1% improvement on validation accuracy).

Self-training: For U-Net or DeepLabv3+ with self-training, we first pre-trained the U-Net or DeepLabv3+ model based on a small set of labeled training patches to make predictions on unlabeled image patches in the test area. Those image patches with high confidence predictions were added into the training set to re-train the U-Net model. The confidence of one image patch was calculated based on the average confidence over all pixels (predicted class probabilities) in the patch. The confidence thresholds used in the first dataset were 0.85, 0.85, 0.90, 0.90, and 0.95, corresponding to 5, 10, 30, 60, and 171 training patches, respectively. In the second dataset, the confidence thresholds used were 0.85, 0.90, 0.90, 0.90, and 0.95 for 5, 20, 60, 120, and 416 training patches, respectively.

4.2 Comparison on Classification Accuracy

We first compared the classification performance of three approaches on the two datasets. We chose U-Net and DeepLab as representative base models in self-training and graph co-training due to their superior performance. To test the effectiveness of semi-supervised learning, we chose a sub-area containing only five patches from the training surface in each dataset. The results were summarized in Tables 2 and 3, respectively. On the first dataset, we can see that U-Net performed poorly when the training set was small (with an accuracy of 0.68). The reason was that the explanatory features on the surface contained a large number of obstacles that often confused a classifier. More importantly, the number of training labels was very small. DeepLabv3+ could capture long-range spatial context and achieved better performance than U-Net with an accuracy of 0.72, but it still did not full resolved the spectral confusion issue. After using self-training, the overall classification accuracy of U-Net improved from 0.68 to 0.76, but the performance on the dry class was

Table 2. Overall Comparison on Dataset 1

Methods	Class	Prec.	Recall	F	Avg. F	Accuracy
U-Net	Dry	0.61	0.74	0.67	0.68	0.68
	Flood	0.76	0.63	0.69		
U-Net with self-training	Dry	0.82	0.58	0.68	0.75	0.76
	Flood	0.73	0.90	0.81		
U-Net with CRF on contour tree	Dry	0.91	0.76	0.83	0.86	0.86
	Flood	0.83	0.94	0.88		
U-Net with graph co-training on contour tree	Dry	0.98	0.85	0.91	**0.92**	**0.92**
	Flood	0.89	0.98	0.94		
DeepLabv3+	Dry	0.85	0.46	0.61	0.71	0.72
	Flood	0.69	0.94	0.80		
DeepLabv3+ with self-training	Dry	0.80	0.69	0.75	0.80	0.80
	Flood	0.78	0.88	0.84		
DeepLabv3+ with CRF on contour tree	Dry	0.98	0.73	0.84	0.87	0.88
	Flood	0.82	0.99	0.90		
DeepLabv3+ with graph co-training on contour tree	Dry	0.98	0.76	0.88	**0.91**	**0.91**
	Flood	0.84	0.99	0.92		

Table 3. Overall Comparison on Dataset 2

Methods	Class	Prec.	Recall	F	Avg. F	Accuracy
U-Net	Dry	0.70	0.98	0.82	0.75	0.76
	Flood	0.97	0.51	0.67		
U-Net with self-training	Dry	0.88	0.80	0.83	0.83	0.83
	Flood	0.79	0.86	0.82		
U-Net with CRF on contour tree	Dry	0.98	0.85	0.91	0.92	0.92
	Flood	0.86	0.98	0.92		
U-Net with graph co-training on contour tree	Dry	0.97	0.98	0.97	**0.97**	**0.97**
	Flood	0.98	0.96	0.97		
DeepLabv3+	Dry	0.83	0.99	0.90	0.88	0.88
	Flood	0.98	0.76	0.86		
DeepLabv3+ with self-training	Dry	0.95	0.97	0.96	0.96	0.96
	Flood	0.96	0.95	0.95		
DeepLabv3+ with CRF on contour tree	Dry	0.96	0.97	0.97	**0.97**	**0.97**
	Flood	0.97	0.98	0.97		
DeepLabv3+ with graph co-training on contour tree	Dry	0.98	0.97	0.97	**0.97**	**0.97**
	Flood	0.97	0.98	0.97		

still poor (its F-score was around 0.68). The classification accuracy of DeepLabv3+ improved from 0.72 to 0.80 after adding self-training. But DeepLabv3+ with self-training still had errors since the "high confidence" predictions on test area that were added into the training set could still contain errors, which somehow confused the model. Moreover, self-training cannot capture the global 3D topological structure on the surface. U-Net and DeepLabv3+ with CRF on a contour tree dramatically improved the accuracy to around 0.86 and 0.88. The improvement is due to the fact that the model considered the topological dependency in the contour tree and predict topologically continuous surface. However, the model prediction still gave a relatively low precision for the flood class due to false positives from the deep learning predictions. In contrast, our model performed the best due to explicitly modeling topological structural constraints and using graph co-training to enhance unlabeled patches. Its overall accuracy was 0.91 with good performance in both classes. We can observe similar trends in the second dataset. From results on the second dataset in Table 3, we can see that U-Net alone achieved an overall accuracy of 0.76 and DeepLabv3+ achieved an accuracy of 0.88 (which is somehow better than the first dataset due to training patches being

