Quantifying and Reducing Registration Uncertainty of Spatial Vector Labels on Earth Imagery

Wenchong He
Zhe Jiang
whe2,zhe.jiang@ufl.edu
Department of CISE
The University of Florida
Gainesville, Florida, USA

Marcus Kirby
mskirby1@crimson.ua.edu
Department of Computer Science
The University of Alabama
Tuscaloosa, Alabama, USA

Yiqun Xie
xie@umd.edu
Department of GIS
The University of Maryland
College Park, Maryland, USA

Xiaowei Jia
xiaowei@pitt.edu
Department of Computer Science
The University of Pittsburgh
Pittsburgh, Pennsylvania, USA

Da Yan
yanda@uab.edu
Department of Computer Science
The University of Alabama at Birmingham
Birmingham, Alabama, USA

Yang Zhou
yangzhou@auburn.edu
Department of CSSE
Auburn University
Auburn, Alabama, USA

ABSTRACT

Given raster imagery features and imperfect vector training labels with registration uncertainty, this paper studies a deep learning framework that can quantify and reduce the registration uncertainty of training labels as well as train neural network parameters simultaneously. The problem is important in broad applications such as streamline classification on Earth imagery or tissue segmentation on medical imagery, whereby annotating precise vector labels is expensive and time-consuming. However, the problem is challenging due to the gap between the vector representation of class labels and the raster representation of image features and the need for training neural networks with uncertain label locations. Existing research on uncertain training labels often focuses on uncertainty in label class semantics or characterizes label registration uncertainty at the pixel level (not contiguous vectors). To fill the gap, this paper proposes a novel learning framework that explicitly quantifies vector labels' registration uncertainty. We propose a registration-uncertainty-aware loss function and design an iterative uncertainty reduction algorithm by re-estimating the posterior of true vector label locations distribution based on a Gaussian process. Evaluations on real-world datasets in National Hydrography Dataset refinement show that the proposed approach significantly outperforms several baselines in the registration uncertainty estimations performance and classification performance.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539410

CCS CONCEPTS

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

KEYWORDS

weakly supervised spatial deep learning, remote sensing, imperfect vector labels, registration uncertainty

ACM Reference Format:
Wenchong He, Zhe Jiang, Marcus Kirby, Yiqun Xie, Xiaowei Jia, Da Yan, and Yang Zhou. 2022. Quantifying and Reducing Registration Uncertainty of Spatial Vector Labels on Earth Imagery. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539410

1 INTRODUCTION

Over the last decade, deep learning technologies have achieved tremendous success in computer vision and natural language processing applications. Unfortunately, the broader success of deep learning in geoscience (e.g., Earth image classification) has been hindered by the lack of high-quality training labels. In these applications, collecting high-quality vector labels is slow, tedious, and expensive due to sending a field crew on the ground or visually interpreting high-resolution imagery pixels. On the other hand, it is often much easier to annotate coarse vector labels with registration errors (i.e., vector labels that may not align well with image pixels).

Given raster imagery features and imperfect vector training labels with registration uncertainty, this paper studies a weakly supervised spatial deep learning framework that can quantify and reduce vector label registration uncertainty as well as train neural network parameters simultaneously. For example in the application of National Hydrography Dataset refinement, the input raster features include spectral bands of Earth observation imagery, the digital elevation model, lidar point intensity, and topographic indices. The input vector labels of river streams are polylines misaligned with the actual stream pixels on the Earth imagery (i.e., with registration

errors). Given such inputs, the goal is to design a learning framework that can quantify and reduce the registration uncertain of the imperfect label (i.e., refine the polyline labels) and train a deep neural network model that can accurately classify feature imagery pixels into binary classes (stream and non-stream).

However, the problem poses several unique challenges. First, there exists registration uncertainty in vector label registration errors due to unknown true vector label locations. Second, modeling such uncertainty is non-trivial given that the vector labels are often continuous polylines on a 2D plane while input features are discrete pixels in a 2D grid. Such a gap poses additional challenges when designing a pixel-wise loss function of a neural network based on uncertain vector-level training labels and in exploring uncertainty reduction (vector refinement) based on pixel-level class predictions. Third, the problem requires learning neural network parameters and inferring true vector label locations simultaneously.

