Weakly Supervised Spatial Deep Learning for Earth Image Segmentation Based on Imperfect Polyline Labels

ZHE JIANG and WENCHONG HE, Department of Computer & Information Science & Engineering, The University of Florida, USA
MARCUS STEPHEN KIRBY, Department of Computer Science, The University of Alabama, USA
ARPAN MAN SAINJU, Department of Computer Science, Middle Tennessee State University, USA
SHAOWEN WANG, Department of Geography and Geographic Information Science, The University of Illinois at Urbana-Champaign, USA
LAWRENCE V. STANISLAWSKI, ETHAN J. SHAVERS, and E. LYNN USERY, U.S. Geological Survey, Center of Excellence for Geospatial Information Science, USA

In recent years, deep learning has achieved tremendous success in image segmentation for computer vision applications. The performance of these models heavily relies on the availability of large-scale high-quality training labels (e.g., PASCAL VOC 2012). Unfortunately, such large-scale high-quality training data are often unavailable in many real-world spatial or spatiotemporal problems in earth science and remote sensing (e.g., mapping the nationwide river streams for water resource management). Although extensive efforts have been made to reduce the reliance on labeled data (e.g., semi-supervised or unsupervised learning, few-shot learning), the complex nature of geographic data such as spatial heterogeneity still requires sufficient training labels when transferring a pre-trained model from one region to another. On the other hand, it is often much easier to collect lower-quality training labels with imperfect alignment with earth imagery pixels (e.g., through interpreting coarse imagery by non-expert volunteers). However, directly training a deep neural network on imperfect labels with geometric annotation errors could significantly impact model performance. Existing research that overcomes imperfect training labels either focuses on errors in label class semantics or characterizes label location errors at the pixel level. These methods do not fully incorporate the geometric properties of label location errors in the vector representation. To fill the gap, this article proposes a weakly supervised learning framework to simultaneously update deep learning model parameters and infer hidden true vector label locations. Specifically, we model label location errors in the vector representation to partially reserve geometric properties (e.g., spatial contiguity within line segments). Evaluations on real-world datasets in the National Hydrography Dataset (NHD) refinement application illustrate that the proposed framework outperforms baseline methods in classification accuracy.

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). (Disclaimer: Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. government.)
Authors' addresses: Z. Jiang (corresponding author) and W. He, Department of Computer & Information Science & Engineering, The University of Florida, P.O. Box 116120, Gainesville, FL, 32611; emails: {zhe.jiang, whe2}@ufl.edu; M. S. Kirby, Department of Computer Science, The University of Alabama, 35487; A. M. Sainju, Department of Computer Science, Middle Tennessee State University, Murfreesboro, TN, 37132; email: asainju@mtsu.edu; S. Wang, Department of Geography and Geographic Information Science, The University of Illinois at Urbana-Champaign; email: shaowen@illinois.edu; L. V. Stanislawski, E. J. Shavers, and E. L. Usery, U.S. Geological Survey, Center of Excellence for Geospatial Information Science; emails: {lstan, eshavers, usery}@usgs.gov.

2157-6904/2021/12-ART25 $15.00
https://doi.org/10.1145/3480970

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

Additional Key Words and Phrases: Deep learning, earth imagery segmentation, imperfect labels, weakly supervised learning

ACM Reference format:
Zhe Jiang, Wenchong He, Marcus Stephen Kirby, Arpan Man Sainju, Shaowen Wang, Lawrence V. Stanislawski, Ethan J. Shavers, and E. Lynn Usery. 2021. Weakly Supervised Spatial Deep Learning for Earth Image Segmentation Based on Imperfect Polyline Labels. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 25 (December 2021), 20 pages.
https://doi.org/10.1145/3480970

1 INTRODUCTION

In recent years, deep learning techniques (e.g., U-Net [50], DeepLab [8], SegNet [3]) have achieved tremendous success in image segmentation [18, 23, 41, 72]. The performance of these models heavily relies on large-scale high-quality ground-truth training labels (e.g., PASCAL VOC 2012 [16], Cityscapes [10]). Unfortunately, such large-scale high-quality training data are often unavailable in many real-world spatial or spatiotemporal problems in earth science and remote sensing due to expensive and time-consuming field surveys or visual interpretation of earth imagery [14, 15, 26, 30, 33, 56]. Indeed, lacking high-quality ground-truth benchmarking datasets has been identified as one of the major challenges in developing deep learning technologies for spatial and spatiotemporal data in earth science applications by several recent survey articles [4, 39, 60, 62, 74].

One important application is the **National Hydrography Dataset (NHD)** refinement. Surface water is an irreplaceable resource for human life and environmental sustainability. The NHD of the **U.S. Geological Survey (USGS)** is the current most up-to-date, comprehensive, and widely used dataset on surface water features (e.g., rivers, streams, canals, lakes) for the United States [57]. It serves as a cornerstone for many scientific applications, such as assessing the quantity and quality of present and future water resources, modeling climate changes, evaluating agricultural suitability, mapping flood inundation, and monitoring environmental changes [21, 27–29, 31, 40, 47, 51–54, 65, 67]. In recent years, with the increasingly available high-resolution remote sensing data (e.g., high-resolution optical imagery, 3D Lidar point clouds, radar imagery), the USGS has started refinements of the NHD to a higher resolution [44]. High-resolution NHD provides unique opportunities for scientific communities to study problems at fine scales that were not possible before, such as hyper-resolution national water forecasting and precision agriculture. The current refinement process involves training deep learning models on high-resolution remote sensing imagery to automatically delineate river stream channels [58, 68]. However, deep learning models often require manually annotating large amounts of high-quality training labels by well-trained experts, which is slow, tedious, and expensive [64]. Considering the problem at a national scale, the scarcity of high-quality training labels quickly becomes a major bottleneck.

Although extensive efforts have been made in the machine learning community to reduce the reliance on labeled data (e.g., semi-supervised [20, 73] or unsupervised learning [5], few-shot or zero-shot learning [63], co-training [24, 71], or meta-learning [25, 61]), the complex nature of geographic data such as spatial heterogeneity still requires sufficient training labels when transferring a pre-trained model from one region to another [26]. On the other hand, it is often much easier to collect lower-quality training labels by non-expert volunteers (e.g., Amazon Mechanical Turk, Tomnod.com by DigitalGlobe). However, such training labels often have geometric annotation

errors [30, 59]. These errors can be due to manual annotation mistakes, particularly if the annotators are non-experts and thus unable to fully interpret some image pixels or if the annotators interpret background imagery displayed at a coarse resolution (e.g., on a small screen of a smartphone) [11, 13, 19, 22, 26, 55, 56]. Annotation errors can also come from GPS errors when a field crew travels on the ground to delineate the boundary of a land parcel. These geometric annotation errors can significantly impact the effectiveness of existing deep learning algorithms.

Training effective deep learning models based on imperfect annotation labels on high-resolution earth imagery is non-trivial for several reasons. First, the annotation errors in vector labels lead to erroneous class labels of training pixels. This issue may not cause much trouble for polygon segments where annotation errors only impact pixels near segment boundaries, but it is critical for polyline labels since an error of vector location may lead to a completely different set of pixels being labeled as the positive class (e.g., river streams). Second, the problem requires a learning algorithm to infer true label locations and train neural network parameters simultaneously. In other words, there are both unknown deep neural network parameters and hidden true label locations. Furthermore, there are other challenges related to the unique characteristics of earth imagery segmentation. For instance, the geographic space is continuous and often much larger than the input shape of most existing deep neural networks designed for regular camera photos (which is only a few hundred by a few hundred pixels). In addition, the input features contain more spectral channels or topographic layers than camera photos. Thus, we cannot directly transfer a pre-trained deep convolutional neural network (e.g., VGG16, ResNet50) based on camera photos to earth imagery datasets. In addition, the input feature imagery in remote sensing data is often noisy and may not indicate perfect line patterns. Finally, the problem is computationally intensive due to a large number of potential true polyline label locations in continuous space, as a polyline label is determined by the combination of its vertices.