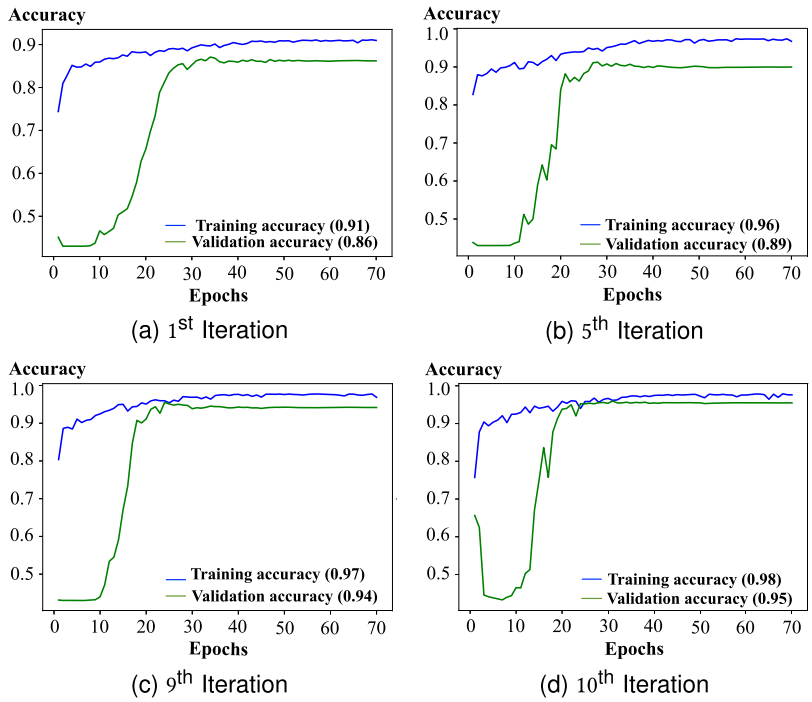

Fig. 4. Training curves of different EM iterations on Dataset 1.

more representative of the test area). Adding self-training to U-Net improved the accuracy to 0.83 due to a better recall in the flood class. Adding self-training to DeepLabv3+ improved its accuracy to 0.96. Such a great performance in self-training was likely due to the stronger base model (0.88 overall accuracy in DeepLabv3+ itself) on the second dataset. The results indicate that self-training heavily relies on the performance of base models because it requires accurate predictions on test data in training set expansion. The DeepLabv3+ model with CRF on contour tree achieved an F1-score of 0.97 while the U-Net model with CRF achieved an F1-score of 0.92. The U-Net model with CRF shows a lower precision for the flood class. In contrast, our approach persistently performed the best with an overall accuracy of 0.97 with both U-Net and DeepLabv3+ base models because it could both capture the topological structural dependency between surface locations and better utilize unlabeled image patches through graph neural network co-training. In summary, the results showed that explicitly modeling the topological structural constraint in graph co-training significantly boosted the classification performance when the training set was small.

For two datasets we also compared the learning curves in different EM iterations as shown in Figures 4 and 5. For both datasets, on the first EM iteration, we can observe a big gap between the training and validation curve. This is because the labeled dataset is small, and there is a significant overfitting issue for U-Net training. The issue can be mitigated after we incorporated topological dependency with the GNN model. In the next several iterations, the gap between the training curves and validation curves decreases, and the validation accuracy improves a lot. At the last iteration, the model converges to an optimum validation accuracy. We also observed some interesting patterns in the learning curve. For example, there was a drop of validation accuracy in the first few epochs of Figure 4(d). The pattern could be due to randomness since the model training was not stabilized yet in the first few epochs. The same trend was not observed in the second dataset (Figure 5(d)).