Existing research on weakly-supervised learning with label uncertainty often focuses on addressing uncertainty in label class semantics, assuming label locations to be correct or irrelevant (e.g., samples are independent and identically distributed) [2, 20, 27]. Techniques include simple data cleaning to filter noise [6], choosing relatively noise-tolerant models [5], designing robust loss function [18, 19] and learning noise distribution [14, 24]. Thus, these techniques cannot address the registration uncertainty in the ground truth labels. Some works focus on label location errors and uncertainty [1, 3, 8, 15, 17, 18, 25, 28] but only infer true label locations at the raster pixel level (e.g., small square patches, object boundary, edge pixels, or medical pixel alignment). These methods do not fully address the label registration uncertainty in the vector representation and cannot guarantee line contiguity during label refinement. One recent work [9] focuses on registration errors of vector labels, but the method is ad-hoc and cannot quantify the registration uncertainty, causing over-confident label refinement. Note that the word "uncertainty" in our work should not be confused with other relevant works [10–12] focusing on "ambiguity" or "uncertainty", which refers to nonunanimous training labels due to different opinions of several experts, even though all of the different opinions are considered correct. In contrast, we assume there exists a single unknown true vector label location that perfectly aligns with corresponding imagery features. In summary, none of existing works can quantify and reduce the uncertainty of vector labels.

To fill the gap, we propose a novel learning framework that can quantify and reduce registration uncertainty of the vector labels. Specifically, we make the following contributions:

- First, we propose a registration-uncertainty-aware loss function to train neural network, which calculates the loss between the probability of each pixel contained in the buffered area of the uncertain vector label and the neural network prediction probabilities.
- Second, we propose to reduce the vector label uncertainty by re-estimating the posterior of true vector locations as a Gaussian process based on the prior vector label distribution (from the previous iteration) and the likelihood (from the predicted pixel class probabilities).
- Third, evaluations on real-world high-resolution remote sensing datasets in National Hydrography Dataset (NHD) refinement show that the proposed framework outperforms baseline methods in both uncertainty estimation performance and classification accuracy. Case studies also confirm the quality of refined vector labels and uncertainty estimations. (e.g., improve Accuracy-versus-Uncertainty from 0.25 to 0.54) (e.g., reducing the false positives and false negatives by 67% and 55%, respectively)

2 PROBLEM STATEMENT

2.1 Preliminaries

Spatial raster framework: A spatial raster framework is a tessellation of a 2D plane into a regular grid. Each grid cell is a pixel. We denote the features of all raster pixels as $\mathbf{X} \in \mathbb{R}^{n \times n \times k}$, and the corresponding pixel class labels as $\mathbf{Y} \in \{0, 1\}^{n \times n}$, where n is the raster length, and k is the input feature channel number. An Example of feature layers are spectral bands of Earth imagery. The class layer can be whether pixels are stream or non-stream. Figure 1(a) provides an example of a spatial raster with 20 by 20 pixels.

Spatial vector: A spatial vector is a geometric representation of a spatial object such as a point, polyline, and polygon on a 2D plane. It is an alternative way of representing spatial data. This paper focuses on **polyline** vectors (i.e., one or more consecutive line segments joined end to end), such as river streams or road segments. Mathematically, a polyline can be expressed by a curve function $\mathbf{L}(s) = (u(s), v(s)), s \in [s_a, s_b]$, where u and v are 2D coordinates and s is a variable to reflect the one degree of freedom. Note that for simplicity, we may omit the variable s and use $\mathbf{L}$ to denote a polyline in this paper. In reality, a stream has a non-zero width, which can be captured by a **buffer** on top of a polyline, denoted as $\mathbf{B}(\mathbf{L})$. Figure 1(b) provides an example.

A spatial vector can be **co-registered** into a raster-based on their common spatial reference system. After registration, a (buffered) vector shape can be converted into a class layer $\mathbf{Y}$ in the raster format (also called **rasterization**), i.e., pixels overlapping with the vector are assigned with corresponding thematic class labels. After rasterization of vector training labels, we can train neural network parameters based on the feature layers $\mathbf{X}$ and the class layer $\mathbf{Y}$.

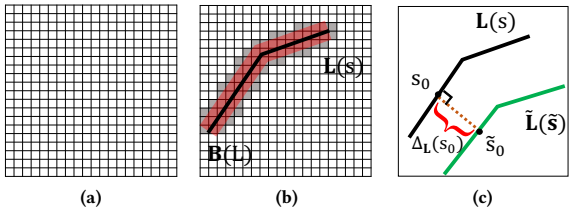


Figure 1: Examples of a spatial raster (a), a vector label and rasterization (b), and registration errors (c)

Registration errors: Registration errors refer to the misalignment between different spatial layers when they are co-registered together. This paper focuses on registration errors between a vector class label layer (e.g., polylines of river streams) and raster image feature layers (e.g., Earth imagery). Such registration errors exist due to manual annotation mistakes at a coarse resolution [7, 23].