Existing research that addresses training label errors often focuses on errors in label class semantics, assuming label locations to be correct or irrelevant (e.g., samples are independent and identically distributed). Techniques include simple data cleaning to filter noise [17], choosing relatively noise-tolerant models [1, 12], designing robust loss function [42, 46, 48], and learning noise distribution [32, 38, 49, 66, 66]. Thus, these techniques cannot address the location errors in the ground-truth labels. There are a few studies on label location errors in image segmentation [2, 7, 70]. These methods often focus on jointly refining misplaced pixels on object edges or boundaries (e.g., active contour based on the level set method [2], edge pixel matching [70]) and thus cannot be applied to our problem where the labels are buffered polylines (e.g., river streams) instead of object edges. [42] models location errors of polyline training labels as shifts of square pixel patches in eight-neighbor directions, but the use of square patches is too rigid for location errors in vector labels. There are other relevant works that focus on capturing label ambiguity or uncertainty [34–36, 43]. The main difference is that these works assume the training labels to be non-unanimous due to different opinions of several experts, and all of the different opinions are considered correct. In contrast, we assume that there is a single correct true label location (that is unknown and not perfectly aligned with the given label location). There are also works focusing on generative models for vector sequences (e.g., synthetic vector graphics [6, 37], trajectories [45]). Other works rely on interactive active learning to address imperfect labels [69], but this approach requires human experts in the loop.

This article proposes a weakly supervised spatial deep learning framework for earth imagery segmentation based on imperfect polyline vector labels with geometric annotation errors. Specifically, we directly model label location errors in the vector representation. To partially maintain the geometric properties of polyline labels (e.g., contiguity within segments) and reduce the number of combinations of candidate true polyline locations, we represent a polyline label as a sequence

of independent line segments. We model the location error of each segment as shifting in the perpendicular direction. Such a design decision proves to be simple and effective. Based on the location error model, our framework jointly updates deep learning model parameters and infers hidden true label segment locations through iterations. Results on real-world remote sensing imagery data for NHD refinement show that our trained model has significantly higher accuracy than baseline methods.

2 PROBLEM STATEMENT

2.1 Preliminaries

Definition 2.1. A *spatial raster framework* is a tesselation of the two-dimensional (2D) plane into a regular m by n grid, where m and n are the numbers of rows and columns. Note that m and n are usually much larger than the input shape of a typical deep convolutional neural network, e.g., beyond a few thousand. Thus, we often need to cut the framework into square patches or windows (e.g., 224 by 224 pixels) as the input of a deep learning model.

Definition 2.2. In a raster framework, there exist multiple *explanatory feature layers*, denoted as $\mathbf{X} \in \mathbb{R}^{m \times n \times F}$, where F is the number of input feature channels. Examples are spectral bands of remote sensors (red, green, blue, near-infrared), digital elevation, and its topographic derivatives (slope, curvature).

Definition 2.3. There also exists a *class layer*, denoted as $\mathbf{Y} \in \{0, 1\}^{m \times n}$. For example, in the NHD refinement application, the two classes are *stream* and *non-stream*. For simplicity, we only consider binary classes in this article, but the study can potentially be generalized to multi-class scenarios.

Definition 2.4. A *polyline* is a connected sequence of line string segments, denoted by its vertices $\mathbf{L} =< \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{n_\mathbf{p}} >$, where $\mathbf{p}_i \in \mathbb{R}^{2\times 1}$ is the 2D coordinates of a vertex and $n_\mathbf{p}$ is the number of vertices in the polyline. Polylines provide a vector representation (alternative to the raster representation) of spatial class labels. For example, a river stream can be represented by a polyline along its center. We can add a *buffer* on a polyline to reflect the river width. The ground-truth labels often contain a set of polylines, denoted as $\mathcal{L} = \{\mathbf{L}^{(i)} | 1 \le i \le N\}$. In reality, it is often very slow and expensive to annotate polyline labels that are perfectly aligned with high-resolution earth imagery. Instead, we may be given a set of imperfect polylines, denoted by $\tilde{\mathcal{L}} = \{\tilde{\mathbf{L}}^{(i)} | 1 \le i \le N\}$, that are not well aligned with the true-positive class pixels.

Note that in the above definition, we denote a polyline as a sequence of varying-length line segments between vertices. Alternatively, we can also denote a polyline by a sequence of equal-length polyline segments, i.e., $\mathbf{L} =< \mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_{n_\mathbf{L}} >$.

Definition 2.5. We can convert class labels from the vector representation $\mathbf{L}$ to the raster representation $\mathbf{Y}$. The process is called *rasterization*. Specifically, after buffering a polyline, the rasterization process assigns all pixels that are covered by the buffered polyline in the positive class (e.g., stream) and leaves the background pixels in the negative class (e.g., non-stream). Thus, the raster representation $\mathbf{Y}$ and vector representation $\mathcal{L}$ are two alternative formats of class labels.

2.2 Problem Definition

Based on the definitions above, we now formally define the problem below.

Input:

- Explanatory feature layers $\mathbf{X}$
- Imperfect polyline labels in the vector format $\tilde{\mathcal{L}} = \{\tilde{\mathbf{L}}^{(i)} | 1 \le i \le N\}$

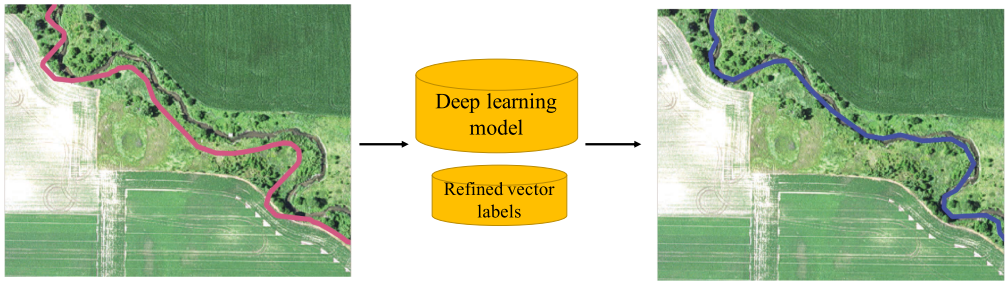


Fig. 1. A real-world problem example in streamline segmentation from high-resolution earth imagery.

- A fixed buffer width d for polyline labels
- A type of base image segmentation model, e.g., U-Net, DeepLab

Output:

- A deep learning model $\mathbf{Y} = f(\mathbf{X}, \Theta)$, where Θ is the set of parameters
- Refined polyline labels $\mathcal{L} = \{\mathbf{L}^{(i)} | 1 \leq i \leq N\}$ as a by-product

Objective:

- Maximize classification accuracy of the deep learning model f
- Maximize the quality of refined labels $\mathcal{L}$

Constraint:

- $\tilde{\mathcal{L}}$ has a maximum error distance Δ_{max}
- The input explanatory feature layers can be noisy

Figure 1 provides a real-world example in streamline segmentation from high-resolution earth imagery. The input ground-truth training labels (polyline in pink color) are misaligned with the true stream pixels (in gray color) in the earth imagery. Thus, directly training a deep learning model from the imperfect label will lead to poor classification performance. Given the earth imagery with an imperfect polyline label (in the left of Figure 1) as well as a base deep learning model (e.g., U-Net [50], DeepLab [8]), the problem aims to learn a deep learning model and refine the polyline label as a by-product (in the middle of Figure 1), such that the model class prediction is accurate (in the right of Figure 1).