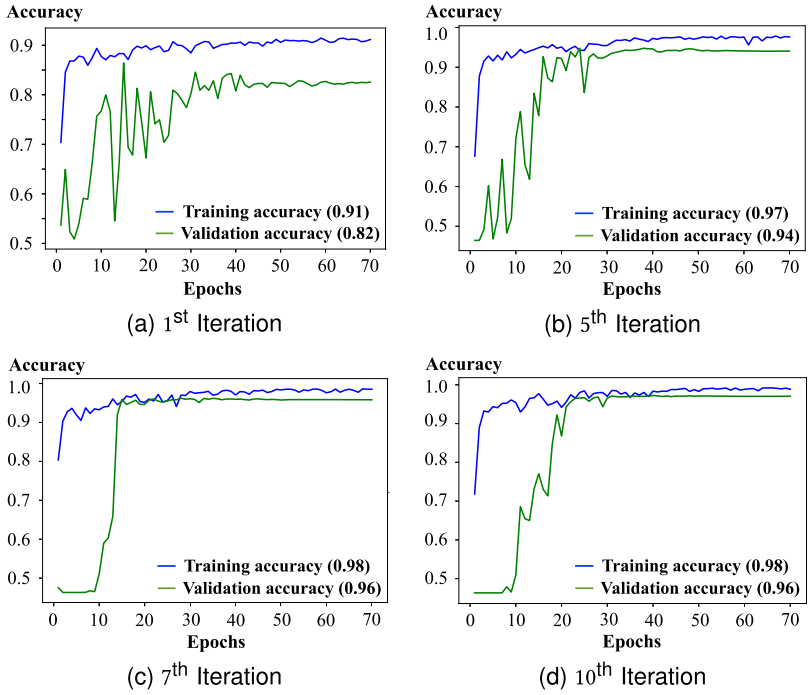

Fig. 5. Training curves of different EM iterations on Dataset 2.

4.3 The Effect of the Number of Training Patches

To fully evaluate the effectiveness of the semi-supervised learning methods, we compared U-Net, FCN, SegNet, and DeepLabv3+ base models with our approach (U-Net or DeepLabv3+ with graph co-training) on different amounts of training patches. We used the U-Net model as a representative base model in self-training and contour tree graph co-training. The FCN and SegNet model details are as follows:

• **FCN:** We used the basic FCN-32s model [30] implemented in Keras.[3] The model consists of convolutional layers with max-pooling and one final bilinear upsampling layer.

• **SegNet:** We used the SegNet model [2] implemented in Keras.[4] The model consists of an encoder and a decoder. The decoder upsamples feature layers based on unpooling operations.

On the first dataset, we increased the size of the training surface from 5 patches to 171 patches (the complete training surface). The overall accuracy of the three candidate methods was plotted in Figure 6. The results on the first dataset were shown in Figure 6(a). We can see that as the size of the training surface increased, the averaged F-score improved in all baselines methods, but the performance of our model persistently outperformed the other two baseline methods. Specifically, when the number of training patches increased from 5 to 20 and 40, the overall accuracy of U-Net improved from 0.68 to 0.81 and 0.86, and other segmentation models SegNet, FCN, and DeepLabv3+ improved from around 0.70 to 87. In general, DeepLabv3+ performed the best compared with other base models because the atrous convolution can extract dense feature maps and capture long-range

[3]https://github.com/aurora95/Keras-FCN.
[4]https://github.com/ykamikawa/tf-keras-SegNet.

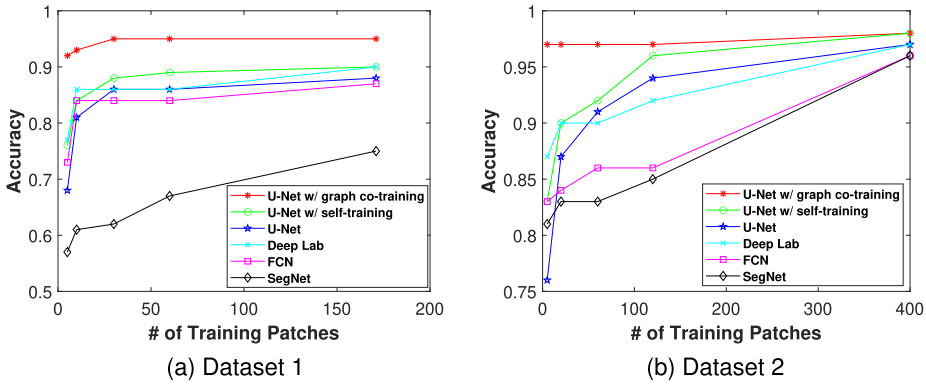

Fig. 6. The comparison of three methods on different sizes of the training set.

spatial context. SegNet performed the worst because the upsampling with unpooling operations may not generalize well to the test area. The accuracy of U-Net with self-training improved from 0.76 to 0.84 and 0.88, but the improvement was not significant compared with our model. The reason was that the self-training method still relied on the assumption that its high-confidence predictions were actually correct, which was not always true. The performance of models reached a plateau after the number of training patches reached around 40 or 60. The maximum accuracy of U-Net only, U-Net with self-training, and our model were 0.88, 0.90, and 0.96, respectively.