Mathematically, we express registration errors between uncertain vector labels $\tilde{\mathbf{L}}(\tilde{s})$ and raster image features by assuming an unknown true vector label locations $\mathbf{L}(s)$ (perfectly align with image

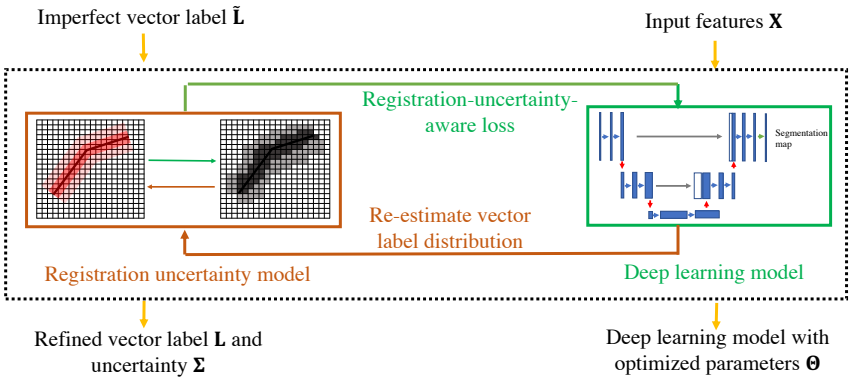


Figure 2: Illustration of the Registration-uncertainty-aware Deep Learning (RuaDL) framework

features). Specifically, the registration error at a particular point $\tilde{\mathbf{L}}(\tilde{s}_0)$ on the polyline $\tilde{\mathbf{L}}(\tilde{s})$ is the vector difference between that point to a corresponding point $\mathbf{L}(s_0)$ along **perpendicular direction** on the true polyline $\mathbf{L}(s)$ as illustrated in Figure 1(c). That is, $\Delta_{\mathbf{L}}(s_0) = \mathbf{L}(s_0) - \tilde{\mathbf{L}}(\tilde{s}_0)$. Since the direction of the registration error is fixed (perpendicular) at every point on the polyline, we only need to express it as a scalar $\Delta_{\mathbf{L}}(s_0)$, whose magnitude reflects the distance and whose sign reflects the direction. Then, we can express the registration errors along a polyline as a scalar function $\Delta_{\mathbf{L}}(s), s \in [s_a, s_b]$.

Registration uncertainty (uncertainty of registration errors): In the above expression of registration errors, the imperfect vector $\tilde{\mathbf{L}}(\tilde{s})$ is often given, but the true vector label location $\mathbf{L}(s)$ is unknown (uncertain). Thus, the registration errors $\Delta_{\mathbf{L}}(s)$ has uncertainty. One of the major goals of this work is to model this uncertainty and even reduce it through weakly supervised learning based on image features. Note that reducing the uncertainty of registration errors also means refining the vector training labels, as we can easily calculate the true vector $\mathbf{L}(s)$ from imperfect vector $\tilde{\mathbf{L}}(\tilde{s})$ if $\Delta_{\mathbf{L}}(s)$ is known (certain).

Uncertainty of pixel labels: As for the spatial raster framework, the pixel labels $\tilde{\mathbf{Y}}$ are obtained by rasterization around the spatial vector label locations $\tilde{\mathbf{L}}$. Since the observed spatial vector labels are uncertain, the raster class layer $\tilde{\mathbf{Y}}$ are also uncertain. Note that our notion of rasterized pixel class uncertainty $\tilde{\mathbf{Y}}$ is different from existing works which assume independence between pixel labels [20, 27]. In our problem, the pixel label uncertainty are generated from spatial vector registration errors, and there exists auto-correlation between nearby locations.

2.2 Problem definition

Based on the above definitions, we formally define our problem.

Input:

- A spatial raster framework with explanatory feature layers $\mathbf{X}$
- Uncertain training labels as a set of polylines $\tilde{\mathcal{L}} = \{\tilde{\mathbf{L}}_1, \tilde{\mathbf{L}}_2, ..., \tilde{\mathbf{L}}_n\}$
- A base deep neural network type (e.g., U-Net, DeepLab)

Output:

- Refined vector labels $\mathcal{L} = \{\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_n\}$ and the uncertainty
- Deep neural network parameters $\mathbf{\Theta}$ that can predict the class layer $\mathbf{Y}$ from features $\mathbf{X}$.