3 THE PROPOSED APPROACH

This section introduces our proposed weakly supervised learning framework. In our framework, we assume that unobserved true polyline label locations $\mathbf{L}$ are hidden variables. Our goal is to update neural network parameters and infer the hidden true label locations at the same time. Our probabilistic formulation consists of two components: the relationship between observed imperfect label locations and the hidden true location and the relationship between hidden true label locations with the earth imagery features. The specific formulation is shown in Equation (1). In this equation, $P(\tilde{\mathbf{L}}|\mathbf{L})$ captures the distribution of location errors in the polyline label and $P(\mathbf{L}|\mathbf{X}, \Theta)$ captures the relationship between the true label and earth imagery features (such a relationship can be learned by a neural network with parameters Θ). Given an imperfect label location $\tilde{\mathbf{L}}$ and imagery features $\mathbf{X}$, our objective is to find the most likely true label location $\mathbf{L}$ and learn neural network parameters Θ at the same time. Note that here we express the probabilities of label locations without specifying the exact random vector definition of $\mathbf{L}$ and $\tilde{\mathbf{L}}$. Our purpose is to illustrate

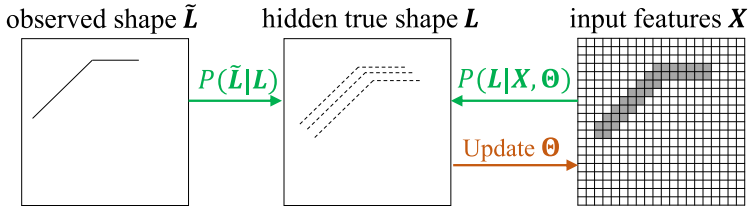


Fig. 2. Illustration of the proposed weakly supervised learning framework for polyline labels.

the main idea first and leave the details in Section 3.1 and Section 3.2.

$$\mathbf{L}, \Theta \leftarrow \arg\max_{\mathbf{L},\Theta} P(\tilde{\mathbf{L}}|\mathbf{L})P(\mathbf{L}|\mathbf{X},\Theta) \tag{1}$$

In order to solve the optimization problem in Equation (1), our framework uses an iterative approach. We can initialize neural network parameters by pre-training the model on input imperfect labels, as listed in step (1) below. Then we can iteratively infer the most likely true label locations and update neural network parameters, as listed as step (2) and step (3) below. We can repeat the iterations of steps (2) and (3) until the model converges (i.e., the validation performance no longer improves).

(1) Initialize neural network parameters Θ_0 from imperfect labels
(2) Fix Θ_0, refine true polyline location $\mathbf{L_0}$: $\mathbf{L_0} \leftarrow \arg\max_{\mathbf{L}} P(\tilde{\mathbf{L}}|\mathbf{L})P(\mathbf{L}|\mathbf{X},\Theta_0)$
(3) Fix $\mathbf{L_0}$, retrain neural network parameters Θ_0: $\Theta_0 \leftarrow \arg\max_{\Theta} P(\tilde{\mathbf{L}}|\mathbf{L_0})P(\mathbf{L_0}|\mathbf{X},\Theta)$

Our main idea is also illustrated in Figure 2. The input includes imperfect polyline label $\tilde{\mathbf{L}}$ (in the left of Figure 2) and explanatory feature layers $\mathbf{X}$ (in the right of Figure 2). There can be multiple possible candidate true label locations $\mathbf{L}$ (in the middle of Figure 2). Our objective is to infer the most likely true label location $\mathbf{L}$ and train the neural network parameters Θ at the same time. The iteration process is highlighted by the arrows. In each iteration, we first fix the neural network parameter Θ and infer the most likely true label location $\mathbf{L}$ among all candidates (the green arrows in Figure 2). Then we use the inferred most likely true label location $\mathbf{L}$ to update neural network parameters Θ for the next iteration (the red arrow in Figure 2). The iteration continues until the model converges (i.e., validation performance no long improves). In this example, we use a single polyline label for simplicity. The idea can be easily extended to a set of polyline labels $\tilde{\mathcal{L}}$ by assuming that different polyline labels are independent from each other.

However, several challenges exist in the implementation of the proposed framework. First, the number of possible true labels $\mathbf{L}$ is exponential to the number of vertices (actually, the number can be infinite considering that the space is continuous). This makes it very hard, if possible, to model $P(\tilde{\mathbf{L}}|\mathbf{L})$ in the continuous space. Thus, we have to discretize the location error of polyline labels. In other words, we can only consider a limited number of candidate true label locations for each observed imperfect label location. Second, when inferring true label locations, we also need to make a good balance between *exploration* (keeping a wide range of candidate true labels to avoid overfitting to a particular wrong label, especially when the neural network has not been well trained yet) and *exploitation* (focusing on a narrow range of candidate true labels for refinement). We discuss the implementation strategies to address the above challenges in the following subsections.

3.1 Statistical Model of Label Location Error $P(\tilde{\mathbf{L}}|\mathbf{L})$

This subsection aims to design statistical models for geometric errors between observed polyline labels and the hidden true polyline labels. For simplicity, we only introduce the model of $P(\tilde{\mathbf{L}}|\mathbf{L})$

for each individual polyline, assuming different polylines to be independent. This assumption is reasonable as different polylines in a real-world scenario often represent different geographic objects (e.g., different river streams).

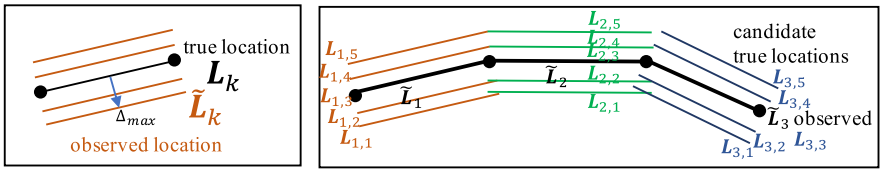
However, modeling $P(\tilde{\mathbf{L}}|\mathbf{L})$ is still non-trivial since there can be an infinite number of potential true polylines in the continuous space. Even if we fix the number of vertices in a true polyline location, the total number of potential true polylines is still exponential to the number of vertices (which can easily reach hundreds to thousands). It is computationally infeasible to model a joint conditional probability mass function for $P(\tilde{\mathbf{L}}|\mathbf{L})$ without some independence assumption on vertices of $\mathbf{L}$.

To address the above computational challenge, we make several assumptions. First, we consider each polyline label as a sequence of independent equal-length polyline segments, i.e., $\mathbf{L} =< \mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_k, \ldots, \mathbf{L}_{n_L} >$. Thus, we can now independently model $P(\tilde{\mathbf{L}}_k|\mathbf{L}_k)$ for each polyline segment. Second, we also assume that the observed imperfect segment is generated by shifting a true label segment along its perpendicular direction (i.e., parallel shifting). In other words, we assume an observed imperfect segment has the exact same vector shape with the hidden true label segment. The only difference is in their locations. We assume the segment shifting error to follow the perpendicular direction in order to reduce the interference between consecutive segments. Third, we further assume that the shifting error distance is a discrete random variable with a fixed number of C choices (i.e., Δ, 2Δ,..., $\Delta_{max} = C\Delta$, where Δ is the unit of location error distance, and Δ_{max} is the maximum range of location error distance). Therefore, for each observed imperfect segment $\tilde{\mathbf{L}}_k$, we have C candidate true segment locations, denoted as $\{\mathbf{L}_{k,c}|1 \leq c \leq C\}$. Under these assumptions, we can model $P(\tilde{\mathbf{L}}_k|\mathbf{L}_{k,c})$ as a probability mass function. For simplicity, we can assume a uniform distribution of location errors, i.e., $P(\tilde{\mathbf{L}}_k|\mathbf{L}_{k,c}) = \frac{1}{C}$ for any $1 \leq c \leq C$. In our formulation, Δ and C are two hyper-parameters that can be determined by the data characteristics in a specific application. For example, Δ can be the size of one or two pixels, and C can be calculated by a rough estimate of the maximum error distance Δ_{max}.

The idea is illustrated in Figure 3. Figure 3(a) shows the location error model from the hidden true location's perspective. Given a true location $\mathbf{L}_k$, the observed imperfect label location $\tilde{\mathbf{L}}_k$ is shifted from the true location by different distances along the perpendicular direction. In this example, the number of possible error distances is five (i.e., $C = 5$). Thus, $P(\tilde{\mathbf{L}}_k|\mathbf{L}_k) = \frac{1}{5}$ for every error distance. Figure 3(b) shows the location error model from the observed imperfect polyline's perspective. The observed imperfect polyline (in black) is partitioned into three equal-length segments and there are five candidate true segment locations for each observed segment (including the observed segment itself). Following the same location error model in Figure 3(a), we have $P(\tilde{\mathbf{L}}_k|\mathbf{L}_{k,c}) = \frac{1}{5}$ for every possible true segment location $\mathbf{L}_{k,c}$. In this article, our implementation assumes a uniform distribution of location error distances for simplicity. Alternatively, we can also assume a relatively lower probability for a larger error distance (in this case, we assume that the true segment location tends to be close to the observed segment).