The results on the second dataset are shown in Figure 6(b). We can see similar trends that the performance of all baseline methods improved as the training set size increases. Our method significantly outperformed the other methods when the number of training patches was small (e.g., below 60). This showed that our method was more effective in addressing limited training labels. However, we also observed that when the number of training patches was sufficiently large (e.g., around 120 or above), the performance of the U-Net with self-training and DeepLabv3+ model also caught up, with an optimal F-score around 0.98 (close to our method). FCN and SegNet models showed around a 0.95 optimal F-score. We also observed that after we continued increasing the training patches, all the models converged at an optimal F-score around 0.98, which is not plotted in Figure 6(b). This showed that the second dataset was relatively easier to classify compared with the first dataset.

4.4 The Effect of GNN Model Configurations

Next, we conducted several self-comparisons to test the effect of different configurations of the GNN model in our method. We did experiments on the first dataset with 60 training patches and 132 test patches. We evaluated the effect of the number of output channels in each graph convolutional layer, the number of neighbor hops in each graph convolutional layer, the total number of graph convolutional layers in the GNN, as well as the type of graph convolutional kernels. When evaluating the effect of the number of output channels in each graph convolutional layer in our GNN model, we fixed the number of neighbor hops in each graph convolutional layer as 1, used three diffusion graph convolutional layers, and increased the number of output channels from 16 to 64. Results in Figure 7(a) show that the effect of the number of output channels was not significant in the final accuracy, and the optimal accuracy was achieved when the number of output channels was 32 and 64. When evaluating the effect of the number of neighbor hops in each graph convolutional layer in our GNN model, we used diffusion graph convolution, fixed the number of output channels to 32, and increased the number of neighbor hops from 1 to 3. Results in

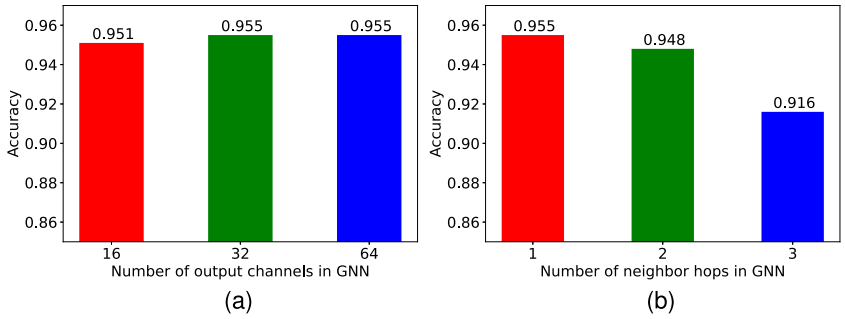

Fig. 7. The effect of the number of output channels and neighbor hops in a graph convolution layer.

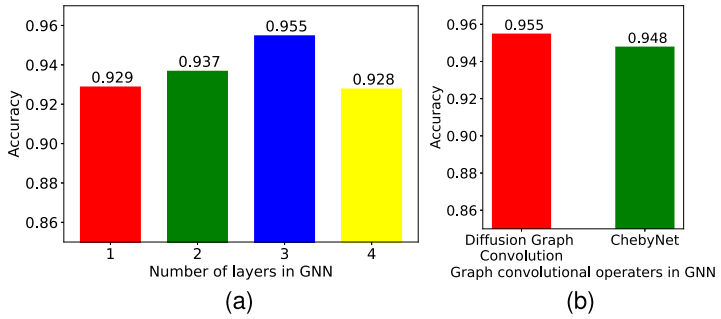

Fig. 8. The effect of the number (a) and the type (b) of graph convolution layers.

Figure 7(b) show that when increasing the number of neighbor hops, the final accuracy decreases. The model can achieve the best performance with 1-hop neighbors in graph convolution.

When evaluating the effect of the number of graph convolutional layers in our GNN model, we fixed the diffusion graph convolution output channel as 32, used 1-hop neighbor, and increased the number of convolutional layers from 1 to 4. Results in Figure 8(a) show that increasing graph convolutional layers can increase model performance, and achieve the best performance with three layers because with more layers the model can well model the higher-order topological structure constraint. When evaluating the effect of the type convolutional filters in our GNN model, we fixed the number of graph convolution layers as 3, the number of output channels as 32, used 1-hop neighbors and compared two types of graph convolution: diffusion graph convolution [29] and ChebyNet [11]. Results in Figure 8(b) show that diffusion graph convolution can achieve better performance than ChebyNet because diffusion graph convolution can model the directional topological dependency.