Objective:

- Maximize the quality of the refined vector labels
- Maximize the uncertainty estimation performance

Constraint:

- Registration errors of a polyline is assumed to be point shift along perpendicular directions and within a maximum distance range Δ_{max}, which is a hyper-parameter
- The class is binary (one type of vector label)

3 THE PROPOSED APPROACH

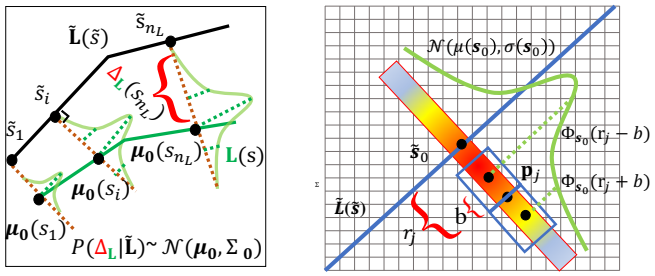
This section introduces the proposed approach that performs model training and vector label refinement in a unified framework. The problem is technically challenging in several aspects. First, we need to model the geometric registration errors in vector shape to capture the uncertainty and auto-correlation of the polyline registration error between nearby spatial locations. Second, to learn the neural network parameters we need to incorporate the uncertainty of the vector labels into the uncertainty of pixel-wise labels. Thirdly, we need to reduce the uncertainty of registration errors based on the pixel-wise prediction of the neural network. To address these challenges, we propose a generic spatial deep learning framework that iteratively updates deep learning model parameters while inferring hidden true vector label distributions. The framework is illustrated in Figure 2. Given an imperfect polyline label $\tilde{\mathbf{L}}$, we model the underlying ground truth $\mathbf{L}$ based on a Gaussian process that explicitly captures the uncertainty of vector label registration errors along a continuous polyline. Based on the uncertainty model, we propose a registration-uncertainty-aware loss function that can learn neural network parameters. The main intuition is to translate the uncertain vector line into soft weights of image pixels in the loss function. Then we propose to reduce the uncertainty of registration errors by re-estimating the posterior distribution of true vector label locations based on predicted pixel class probabilities from neural network model. We now introduce each component.

3.1 Probabilistic model of the registration uncertainty of polylines

Design a probabilistic model for registration uncertainty of polyline is non-trivial for three reasons. First, the registration error at any point on the polyline is a 2D vector with both a distance and a direction. Second, the distance of registration error is a continuous variable. Third, a continuous polyline concerns an infinite number of points, whose registration errors are mutually dependent due to spatial auto-correlation.

To address the above challenges, we propose to use a Gaussian process to model the uncertainty of polyline registration errors. Recall that in the previous section, we express the registration errors along a polyline as a scalar function $\Delta_{\mathbf{L}}(s), s \in [s_a, s_b]$. Thus, we assume $\Delta_{\mathbf{L}}(s)$ to follow a Gaussian process [16], as expressed in Equation 1 with the prior mean $\mu(s) = 0$ and the covariance function based on a kernel $k(s, s') = \sigma_0^2 \exp(-|s - s'|^2/2l^2)$, where σ_0^2 is the variance and l is the length scale (kernel bandwidth). These hyper-parameters can be determined based on prior knowledge about the quality of the observed polyline label or through cross-validation. Intuitively, we should select a relatively large variance at the beginning to reflect the high uncertainty of true polyline locations. Note that our work is different from existing Gaussian process models for registration uncertainty in medical images [15], which focuses on registration errors between pixels from two corresponding images.

$$\Delta_{\mathbf{L}}(s) \sim \mathrm{GP}(\mu(s), k(s, s')), s, s' \in [s_a, s_b] \tag{1}$$



(a) Gaussian process model

(b) The probability of a pixel falling into the buffer of true polyline given registration error distribution $P(\Delta_{\mathbf{L}}|\tilde{\mathbf{L}}) \sim \mathcal{N}(\boldsymbol{\mu_0}, \Sigma_0)$