3.2 Estimating Posterior Probability of True Segment Locations Based on Features

This step aims to estimate the posterior probability of a true segment location based on explanatory feature layers, i.e., $P(\mathbf{L}|\mathbf{X}, \Theta)$ in Equation (1). As discussed earlier, the probability $P(\mathbf{L}|\mathbf{X}, \Theta)$ can be estimated by the class predictions of the current neural network parameters Θ. There are several implementation issues that need to be resolved. First, the base neural network model is trained on square patches (e.g., 224 by 224 pixels) instead of the entire raster framework. A label segment can cross the border of several patches. Second, there are a large number of polyline segments. Evaluating the predicted class probability of each of them repeatedly is time-consuming.



(a) Location error of a segment (b) Location errors of consecutive segments

Fig. 3. Statistical model for polyline segment location error. The red, green, and blue segments are different sets of candidate true segment locations (best viewed in color).

ALGORITHM 1: The overall learning algorithm

Input:

- $\mathbf{X}$: Earth imagery pixel features
- $\tilde{\mathcal{L}} = \{\tilde{\mathbf{L}}^{(i)} | 1 \le i \le N\}$: A set of imperfect polyline labels
- d: a fixed buffer width for line segments

Output:

- Θ: Neural network parameters
- $\mathcal{L} = \{\mathbf{L}^{(i)} | 1 \le i \le N\}$: A set of refined polyline labels as a by-product

Pre-train a neural network $\mathbf{\Theta_0}$ based on imperfect labels $(\mathbf{X}, \tilde{\mathcal{L}})$

Partition each polyline $\tilde{\mathbf{L}}^{(i)}$ into fix-length segments $\tilde{\mathbf{L}}^{(i)} =< \tilde{\mathbf{L}}_1^{(i)}, \tilde{\mathbf{L}}_2^{(i)}, \ldots, \tilde{\mathbf{L}}_k^{(i)}, \ldots, \tilde{\mathbf{L}}_{n_L}^{(i)} >$

For each imperfect segment $\tilde{\mathbf{L}}_k^{(i)}$, generate candidate true segments $\{\mathbf{L}_{k,c}^{(i)} | c\}$ through perpendicular shift

while the model is not converged **do**

  Compute $P(\mathbf{L}_{k,c}^{(i)} | \mathbf{X}, \Theta_\mathbf{0})$ by model prediction for all $\{\mathbf{L}_{k,c}^{(i)} | i, k, c\}$ with buffer d

  Select the top candidate $c_0 \leftarrow \arg\max_c P(\tilde{\mathbf{L}}_k^{(i)} | \mathbf{L}_{k,c}^{(i)}) P(\mathbf{L}_{k,c}^{(i)} | \mathbf{X}, \Theta_\mathbf{0})$ for every input segment $\tilde{\mathbf{L}}_k^{(i)}$

  Handle a tie with (1) outermost first, (2) random selection, or (3) default to imperfect segment

  Update $\mathbf{\Theta_0}$ by model re-training based on selected label segments

return Θ_0, $\{\mathbf{L}_{k,c_0}^{(i)} | i, k\}$

Considering that these probabilities need to be evaluated for every iteration, the total time cost can be very large.

To address the computational challenges above, we design several implementation strategies. First, to address the issue that a segment can cross the patch border and reduce the number of redundant class predictions for separate segments, we choose to predict the class probabilities of all patches only once within each iteration and mosaic their probability maps together into the entire raster framework. In this way, the combined probability map can be reused to evaluate different segment probability $P(\mathbf{L}_k | \mathbf{X}, \Theta)$ without repeatedly running the neural network prediction. Specifically, to estimate the predicted probability for each segment $\mathbf{L}_k$, we created a bounding box of the buffered segment and extracted the pixels within the buffer of the polyline segment. Then, $P(\mathbf{L}_k | \mathbf{X}, \Theta)$ is calculated by the predicted probabilities for class 1 on all pixels within the polyline segment buffer. In order to avoid the bias toward large segment buffer size, we normalized the total probabilities within a buffer (by the number of pixels).

3.3 Overall Algorithm Flow

Now we introduce the overall algorithm flow. Algorithm 1 provides an overview of the algorithm structure. The algorithm first pre-trains a deep learning segmentation model (e.g., U-Net, DeepLab) based on imperfect input polyline labels (after buffering and rasterization). The

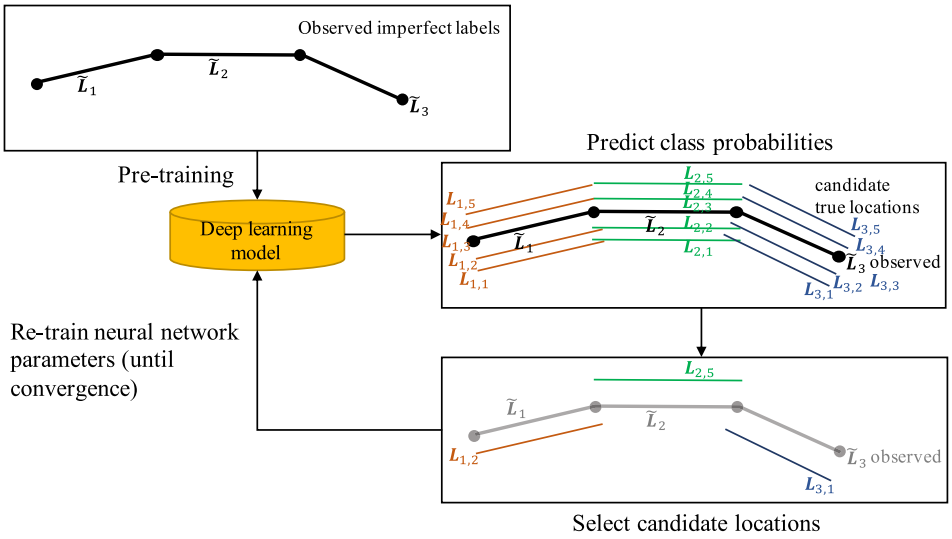
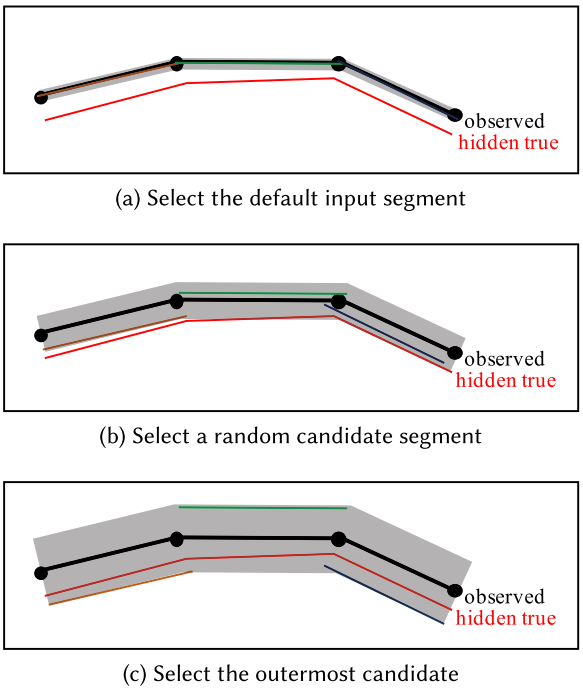


Fig. 4. An example of the algorithm flow. The red, green, and blue segments are different sets of candidate true segment locations (best viewed in color).

preprocessing step generates a set of candidate true segment locations for every input segment. The candidate generation follows our implementation strategy in Section 3.1 and Figure 3. Then, the algorithm goes through iterations. In each iteration, the model first predicts the class probabilities of all pixels in the training and validation windows in the raster framework. It calculates a weight for every candidate true segment location, which is measured by the location error model and deep learning model output class probabilities (steps 5 and 6). Note that if a deep learning model predicts near-zero class probability for all candidates, we have three strategies (details in the next paragraph). Based on the selected candidate true segment locations, we re-train the neural network model (step 8) and continue to the next iteration. The iteration stops when the validation accuracy no longer improves. Figure 4 illustrates the entire algorithm flow. From the discussion above, we can see that the time cost of the algorithm is linear to the number of polyline segments (due to the independence assumption between consecutive segments).