4.5 Case Study

We also conducted a case study to interpret the performance of different models through visualization. In Figures 9 and 10, we provide the aerial image, digital elevation, and the ground truth class labels in the two datasets, as well as the predictions of U-Net, U-Net with self-training, and U-Net with graph co-training (our model) on two datasets with five training patches. Figure 9(a) shows the input surface features as spectral bands of the earth imagery. From the image, we can see that the lower half of the area is flooded (in brown color) but the flooded area was obscured by tree canopies (in green color). The tree canopies created spectral confusion for classifiers since

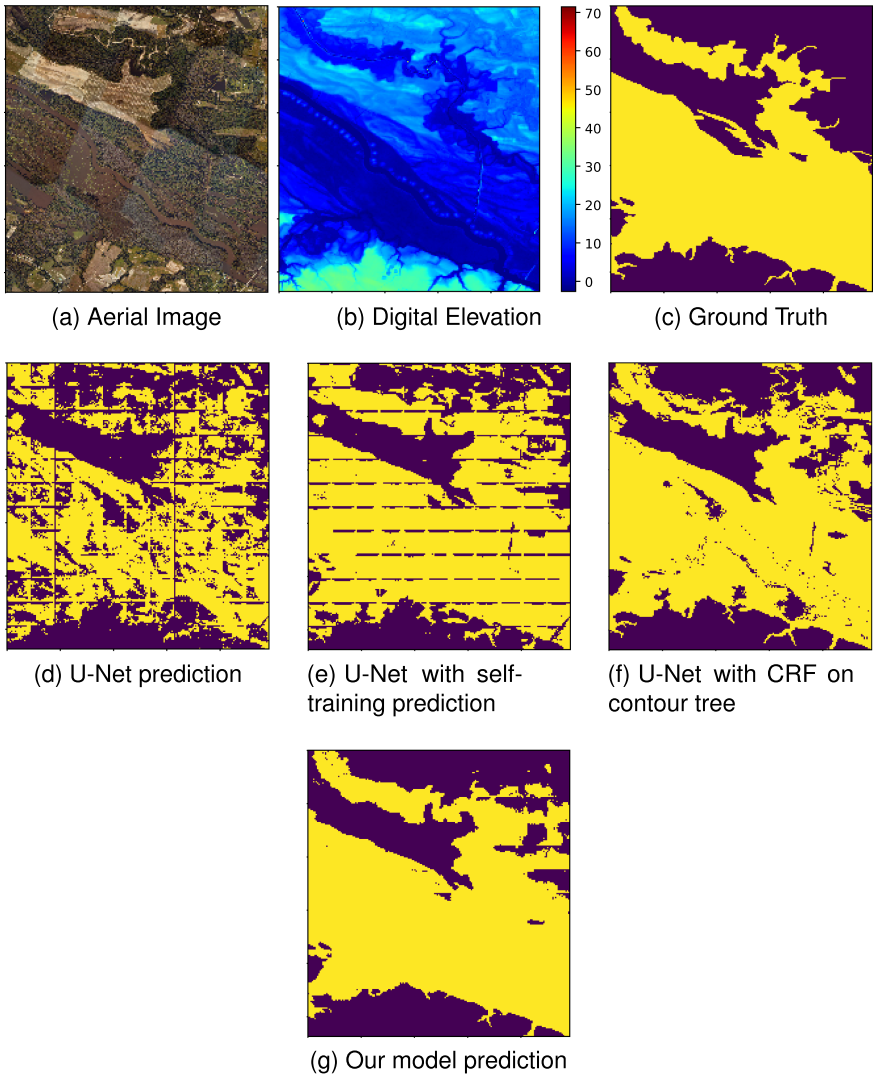

Fig. 9. The aerial image (a), digital elevation (b), ground truth (c), U-Net prediction (d), U-Net with self-training prediction (e), U-Net with CRF on contour tree prediction (f), and our model prediction (g) on Dataset 1 (Yellow is flood class, purple is dry class).

the same spectral signatures also exist in the dry areas. Figure 9(b) shows the elevation surface and its topography. The U-Net prediction results in Figure 9(d) show much salt-and-pepper noise and misclassifications compared with the ground truth in Figure 9(c) due to spectral confusion among those area covered by obstacles (e.g., trees canopies). Moreover, the prediction class map was not smooth near the boundary of each image patch because U-Net classified each image patch (224×224) independently from each other and thus cannot capture the topological dependency across image patches. This limitation causes discontinuity in the prediction of the high-resolution test image. The U-Net with self-training in Figure 9(e) could alleviate the salt-and-pepper noise errors and make smoother classification, but the prediction results still show vertical and