Figure 3: Gaussian process model for registration uncertainty

In order to allow re-estimation (refine) the Gaussian process model (reduce its uncertainty through learning iterations), we sample a finite number of equal-interval points along the imperfect polyline $\tilde{\mathbf{L}}(\tilde{s})$, denoted as $\tilde{\mathbf{L}} = \{\tilde{\mathbf{L}}(\tilde{s}_1), \tilde{\mathbf{L}}(\tilde{s}_2), ..., \tilde{\mathbf{L}}(\tilde{s}_{n_\mathbf{L}})\}$, where $n_\mathbf{L}$ is the number of sample points. We denote the corresponding sample points along the true polyline $\mathbf{L}(s)$ as $\mathbf{L} = \{\mathbf{L}(s_1), \mathbf{L}(s_2), ..., \mathbf{L}(s_{n_\mathbf{L}})\}$. Thus, given the imperfect polyline location $\tilde{\mathbf{L}}$, the true vector label locations $\mathbf{L}$ can be uniquely determined by $\Delta_{\mathbf{L}}$, which follows a multi-variate Gaussian distribution, as expressed in Equation 2. The mean vector $\boldsymbol{\mu_0}$ and covariance matrix Σ_0 are determined by the mean function $\mu(s)$ and kernel function $k(s, s')$ of the Gaussian process in Equation 1. As shown in Figure 3(a), the red bracket shows the registration errors $\Delta_{\mathbf{L}} = \{\Delta_{\mathbf{L}}(s_1), \Delta_{\mathbf{L}}(s_i), ..., \Delta_{\mathbf{L}}(s_{n_\mathbf{L}})\}$ at each point. The green line illustrates the registration errors distribution, which follow a multi-variate Gaussian distribution, with mean vector of $\boldsymbol{\mu_0} = \{\boldsymbol{\mu_0}(s_1), \boldsymbol{\mu_0}(s_i), ..., \boldsymbol{\mu_0}(s_{n_\mathbf{L}})\}$ and covariance Σ_0. The non-zero covariance between sampled points can capture the spatial auto-correlation of registration errors.

$$P(\Delta_{\mathbf{L}}|\tilde{\mathbf{L}}) \sim \mathcal{N}(\boldsymbol{\mu_0}, \Sigma_0) \tag{2}$$

3.2 Training neural network parameters with uncertain polyline label

This part aims to learn neural network parameters with uncertain vector labels. Instead of training a neural network with the observed imperfect vector label $\tilde{\mathbf{L}}$ directly, we employ a probabilistic distribution of the unknown true vector label $P(\mathbf{L}|\tilde{\mathbf{L}})$, which can be determined from $P(\Delta_{\mathbf{L}}|\tilde{\mathbf{L}})$. Then we designed an uncertainty-aware loss function. This is challenging because we need to incorporate the pixel-level class probability predictions from the network and vector-level uncertainty of training labels.

Let us first review the common binary cross-entropy loss function based on a fixed (certain) polyline training label $\mathbf{L}$ in Equation 3, where $\mathbf{p}_j$ represents a pixel, $\mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})}$ is an indicator function to check if pixel $\mathbf{p}_j$ is within the polyline buffer $B(\mathbf{L})$, $\hat{\mathbf{Y}}$ and $\hat{y}_j$ are the predicted class probabilities on all pixels and on pixel $\mathbf{p}_j$, respectively. The indicator function $\mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})}$ is a hard label (0 or 1) for pixels acquired from the certain polyline label and $p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta})$ is the neural network prediction probability.

$$\begin{aligned} L(\hat{\mathbf{Y}}, \mathbf{L}) = \sum_j \{ & \mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})} \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta}) \\ & + (1 - \mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})})(1 - \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta})) \} \end{aligned} \tag{3}$$

When the vector training labels are uncertain, i.e., the vector label $\mathbf{L}$ follows a probabilistic distribution $P(\mathbf{L}|\tilde{\mathbf{L}})$, we generalize the above loss function into its expectation on $\mathbf{L}$. This is expressed in Equation 4, where the distribution of unknown true vector label location $\mathbf{L}$ can be calculated from a Gaussian prior (Equation 1) or from a posterior estimation given image features (Equation 8). For now, we assume that $P(\mathbf{L}|\tilde{\mathbf{L}})$ is already calculated.

$$\begin{aligned} L(\hat{\mathbf{Y}}, \mathbf{L}) = & \mathbb{E}_{\mathbf{L} \sim P(\mathbf{L}|\tilde{\mathbf{L}})} \sum_j \{ \mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})} \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta}) \\ & + (1 - \mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})})(1 - \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta})) \} \end{aligned} \tag{4}$$

We can simplify the above uncertainty-aware loss function by taking the expectation operation onto the indicator function inside the sum. As shown in Equation 5, where the expectation of the indicator function becomes the probability of a pixel falling into the buffer of the true polyline $B(\mathbf{L})$. The probability $P(\mathbf{p}_j \in B(\mathbf{L}))$ can be considered as a soft weight on each pixel in the loss function. The soft weight can capture the uncertainty of pixel labels (in contrast to the hard weight in the indicator function based on a certain polyline label in Equation 3).