One important implementation issue is that the predicted class probabilities from the neural network may contain systematic errors; e.g., a polyline segment can be missed in the predicted class probability map (false negatives). In practice, we observe this case frequently at the first few algorithm iterations when the neural network model is inaccurate (due to the low quality of input labels). In this case, the predicted class probabilities $P(\mathbf{L}_{k,c}^{(i)}|\mathbf{X}, \Theta_\mathbf{0})$ are equally near zero for all candidates c. Thus, we need to design a strategy to select candidate true segment locations without overfitting to a particular poor polyline. We considered three different strategies: (1) select the default observed segment, (2) select the outermost segment with the largest location distance, and (3) randomly select a segment among all candidates.

The strategy made in this step is very important. As discussed earlier, when inferring true label locations, we need to make a good balance between *exploration* (keeping a wide range of candidate true labels to avoid overfitting to a particular wrong label, especially when the neural network has not been well trained yet at the first few iterations) and *exploitation* (focusing on a narrow range of candidate true labels for refinement). From this perspective, selecting the outermost segment label



(a) Select the default input segment

(b) Select a random candidate segment

(c) Select the outermost candidate

Fig. 5. The effect of candidate segment selection in empty probability area. The shaded area is the predicted stream class probability from a model trained on three selected candidates, which are shown in red, green, and blue, respectively. The input label segment and the hidden true label segment are shown in black and red, respectively (best viewed in color).

will provide the most benefit, as it will help to train a model that will predict the stream class into a wide range of areas beyond the input imperfect line buffer. The random selection strategy is in the middle and will provide a modest range of predicted stream class areas. The strategy of selecting the default input segment is the worst, as it easily leads to overfitting to the input imperfect line and get stuck in it. This effect is illustrated in Figure 5. A model trained on the outermost segments have the best chance of covering the true label location in red color (Figure 8(c)), a model trained on the random candidate has a modest range of coverage (Figure 8(b)), but a model trained on the default input segments will have a serious overfitting issue (Figure 8(a)). Although a model trained from the outermost candidates has the widest range of stream class prediction (containing false positives), we anticipate that the predicted range will be narrower (more focused) in the later iterations. This is observed in our experiments on real-world data. Note that these choices are only made when the predicted pixel class probabilities are equal for all candidates (they are all near zero). If the class probabilities from the neural network are non-zero on candidate segments, we still select the top candidate with the highest probability.

Another implementation detail is the measure of training and validation performance during the training iterations. There are no manually refined ("perfect") segment labels being provided from the input. Thus, the training and validation performance is all based on the currently refined label segments through candidate selection. We assume that the iterations will gradually improve the label quality. In the evaluation on real-world datasets, we find that this assumption works well for model validation. However, we acknowledge that the performance of the proposed learning framework can collapse if the original input labels are completely wrong.

4 EVALUATION

The goal is to compare the proposed model with the baseline methods in classification performance. We will also analyze the training curves and parameter sensitivity of the proposed model. All experiments were conducted on a deep learning workstation with 4 NVIDIA RTX 6000 GPUs connected by NV-Link (each with 24GB GPU memory) and 128GB RAM. For the base deep learning model, we used U-Net [50] and DeepLab [8].

Dataset Description: We evaluated our proposed method in a real-world application of NHD refinement based on high-resolution remote sensing data. We used two datasets collected from the Rowan Creek in North Carolina and Panther Creek in Iowa. The input features include earth imagery from the **National Agriculture Imagery Program (NAIP)** with red, green, blue, and near-infrared channels; digital elevation model; Lidar point cloud intensity; and slope derived from elevation. The input imperfect streamline location shapefile was collected from an earlier coarse version of NHD and visually interpreted coarse background imagery (these polylines are not well aligned with the stream pixels). The ground-truth streamline polyline labels for testing are manually refined by hydrologists (this true location was hidden from our model in training and validation and was only used for testing). All imagery was resampled into a 1-meter resolution. We used a 2-meter buffer and a 16-meter buffer to rasterize the polylines in two study areas respectively due to different river widths.

In each dataset, we split the study area into two disjoint parts: the upper part is for testing, and the lower part is for training and validation. For the first dataset, we randomly selected 698 windows for training and 40 windows for validation. The training windows and validation windows are not overlapping with each other to keep independence. We augmented training and validation windows by flipping horizontally and vertically as well as 90-degree rotation. Thus, the total number of training and validation windows was 2,792 and 160, respectively. We randomly selected 200 windows in the upper part for testing. The window size is 224 by 224 pixels. For the second dataset, the total number of training and validation windows was 1,008 and 60 after augmentation. The number of test windows was 300. Note that for class labels in the training and validation windows, we used imperfect lines that are refined by the current iteration. We only used manually refined lines in testing windows.

Model Architecture: We did experiments with two kinds of segmentation models as our base model: U-Net and DeepLabv3+. We used the U-Net model with a 224 by 224 input shape implemented in Keras.[1] The U-Net model consists of an encoder-decoder structure. The encoder has six double-convolution layers and five max-pooling layers. The numbers of output channels for those convolution layers are 32, 64, 128, 256, 512, and 1,024, respectively. There is batch normalization within each convolutional layer before ReLU (rectified linear unit) non-linear activation. The decoder of the model upsamples the encoded feature map to a higher resolution based on transpose convolution. The decoder concatenates upsampled global features with the corresponding local features from the encoder.

We used the DeepLabv3+ model [9] with an input shape of 224 by 224. It was implemented in Keras.[2] The DeepLabv3+ model has an encoder-decoder structure. The encoder applies **Atrous Spatial Pyramid Pooling (ASPP)** with four different rates to detect multiple-scale features. The encoder uses an output stride of 16. Then the encoder features are first bilinearly upsampled by a factor of 4 and then concatenated with the corresponding low-level features from the network backbone that have the same spatial resolution. Then another bilinear upsampling by a factor of

[1] https://github.com/ZFTurbo/ZF_UNET_224_Pretrained_Model.
[2] https://github.com/rishizek/tensorflow-deeplab-v3-plus.

4 is applied to obtain the original resolution. Note that the network backbone ResNet-101 in the original paper cannot be utilized for our tasks, because there is no available pre-trained model for the earth image with seven input spectral bands. The network backbone we used is four double-convolution layers and four max-pooling layers. We initialized the backbone model randomly and trained it with the DeepLab model together.

Model Hyper-parameters: For the U-Net model, we used double-convolution layers and batch normalization. The dropout rate is 0.2. The mini-batch size is 32. We used the Adam optimizer and negative of dice co-efficient as the loss function. The dice co-efficient loss function is the same as F1-score except that it allows for soft predicted class probabilities. We used a decaying learning rate that reduced the learning rate by half if the validation loss did not improve over five epochs (with an initial learning rate of 10^{-1} and a minimum learning rate of 10^{-5}). We also used early stopping to stop model fitting if the validation loss did not improve over 20 epochs. We used a maximum of 50 epochs in model pre-training and each iteration.

For the DeepLabv3+ model, the backbone model has four double-convolution layers and max-pooling layers. The dropout rate in the backbone model is 0.2, and we did not use dropout in the ASPP and upsampling part. The mini-batch size is 32. We used the Adam optimizer. The loss function is the same as the U-Net model, i.e., negative of dice coefficient. We used a decaying learning rate that reduced the learning rate by half if the validation loss did not improve over five epochs (with an initial learning rate of 0.05 and a minimum learning rate of 10^{-5}). We also used early stopping to stop model fitting if the validation loss did not improve over 20 epochs. We used a maximum of 50 epochs in model pre-training and each iteration.