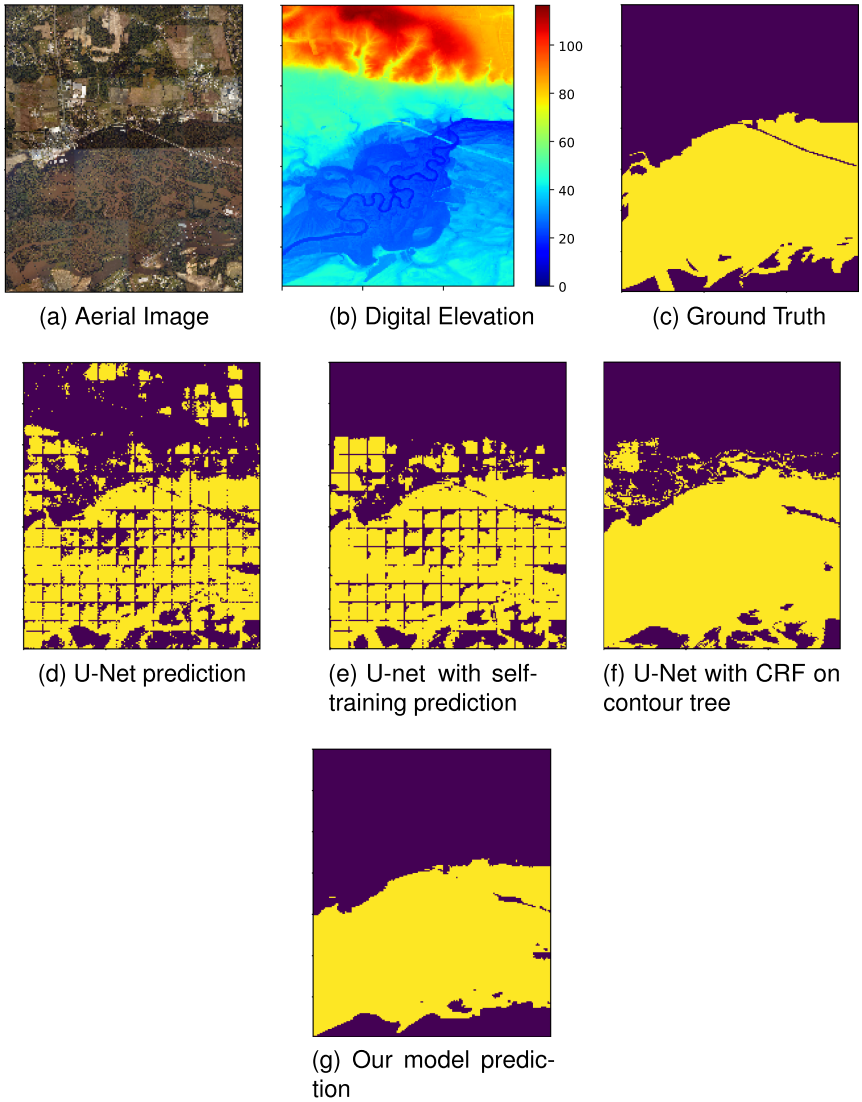

Fig. 10. The aerial image (a), digital elevation (b), ground truth (c), U-Net prediction (d), U-Net with self-training prediction (e), U-Net with CRF on contour tree prediction (f), and our model prediction (g) on Dataset 2 (Yellow is flood class, purple is dry class).

horizontal artifacts near patch boundaries. The U-Net model with CRF on the contour tree dramatically reduces the classification errors due to incorporating the topological structure (Figure 9(f)). But the results still contain errors when the initial classification is heavily erroneous. In contrast, our model with graph co-training in Figure 9(g) can capture the dependency across image patches, which gave smoother and more accurate predictions on the whole test image. Similar results are observed on the second dataset in Figure 10.

4.6 Analysis of Computational Time Costs

We evaluated the computational efficiency of our proposed approach. The experiments were conducted on our deep learning workstation with 4 NVIDIA RTX 6000 GPUs connected by NV-Link

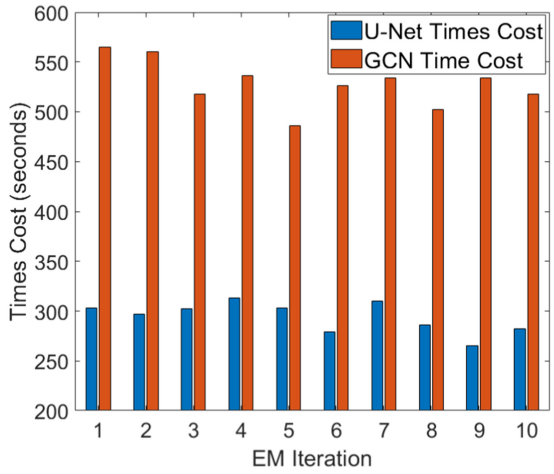

Fig. 11. Time cost of GCN model and U-Net model training during different iterations. (best viewed in color).

(each GPU has 24GB memory). The total number of unlabeled input image patches in the test area for CNN is 132. The time costs in different EM iterations were summarized in Figure 11. The blue bar and red bar showed the time costs of the GCN model training and the U-Net model training within each EM iteration. From the results, we can see that the GCN model training took about half of the time cost of U-Net training and their time costs were relatively stable across EM iterations. Each iteration took about 15 minutes. The numbers were highly dependent on the hardware platform. We acknowledge that the time cost of our approach is higher than the baseline of a single U-Net training, but our model can address the limited training sample issue and take advantage of the topological structure of the terrain surface.