$$\begin{aligned} L(\hat{\mathbf{Y}}, \mathbf{L}) &= \sum_j \mathbb{E}_{\mathbf{L} \sim P(\mathbf{L}|\tilde{\mathbf{L}})} \mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})} \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta}) \\ &\quad + \mathbb{E}_{\mathbf{L} \sim P(\mathbf{L}|\tilde{\mathbf{L}})} (1 - \mathbb{I}_{\mathbf{p}_j \in B(\mathbf{L})})(1 - \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta})) \\ &= \sum_j P(\mathbf{p}_j \in B(\mathbf{L})) \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta}) \\ &\quad + (1 - P(\mathbf{p}_j \in B(\mathbf{L})))(1 - \log p(\hat{y}_j|\mathbf{X}, \boldsymbol{\Theta})) \end{aligned} \tag{5}$$

However, it is still non-trivial to compute the probability of a pixel falling into the buffer of the true polyline label, i.e., $P(\mathbf{p}_j \in B(\mathbf{L}))$,

given the probabilistic distribution of the polyline. We present a theorem to compute this probability.

Theorem 3.1. *Given an imperfect polyline $\tilde{\mathbf{L}}(\tilde{s})$, whose registration errors follow a Gaussian process $\Delta_{\mathbf{L}}(s) \sim GP(\mu(s), k(s,s'))$, and an image pixel $\mathbf{p}_j$. Assume that the projection of $\mathbf{p}_j$ to the polyline is point s_0, whose corresponding registration error along the perpendicular direction is $\Delta_{\mathbf{L}}(s_0) \sim N(\mu(s_0), \sigma^2(s_0))$. $\mathbf{B}(\mathbf{L})$ is the polygon that buffer on top of the polyline $\mathbf{L}$. Then $P(\mathbf{p}_j \in \mathbf{B}(\mathbf{L}))$ can be computed by Equation 6, where Φ is the cumulative distribution function of a standard normal distribution, r_j is the scalar distance to pixel $\mathbf{p}_j$ to the imperfect polyline, and b is the buffer width.*

$$P(\mathbf{p}_j \in B(\mathbf{L})) = \Phi(\frac{r_j - \mu(s_0) + b}{\sigma(s_0)}) - \Phi(\frac{r_j - \mu(s_0) - b}{\sigma(s_0)}) \tag{6}$$

Proof. As illustrated in Figure 3(b), the main intuition is that the probability of a pixel falling into the true polyline buffer $P(\mathbf{p}_j \in \mathbf{B}(\mathbf{L}))$ equals the probability of the true polyline location $\tilde{s}_0$ being located within the buffer distance of the pixel $r_j - b$ and $r_j + b$, which is equal to $\Phi_{s_0}(r_j + b) - \Phi_{s_0}(r_j - b)$. The detailed proof is provided in the supplementary materials. □

3.3 Reducing uncertainty of polyline labels through posterior estimation

The previous two subsections introduce a probabilistic model of the uncertainty of polyline label registration errors and an uncertainty-aware loss function to train deep neural network parameters. This subsection proposes a method to re-estimate the uncertainty distribution (i.e., to reduce the uncertainty) of polyline labels based on the neural network pixel class predictions.

The main idea is expressed in Figure 4. The posterior is re-estimated from the prior distribution and likelihood function based on neural network predictions. First, the prior distribution of polyline registration errors is modeled with Gaussian process in subsection 3.1 $P(\Delta_{\mathbf{L}}|\tilde{\mathbf{L}}) \sim \mathcal{N}(\boldsymbol{\mu_0}, \Sigma_0)$ as shown in the Figure 4(a). Second, we denote streamline class (positive) from the neural netowork prediction as $\tilde{\mathbf{Y}}$ (black and red pixels in Figure 4(b)). Then we model the the likelihood function $\mathcal{F}(\Delta_{\mathbf{L}}|\tilde{\mathbf{Y}}, \mathbf{X}, \boldsymbol{\Theta})$ as a multi-variate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu_1}, \Sigma_1)$, whose mean and variance are determined by the prediction positive class along the perpendicular line of the imperfect polyline location (as illustrated by the red pixels in Figure 4(b)). Specifically, we compute the location mean and variance $(\boldsymbol{\mu_1}(s_i), \boldsymbol{\sigma_1}(s_i))$ independently at different sample points. Then $\boldsymbol{\mu_1} = \{\boldsymbol{\mu_1}(s_1), \boldsymbol{\mu_1}(s_2), ..., \boldsymbol{\mu_1}(s_{n_{\mathbf{L}}})\}$ and covariance $\Sigma_1 = diag\{\boldsymbol{\sigma_1}(s_1), \boldsymbol{\sigma_1}(s_2), ..., \boldsymbol{\sigma_1}(s_{n_{\mathbf{L}}})\}$. Then the posterior of registration error follows a Gaussian distribution as Theorem 3.2 shown. The posterior estimation of Gaussian process can take advantage of the covariance between samples to interpolate its mean estimate and variance. As an example in Figure 4(c), there is no positive class prediction along the perpendicular direction of point $\tilde{s}_1$, and the variance $\boldsymbol{\sigma_1}(s_1)$ is relative larger than other points. For next iteration the registration errors prior distribution can be updated by the estimated posterior.