For candidate true shape location generations in our method, we split the input imperfect polylines into small chunks (each with a length of 10 meters). For simplicity, we generated candidate true locations by shifting the segment in perpendicular directions (15 candidates above and 15 candidates below, 31 in total including the input shape segment itself). The perpendicular distance between two candidate segments was 1.5 meters. Within each iteration, we re-trained our model from scratch with the newly inferred label locations.

Evaluation Metrics: We used precision, recall, and F1-score on the streamline class to evaluate candidate methods.

4.1 Comparison on Classification Performance

We first compared the overall classification performance between the baseline U-Net model and our proposed model. The setup was the same as described at the beginning of this section. The results on two datasets are summarized in Tables 1 and 2. The first column in the confusion matrices was the number of pixels predicted into the non-stream class. The second column in the confusion matrices was the number of pixels predicted into the stream class. We can see that on the first dataset, the pre-trained U-Net model from the imperfect ground-truth label has very poor precision and recall in the streamline class (the overall F1-score was 0.46). In contrast, our proposed U-Net with iterations improved the precision from 0.39 to 0.66 and improved the recall from 0.57 to 0.68. The overall F1-score in our method is 0.67, significantly higher than the baseline U-Net model. A close look at the confusion matrix shows that our method reduces the number of false positives from 147,480 to 48,658 (by 67%) and reduces the number of false negatives from 79,854 to 44,303 (by 55%). The metrics confirm that our proposed method significantly enhanced the baseline image segmentation model when the ground-truth segment labels are imperfect. We observe an improvement in our method against the baseline DeepLab model (from 0.42 to 0.64 in F1-score). The same levels of improvements are also observed in the second dataset in Table 2.

Table 1. Comparison of Classification Performance on the First Dataset

Method	Class	Confusion Matrix		Precision	Recall	F Score
U-Net	Non-stream	9,750,497	147,480	0.99	0.99	0.99
	Stream	79,854	57,369	**0.39**	**0.57**	**0.46**
Our Method on U-Net	Non-stream	9,849,319	48,658	1.00	1.00	1.00
	Stream	44,303	92,920	**0.66**	**0.68**	**0.67**
DeepLabv3+	Non-stream	9,764,563	133,414	0.99	0.99	0.99
	Stream	64,381	72,842	**0.35**	**0.53**	**0.42**
Our Method on DeepLabv3+	Non-stream	9,844,775	53,202	1.00	0.99	0.99
	Stream	47,433	89,790	**0.63**	**0.65**	**0.64**

Table 2. Comparison of Classification Performance on the Second Dataset

Method	Class	Confusion Matrix		Precision	Recall	F-score
U-Net	Non-stream	13,617,128	101,758	0.95	0.99	0.97
	Stream	657,808	676,106	**0.87**	**0.51**	**0.64**
Our Method on U-Net	Non-stream	13,392,213	326,673	0.99	0.98	0.99
	Stream	69,322	1,264,592	**0.79**	**0.95**	**0.86**
DeepLabv3+	Non-stream	13,623,101	95,785	0.93	0.99	0.96
	Stream	997,758	336,156	**0.78**	**0.25**	**0.38**
Our Method on DeepLabv3+	Non-stream	13,274,401	444,485	0.99	0.97	0.98
	Stream	146,311	1,187,603	**0.73**	**0.89**	**0.80**

Table 3. Comparison of Classification Performance on Default Candidate Selection Methods

Method	Class	Confusion Matrix		Precision	Recall	F-score
Outermost first	Non-stream	9,849,319	48,658	1.00	1.00	1.00
	Stream	44,303	92,920	**0.66**	**0.68**	**0.67**
Randomly select	Non-stream	986,0471	37,506	0.99	1.00	0.99
	Stream	58,946	78,277	**0.68**	**0.57**	**0.62**
Default to input	Non-stream	9,865,697	32,280	0.99	1.00	0.99
	Stream	79,881	57,342	**0.64**	**0.42**	**0.51**

4.2 The Effect of Default Candidate Segment Selection

We compared the effect of default candidate true segment selection when there are low probabilities on all candidates. As discussed in Section 3.2, this situation happens frequently in the first couple of iterations when the labels have not been well refined yet. It is very important to select the candidates that keep a wide range of exploration without focusing on a narrow range (to avoid overfitting). The experiment results confirm our analysis. We compared three different strategies of candidate selection when none of the candidate true segment locations are confident based on current neural network model class probability predictions. From the results in Table 3, we can see that the strategy of selecting the input imperfect segment produces the worse results. The reason is that default to the input segment leads to the model overfitting to the narrow range of the input polyline segment. In contrast, selecting random segment location produces significantly better results due to a wider range of spatial footprints to avoid overfitting. Selecting the outmost segment provides the best results as it explores the largest range of potential locations.

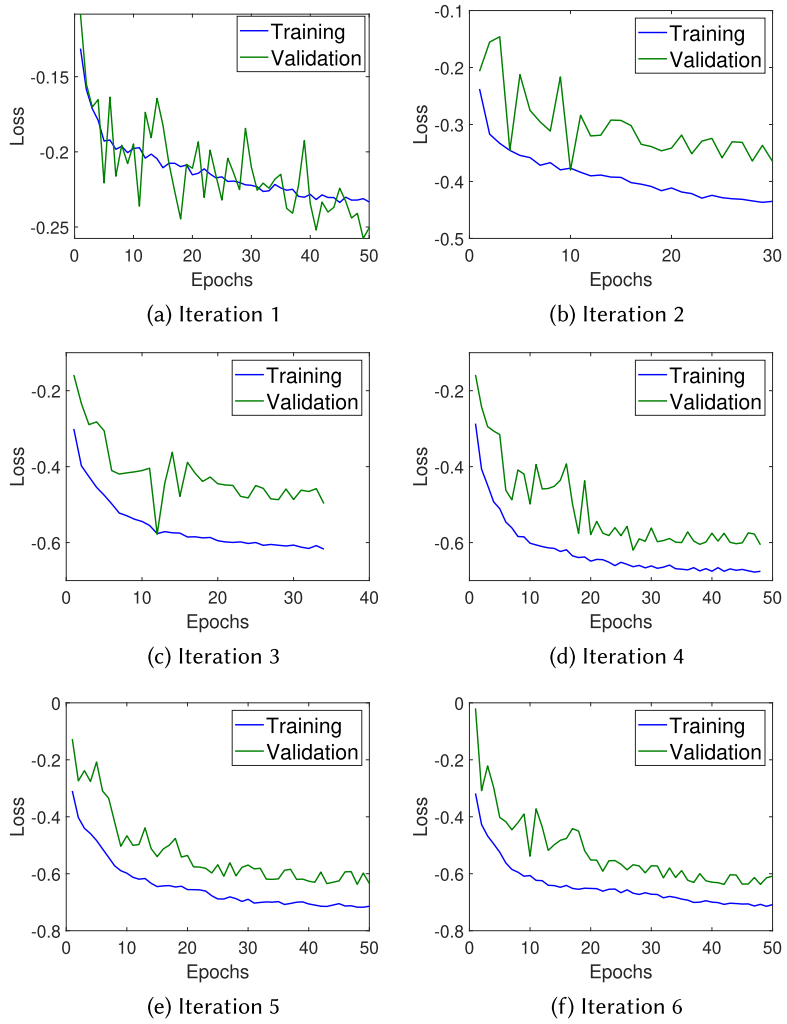


Fig. 6. Training curves of different Iterations.

4.3 The Learning Curve of Different Iterations

In order to understand the influence on the training process of each iteration, we plotted the training and validation F1-score (based on the current inferred "true" label location) after each iteration in Figure 7. The F1-score after each iteration was from the re-trained U-Net model (through up to 50 epochs) based on the currently inferred label location. We can see that the F1-score continued improving during iterations. The training and validation F1-scores in the first iteration (slightly above 0.2) are worse than those of the pre-trained U-Net model. The F1-scores significantly improve in the second and third iterations and converge at the sixth iteration (with a validation F1-score of 0.60).

In order to examine the detailed model learning process, we also plotted the training curves (training and validation loss over every epoch) within each iteration. Those training curves are shown in Figure 6. As the iteration continues, the gap between training loss and validation loss decreased. We also observed that the converged training and validation loss was lower through the iterations, largely due to the continual improvement over label locations.