5 CONCLUSION AND FUTURE WORKS

This paper focuses on the problem of earth imagery segmentation on terrain surface with limited training labels. The problem is important for many applications such as water surface mapping in hydrology but is challenging due to the existence of topological structure and limited training labels. Existing methods are often limited in not explicitly modeling the topological structural dependency on the class surface. In contrast, we proposed a new method that represents the topological surface as a contour tree skeleton based on the physics of water flow directions. Our method co-trains a convolutional neural network on image patches and a graph neural network on the contour tree. Evaluations on real-world flood mapping datasets show that our method significantly outperforms baselines, especially when the size of the training surface is small.

In future work, we plan to expand our evaluations to more applications such as rock art surface segmentation in archaeology and protein elevation surface segmentation. We also plan to generalize our ideas from transductive learning to inductive learning.

REFERENCES

[1] Bjoern Andres, Jörg H. Kappes, Thorsten Beier, Ullrich Köthe, and Fred A. Hamprecht. 2011. Probabilistic image segmentation with closedness constraints. In *2011 International Conference on Computer Vision*. IEEE, 2611–2618.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 12 (2017), 2481–2495.

[3] Jonathan T. Barron and Ben Poole. 2016. The fast bilateral solver. In *European Conference on Computer Vision*. Springer, 617–632.
[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.
[5] Hamish Carr, Jack Snoeyink, and Ulrike Axen. 2003. Computing contour trees in all dimensions. *Computational Geometry* 24, 2 (2003), 75–94.
[6] Chao Chen, Daniel Freedman, and Christoph H. Lampert. 2011. Enforcing topological constraints in random field image segmentation. In *CVPR 2011*. IEEE, 2089–2096.
[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2017), 834–848.
[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587* (2017).
[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 801–818.
[10] Wuyang Chen, Ziyu Jiang, Zhangyang Wang, Kexin Cui, and Xiaoning Qian. 2019. Collaborative global-local networks for memory-efficient segmentation of ultra-high resolution images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8924–8933.
[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.
[12] Emre Eftelioglu, Zhe Jiang, Reem Ali, and Shashi Shekhar. 2016. Spatial computing perspective on food energy and water nexus. *Journal of Environmental Studies and Sciences* 6, 1 (2016), 62–76.
[13] Emre Eftelioglu, Zhe Jiang, Xun Tang, and Shashi Shekhar. 2017. The nexus of food, energy, and water resources: Visions and challenges in spatial computing. In *Advances in Geocomputation*. Springer, 5–20.
[14] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. 2016. Fusenet: Incorporating depth into semantic segmentation via fusion-based CNN architecture. In *Asian Conference on Computer Vision*. Springer, 213–228.
[15] Wenchong He and Zhe Jiang. 2020. Semi-supervised learning with the EM algorithm: A comparative study between unstructured and structured prediction. *IEEE Transactions on Knowledge and Data Engineering*.
[16] Wenchong He, Arpan Man Sainju, Zhe Jiang, and Da Yan. 2021. Deep neural network for 3D surface segmentation based on contour tree hierarchy. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 253–261.
[17] Yufan He, Aaron Carass, Yihao Liu, Bruno M. Jedynak, Sharon D. Solomon, Shiv Saidha, Peter A. Calabresi, and Jerry L. Prince. 2019. Deep learning based topology guaranteed surface and MME segmentation of multiple sclerosis subjects from retinal OCT. *Biomedical Optics Express* 10, 10 (2019), 5042–5058.
[18] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. 2019. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems*. 5658–5669.
[19] Zhe Jiang. 2018. A survey on spatial prediction methods. *IEEE Transactions on Knowledge and Data Engineering* 31, 9 (2018), 1645–1664.
[20] Zhe Jiang. 2020. Spatial structured prediction models: Applications, challenges, and techniques. *IEEE Access* 8 (2020), 38714–38727.
[21] Zhe Jiang and Arpan Man Sainju. 2019. Hidden Markov contour tree: A spatial structured model for hydrological applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 804–813.
[22] Zhe Jiang and Arpan Man Sainju. 2021. A hidden Markov tree model for flood extent mapping in heavily vegetated areas based on high resolution aerial imagery and DEM: A case study on Hurricane Matthew floods. *International Journal of Remote Sensing* 42, 3 (2021), 1160–1179.
[23] Zhe Jiang and Shashi Shekhar. 2017. *Spatial Big Data Science*. Schweiz: Springer International Publishing AG.
[24] Zhe Jiang, Miao Xie, and Arpan Man Sainju. 2021. Geographical hidden Markov tree. *IEEE Transactions on Knowledge and Data Engineering* 33, 2 (2021), 506–520.
[25] Tarun Kalluri, Girish Varma, Manmohan Chandraker, and C. V. Jawahar. 2019. Universal semi-supervised semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*. 5259–5270.
[26] Anuj Karpatne, Zhe Jiang, Ranga Raju Vatsavai, Shashi Shekhar, and Vipin Kumar. 2016. Monitoring land-cover changes: A machine-learning perspective. *IEEE Geoscience and Remote Sensing Magazine* 4, 2 (2016), 8–21.
[27] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[28] Andrew Lang, Aaron Carass, Ava K. Bittner, Howard S. Ying, and Jerry L. Prince. 2017. Improving graph-based OCT segmentation for severe pathology in Retinitis Pigmentosa patients. In *Medical Imaging 2017: Biomedical Applications in Molecular, Structural, and Functional Imaging*, Vol. 10137. International Society for Optics and Photonics, 101371M.