Theorem 3.2. *Given the registration error prior distribution $P(\Delta_{\mathbf{L}}) \sim \mathcal{N}(\boldsymbol{\mu_0}, \Sigma_0)$ and likelihood estimation $\mathcal{F}(\Delta_{\mathbf{L}}|\tilde{\mathbf{Y}}, \mathbf{X}, \boldsymbol{\Theta}) \sim \mathcal{N}(\boldsymbol{\mu_1}, \Sigma_1)$ from neural netowrk prediction $\tilde{\mathbf{Y}}$. The posterior true polyline location distribution follows a Gaussian distribution*

$$P(\Delta_{\mathbf{L}}|\tilde{\mathbf{Y}}, \mathbf{X}, \boldsymbol{\Theta}) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{7}$$

where

$$\Sigma = (\Sigma_0^{-1} + \Sigma_1^{-1})^{-1}, \boldsymbol{\mu} = \Sigma(\Sigma_0^{-1}\boldsymbol{\mu}_0 + \Sigma_1^{-1}\boldsymbol{\mu}_1) \tag{8}$$

Proof. The intuition is to compute the posterior by Bayes' theorem with the prior and likelihood function. We omit the details due to limited space. □

3.4 Overall algorithm

Algorithm 1 provides an overview of the proposed weakly supervised learning framework. The framework first initializes a Gaussian process prior of the uncertain polyline registration errors. Based on the uncertainty-aware loss function, it trains a neural network and predicts pixel classes. Then the framework conducts iterations. Each iteration first estimates the independent Gaussian distribution of true polyline locations at sample points based on predicted pixel classes. Based on the prior (or posterior) distribution of true polyline locations from the previous iteration, the algorithm updates the new posterior distribution. After that, it re-trains the neural network based on the new posterior distribution and makes class predictions. The iterations continue until the model converges (e.g., no improvement in converged validation loss).

Algorithm 1 The overview of RuaDL framework

Input:
- $\mathbf{X}$: Explanatory feature layers
- $\tilde{\mathcal{L}} = \{\tilde{\mathbf{L}}\}$: Imperfect polyline labels

Output:
- $\boldsymbol{\Theta}$: Neural network parameters
- $\mathcal{L} = \{\mathbf{L} : \boldsymbol{\mu}, \Sigma\}$: Refined polyline labels and uncertainty

1: Create equal interval sample points along each polyline
2: Initialize a Gaussian process prior of registration errors $P(\Delta_{\mathbf{L}}|\tilde{\mathbf{L}}) \sim \mathcal{N}(\boldsymbol{\mu_0}, \Sigma_0)$ // Equation 2
3: Train neural network $\boldsymbol{\Theta}$ based on $\mathbf{X}$ and $\Delta_{\mathbf{L}}$ distribution, then predict pixel class probabilities $P(\hat{\mathbf{Y}}|\mathbf{X}, \Theta)$ // Equation 4
4: **while** model not converged **do**
5: Estimate $(\boldsymbol{\mu_1}, \Sigma_1)$ of likelihood function $\mathcal{F}(\Delta_{\mathbf{L}}|\hat{\mathbf{Y}}, \mathbf{X}, \Theta)$ based on class predictions $\hat{\mathbf{Y}}$ // Figure 4 (b)
6: Update $(\boldsymbol{\mu}, \Sigma)$ of posterior $P(\Delta_{\mathbf{L}}|\hat{\mathbf{Y}}, \mathbf{X}, \boldsymbol{\Theta})$ // Equation 8
7: Re-train neural network $\boldsymbol{\Theta}$ based on posterior $P(\Delta_{\mathbf{L}}|\hat{\mathbf{Y}}, \mathbf{X}, \boldsymbol{\Theta})$ and predict pixel class probabilities $P(\hat{\mathbf{Y}}|\mathbf{X}, \Theta)$// Equation 4
8: **return** $\boldsymbol{\Theta}$, $\{\mathbf{L}\} = \{\boldsymbol{\mu}, \Sigma\}$ //The estimated mean and uncertainty of true polyline distribution is returned