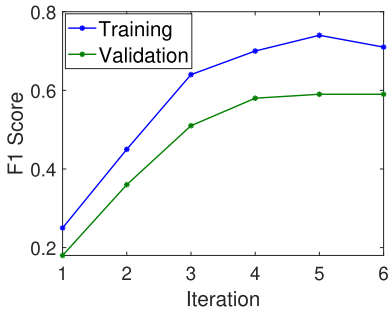
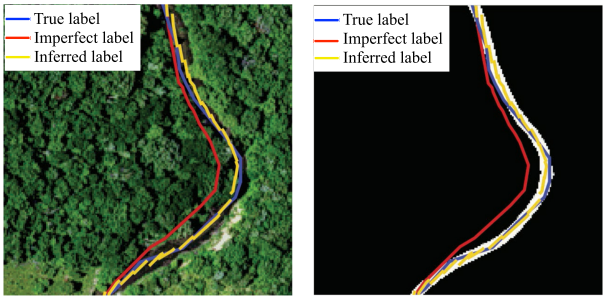


Fig. 7. The converged F1-scores on training and validation datasets after each iteration.



(a) Earth imagery background (b) River channel background

Fig. 8. Visualization of the inferred true label location, manually refined (true) label location, and initial imperfect line location (best viewed in color).

4.4 Inference of True Label Locations

We also visualized the inferred (selected) true label locations during the iteration. Figure 8 showed the comparison of our inferred true label locations (in brown) with the manually refined true label locations (in blue) as well as the initial imperfect label locations (in red). The actual footprint of the river channel was also shown in the background imagery, including a true color earth imagery in Figure 8(a) and a binary map in Figure 8. From the comparison, we can see that our inferred true label locations (those selected candidate segments in brown) are far closer to the manually refined true label locations (in blue) than the initial imperfect label (in red). These visualized results verified that our iteration framework could infer the true label locations while training the U-Net model. What was even more interesting was that our algorithm seemed to make the best efforts in inferring true label locations. For example, when the initial imperfect label line segments (in red) are not well oriented with the true line location (in blue), our selected candidates (in brown) still crossed over with the blue line as much as possible. And when the initial imperfect label line had the same orientation as the true line, our inferred line location was almost perfectly aligned with the true line.

4.5 Interpretation of Final Prediction Map

We also visualized the predicted streamline class maps in the test region from our model and the U-Net model. Due to limited space, we selected one representative sub-area in the test region to have a zoomed-in view. The results are shown in Figure 9. Figure 9(a) shows the manually refined streamline labels, which are the "perfect" ground truth in testing. Figure 9(b) shows the

(a) Manually refined streamline (b) U-Net prediction (c) Our model prediction

Fig. 9. Visualization of the final predicted class maps in the test region.

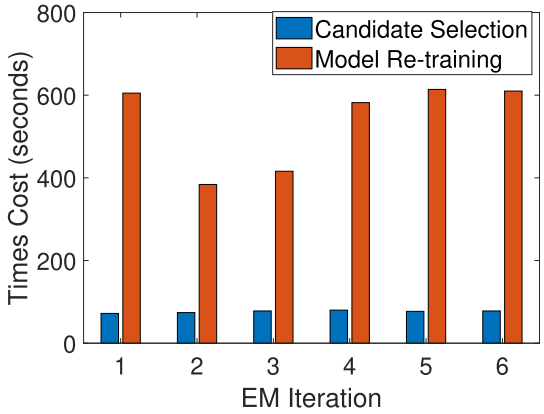


Fig. 10. Time cost of candidate selection and model re-training during different iterations.

predicted streamline locations by the baseline U-Net model. We can see that it contained many false positives (false streamlines predicted) and false negatives (missing true streamlines). For example, the upper right branch of the stream was barely identified by U-Net. The upper left branch was also not continuous in the U-Net predictions. In addition, there are false stream segments in U-Net predictions on the top. In contrast, our results (shown in Figure 9(c)) are far better with fewer false positives and false negatives. We also did a careful examination of the entire predicted maps over the entire region and found similar trends as shown in this figure.

4.6 Analysis of Computational Time Costs

We evaluated the computational efficiency of our proposed EM framework. The experiments were conducted on our deep learning workstation with 4 NVIDIA RTX 6000 GPUs connected by NV-Link (each GPU has 24GB memory). Model training was conducted on all four GPUs through the distributed training tool in Tensorflow. The time costs between iterations are summarized in Figure 10. The blue bars and red bars show the time costs of candidate selection (re-generating rasterized true label map) in the CPU and model re-training in the GPUs, respectively. From the results, we can see that the candidate selection part took far less time than the model training and its time cost was relatively stable across iterations. The time cost of model training varied across iterations due to early stopping. The longest training time was around 10 minutes in one iteration. The numbers are highly dependent on the hardware platform.

5 CONCLUSION AND FUTURE WORKS

We investigated deep learning models for earth imagery segmentation from imperfect ground-truth labels with geometric annotation errors. The problem is important for broad applications in earth sciences, such as refining the National Hydrologic Dataset through high-resolution remote sensing imagery. However, the problem is non-trivial due to the requirement to infer the true geometric label locations and update neural network parameters simultaneously. We propose a generic deep learning framework to simultaneously infer hidden true label locations and train deep learning model parameters. Evaluations on real-world datasets confirm that the proposed framework significantly outperformed the baseline method.

For future work, we plan to continue to improve our proposed weakly supervised learning framework. For example, one limitation of the current framework is that the refined polylines are discontinuous between segments (although their location footprints align better with the true polylines). We plan to improve our polyline refinement component to generate continuous polylines. We also plan to generalize our framework from polyline labels to polygon labels. Another potential future direction is using deep reinforcement learning to automatically annotate labels.

REFERENCES

[1] Joaquín Abellán and Andrés R. Masegosa. 2012. Bagging schemes on the presence of class noise in classification. *Expert Systems with Applications* 39, 8 (2012), 6827–6837.

[2] David Acuna, Amlan Kar, and Sanja Fidler. 2019. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11075–11083.

[3] 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.

[4] John E. Ball, Derek T. Anderson, and Chee Seng Chan Sr. 2017. Comprehensive survey of deep learning in remote sensing: Theories, tools, and challenges for the community. *Journal of Applied Remote Sensing* 11, 4 (2017), 42609.

[5] Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. JMLR Workshop and Conference Proceedings, 17–36.

[6] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. DeepSVG: A hierarchical generative network for vector graphics animation. arXiv preprint arXiv:2007.11301 (2020).

[7] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. 2019. AutoCorrect: Deep inductive alignment of noisy geometric annotations. arXiv preprint arXiv:1908.05263 (2019).

[8] 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.

[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'18)*. 801–818.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*.

[11] Lívia Castro Degrossi, João Porto de Albuquerque, Roberto dos Santos Rocha, and Alexander Zipf. 2018. A taxonomy of quality assessment methods for volunteered and crowdsourced geographic information. *Transactions in GIS* 22, 2 (2018), 542–560.

[12] Thomas G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 40, 2 (2000), 139–157.

[13] Helen Dorn, Tobias Törnros, and Alexander Zipf. 2015. Quality evaluation of VGI using authoritative data—A comparison with land use data in Southern Germany. *ISPRS International Journal of Geo-Information* 4, 3 (2015), 1657–1671.

[14] 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.

[15] 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.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. [n.d.]. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[17] Benoit Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 25, 5 (2014), 845–869.

[18] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. 2018. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing* 70 (2018), 41–65.

[19] Michael F. Goodchild and Linna Li. 2012. Assuring the quality of volunteered geographic information. *Spatial Statistics* 1 (2012), 110–120.

[20] 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* (2020).

[21] 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'21)*. SIAM, 253–261.

[22] Benjamin Herfort, Hao Li, Sascha Fendrich, Sven Lautenbach, and Alexander Zipf. 2019. Mapping human settlements with higher accuracy and less volunteer efforts by combining crowdsourcing and deep learning. *Remote Sensing* 11, 15 (2019), 1799.