[29] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[31] Lingni Ma, Jörg Stückler, Christian Kerl, and Daniel Cremers. 2017. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 598–605.

[32] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2020. Image segmentation using deep learning: a survey. *arXiv preprint arXiv:2001.05566* (2020).

[33] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. 2018. Beyond the pixel-wise loss for topology-aware delineation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3136–3145.

[34] National Oceanic and Atmospheric Administration. [n.d.]. Data and Imagery from NOAA's National Geodetic Survey. https://www.ngs.noaa.gov.

[35] National Oceanic and Atmospheric Administration. 2018. National Water Model: Improving NOAA's Water Prediction Services. http://water.noaa.gov/documents/wrn-national-water-model.pdf.

[36] NCSU Libraries. 2018. LIDAR Based Elevation Data for North Carolina. https://www.lib.ncsu.edu/gis/elevation.

[37] Radford M. Neal and Geoffrey E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*. Springer, 355–368.

[38] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*. 1520–1528.

[39] Sebastian Nowozin and Christoph H. Lampert. 2009. Global connectivity potentials for random field models. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 818–825.

[40] Manfred Opper and David Saad. 2001. *Advanced Mean Field Methods: Theory and Practice*. MIT press.

[41] Yassine Ouali, Céline Hudelot, and Myriam Tami. 2020. Semi-supervised semantic segmentation with cross-consistency training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12674–12684.

[42] Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. 2007. Robust on-line computation of Reeb graphs: Simplicity and speed. In *ACM SIGGRAPH 2007 papers*. 58–es.

[43] Meng Qu, Yoshua Bengio, and Jian Tang. 2019. GMNN: Graph Markov neural networks. *arXiv preprint arXiv:1905.06214* (2019).

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 234–241.

[45] Arpan Man Sainju, Wenchong He, and Zhe Jiang. 2020. A hidden Markov contour tree model for spatial structured prediction. *IEEE Transactions on Knowledge and Data Engineering*.

[46] Arpan Man Sainju, Wenchong He, Zhe Jiang, and Da Yan. 2020. Spatial classification with limited observations based on physics-aware structural constraint. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 898–905.

[47] Arpan Man Sainju, Wenchong He, Zhe Jiang, Da Yan, and Haiquan Chen. 2021. Flood inundation mapping with limited observations based on physics-aware topography constraint. *Frontiers in Big Data* 4 (2021), 47. https://doi.org/10.3389/fdata.2021.707951

[48] Shashi Shekhar, Zhe Jiang, Reem Y. Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. 2015. Spatiotemporal data mining: A computational perspective. *ISPRS International Journal of Geo-Information* 4, 4 (2015), 2306–2338.

[49] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. 2008. Graph cut based image segmentation with connectivity priors. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

[50] Jinghua Wang, Zhenhua Wang, Dacheng Tao, Simon See, and Gang Wang. 2016. Learning common and specific features for RGB-D semantic segmentation with deconvolutional networks. In *European Conference on Computer Vision*. Springer, 664–679.

[51] Senzhang Wang, Jiannong Cao, and Philip Yu. 2020. Deep learning for spatio-temporal data mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

[52] Weiyue Wang and Ulrich Neumann. 2018. Depth-aware CNN for RGB-D segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 135–150.

[53] Miao Xie, Zhe Jiang, and Arpan Man Sainju. 2018. Geographical hidden Markov tree for flood extent mapping. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (London, United Kingdom) *(KDD'18)*. ACM, New York, NY, USA, 2545–2554.

[54] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. 2018. ICNet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 405–420.

[55] Leixin Zhou, Zisha Zhong, Abhay Shah, Bensheng Qiu, John Buatti, and Xiaodong Wu. 2019. Deep neural networks for surface segmentation meet conditional random fields. *arXiv preprint arXiv:1906.04714* (2019).

Received December 2020; revised March 2021; accepted August 2021