[23] Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He, and Paul Kennedy. 2019. Deep learning techniques for medical image segmentation: Achievements and challenges. *Journal of Digital Imaging* 32, 4 (2019), 582–596.

[24] Yi Hong and Weiping Zhu. 2015. Spatial co-training for semi-supervised image classification. *Pattern Recognition Letters* 63 (2015), 59–65.

[25] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439 (2020).

[26] Zhe Jiang. 2018. A survey on spatial prediction methods. *IEEE Transactions on Knowledge and Data Engineering* 31, 9 (2018), 1645–1664.

[27] Zhe Jiang. 2020. Spatial structured prediction models: Applications, challenges, and techniques. *IEEE Access* 8 (2020), 38714–38727.

[28] 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.

[29] 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.

[30] Zhe Jiang and Shashi Shekhar. 2017. *Spatial Big Data Science*. Schweiz: Springer International Publishing AG.

[31] 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.

[32] Hiroshi Kajino, Yuta Tsuboi, Issei Sato, and Hisashi Kashima. 2012. Learning from crowds and experts. In *Proceedings of the Human Computation Workshop*. 107–113.

[33] 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.

[34] A Kendall, V. Badrinarayanan, and R. Cipolla. 2017. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *British Machine Vision Conference 2017 (BMVC'17)*.

[35] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in Bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 5580–5590.

[36] Simon A. A. Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R. Ledsam, Klaus H. Maier-Hein, S. M. Ali Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. 2018. A probabilistic U-net for segmentation of ambiguous images. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 6965–6975.

[37] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. 2019. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7930–7939.

[38] Zhiwu Lu, Zhenyong Fu, Tao Xiang, Peng Han, Liwei Wang, and Xin Gao. 2017. Learning from weak and noisy labels for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 3 (2017), 486–500. https://doi.org/10.1109/TPAMI.2016.2552172

[39] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson. 2019. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS Journal of Photogrammetry and Remote Sensing* 152, November 2018 (2019), 166–177. https://doi.org/10.1016/j.isprsjprs.2019.04.015

[40] David R. Maidment. 2017. Conceptual framework for the national flood interoperability experiment. *JAWRA Journal of the American Water Resources Association* 53, 2 (2017), 245–257.

[41] 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).

[42] Volodymyr Mnih and Geoffrey E. Hinton. 2012. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML'12)*. 567–574.

[43] Miguel Monteiro, Loïc Le Folgoc, Daniel Coelho de Castro, Nick Pawlowski, Bernardo Marques, Konstantinos Kamnitsas, Mark van der Wilk, and Ben Glocker. 2020. Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 (NeurIPS'20), virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/95f8d9901ca8878e291552f001f67692-Abstract.html.

[44] Richard B. Moore, Lucinda D. McKay, Alan H. Rea, Timothy R. Bondelid, Curtis V. Price, Thomas G. Dewald, and Craig M. Johnston. 2019. *User's Guide for the National Hydrography Dataset Plus (NHDPlus) High Resolution*. Technical Report. US Geological Survey.

[45] Kun Ouyang, Reza Shokri, David S. Rosenblum, and Wenzhuo Yang. 2018. A non-parametric generative model for human trajectories. In *IJCAI*. 3812–3817.

[46] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1944–1952.

[47] Sandra K. Poppenga, Dean B. Gesch, and Bruce B. Worstell. 2013. Hydrography change detection: The usefulness of surface channels derived from LiDAR DEMs for updating mapped hydrography 1. *JAWRA Journal of the American Water Resources Association* 49, 2 (2013), 371–389.

[48] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. arXiv preprint arXiv:1412.6596 (2014).

[49] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. arXiv preprint arXiv:1705.10694 (2017).

[50] 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.

[51] 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* (2020).

[52] 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.

[53] 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

[54] L. D. Schultz, M. P. Heck, David Hockman-Wert, T. Allai, S. Wenger, N. A. Cook, and Jason B. Dunham. 2017. Spatial and temporal variability in the effects of wildfire and drought on thermal habitat for a desert trout. *Journal of Arid Environments* 145 (2017), 60–68.

[55] Hansi Senaratne, Amin Mobasheri, Ahmed Loai Ali, Cristina Capineri, and Mordechai Haklay. 2017. A review of volunteered geographic information quality assessment methods. *International Journal of Geographical Information Science* 31, 1 (2017), 139–167.

[56] 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.

[57] J. D. Simley and W. J. Carswell Jr. 2009. The national map–hydrography: US geological survey fact sheet 2009–3054. US Geological Survey National Center, Reston, VA.

[58] Lawrence V. Stanislawski, Ethan J. Shavers, Shaowen Wang, Zhe Jiang, E. Lynn Usery, Evan Moak, Alexander Duffy, and Joel Schott. 2021. Extensibility of U-Net neural network model for hydrographic feature extraction and implications for hydrologic modeling. *Remote Sensing* 13, 12 (2021), 2368.

[59] Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N. Chiang, Zhihao Wu, and Xiaowei Ding. 2020. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Analysis* (2020), 101693.

[60] Grigorios Tsagkatakis, Anastasia Aidini, Konstantina Fotiadou, Michalis Giannopoulos, Anastasia Pentari, and Panagiotis Tsakalides. 2019. Survey of deep-learning approaches for remote sensing observation enhancement. *Sensors* 19, 18 (2019), 3929.

[61] Joaquin Vanschoren. 2018. Meta-learning: A survey. arXiv preprint arXiv:1810.03548 (2018).

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

[63] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. 2018. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7278–7286.

[64] Zhihao Wei, Kebin Jia, Pengyu Liu, Xiaowei Jia, Yiqun Xie, and Zhe Jiang. 2021. Large-scale river mapping using contrastive learning and multi-source satellite imagery. *Remote Sensing* 13, 15 (2021), 2893.

[65] Wendy Wright, Beth Nielsen, Jeffrey D. Mullen, and John Dowd. 2012. *Agricultural Groundwater Policy during Drought: A Spatially Differentiated Approach for the Flint River Basin*. Technical Report.

[66] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2691–2699.

[67] 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 & Data Mining*. 2545–2554.

[68] Zewei Xu, Shaowen Wang, Lawrence V. Stanislawski, Zhe Jiang, Nattapon Jaroenchai, Arpan Man Sainju, Ethan Shavers, E. Lynn Usery, Li Chen, Zhiyu Li, et al. 2021. An attention U-Net model for detection of fine-scale hydrologic streamlines. *Environmental Modelling & Software* (2021), 104992.

[69] Lin Yang, Yizhe Zhang, Jianxu Chen, Siyuan Zhang, and Danny Z. Chen. 2017. Suggestive annotation: A deep active learning framework for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 399–407.

[70] Zhiding Yu, Weiyang Liu, Yang Zou, Chen Feng, Srikumar Ramalingam, B. V. K. Vijaya Kumar, and Jan Kautz. 2018. Simultaneous edge alignment and learning. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 388–404.

[71] Xiangrong Zhang, Qiang Song, Ruochen Liu, Wenna Wang, and Licheng Jiao. 2014. Modified co-training with spectral and spatial views for semisupervised hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7, 6 (2014), 2044–2055.

[72] Tongxue Zhou, Su Ruan, and Stéphane Canu. 2019. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array* 3 (2019), 100004.

[73] Xiaojin Zhu. 2005. Semi-supervised learning literature survey. (2005).

[74] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. 2017. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine* 5, 4 (2017), 8–36.

Received December 2020; revised March 2021; accepted August 2021

AUTHOR QUERIES

Q1: AU: Please provide complete mailing and email addresses for all authors in Authors' addresses line.
Q2: AU: OK that in-text mentions for Figures 6 through 8 are out of order?
Q3: AU: Please provide volume and page numbers.
Q4: AU: Please check difference in years here.
Q5: AU: In Proceedings/Conference references, please spell out title first and then use abbreviation and abbreviated year in parentheses throughout references where not already done.
Q6: AU: Please provide volume and page numbers.
Q7: AU: Please provide volume number.
Q8: AU: Please provide volume and page numbers.
Q9: AU: Please provide volume number.
Q10: AU: Is this reference complete?