Time complexity of Algorithm 1: The cost includes neural network training, estimate $(\boldsymbol{\mu_1}, \Sigma_1)$ based on the model predictions, and update $(\boldsymbol{\mu}, \Sigma)$ based on equation 8. Estimating $(\boldsymbol{\mu_1}, \Sigma_1)$ costs $O(N)$, where N is the total number of sampled points along the imperfect polyline. The time complexity is linear because we assume independence between different points in this step. Updating $(\boldsymbol{\mu}, \Sigma)$ based on equation 8 costs $O(N^3)$, because we need to take inversion of the covariance matrix. To reduce the time complexity, we divide the polyline into multiple segments (blocks) and each segment contains k points. Then the time complexity is reduced to

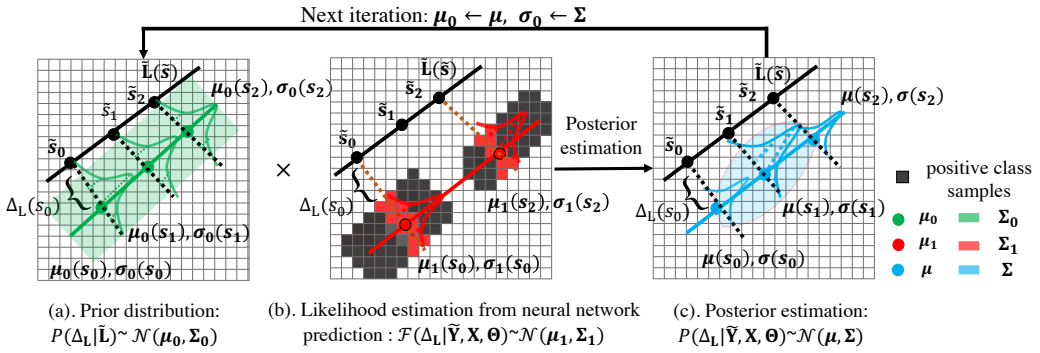


Figure 4: Illustration of reducing uncertainty for polyline labels through posterior estimation

$O(Nk^2)$. In practice, we can choose a reasonable value of k based on domain knowledge (e.g., to what extent the registration errors between nearby points along a polyline resemble each other).

4 EVALUATION

The goal is to compare the proposed model with the baselines in uncertainty quantification and model classification performance. We will also analyze the training iterations 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 128 GB RAM. For the deep learning base methods, We used **U-Net** [22] and **DeepLab** [4]. We also implemented three weakly supervised learning baselines. For baselines, we obtain the model uncertainty based on the output probability after softmax layer (confidence).

- **Base model (BaseDL):** We used the U-Net model with a 224 by 224 input shape implemented in Keras [22](source codes [21, 26]) and the DeepLabv3+ model [4].
- **Base model with self-training (SelfDL):** We selected those patches with high confidence predictions and add those into the training set for the next iteration.
- **Base model with patch-based translation (PatchDL):** We implemented patch-based translation[18] by computing the class posterior probability to train the base model iteratively.
- **Weakly-supervised spatial deep learning (WssDL):** We implemented framework for vector labels with registration errors[9] that iteratively updates neural networks parameters and infers true vector label locations with dynamic programming.
- **Registration-uncertainty-aware deep learning framework (RuaDL):** We used the base deep learning models implemented in Python and Keras with our framework.

Dataset description: We evaluated our proposed method in two real-world high-resolution remote sensing datasets of the National Hydrography Dataset (NHD) refinement application. The two datasets were collected from two watersheds at the Rowan Creek and Panther Creek, USA. 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 non-expert volunteers by visually interpreting the imagery. The truth streamline locations for testing were manually refined by hydrologists for the evaluation purpose and were hidden from model training and validation. All imagery was resampled into a 1-meter resolution. We used a 4-meter buffer to rasterize polylines in the first dataset and a 16-meter buffer in the second dataset (due to different river stream widths).

The number of sample windows (224 by 224 pixels) in the training, validation, and test sets were 2792, 200, and 160 in the first dataset and 1008, 60, and 300 in the second dataset.

Model hyper-parameters: For the base model, the dropout rate is 0.2. We used the negative of dice co-efficient as the loss function, a decaying rate that reduced the learning rate by half if the validation loss did not improve over five epochs (with an initial learning rate of 0.01 and a minimum of 10^{-5}). We also used early stopping with a patience of 20 epochs and a maximum of 50 epochs.

Classification evaluation metrics: We used precision, recall, and F1 score on the streamline class to evaluate classification performance and use IoU (intersection over union with manually refined lines) to measure the quality of refined training lines.

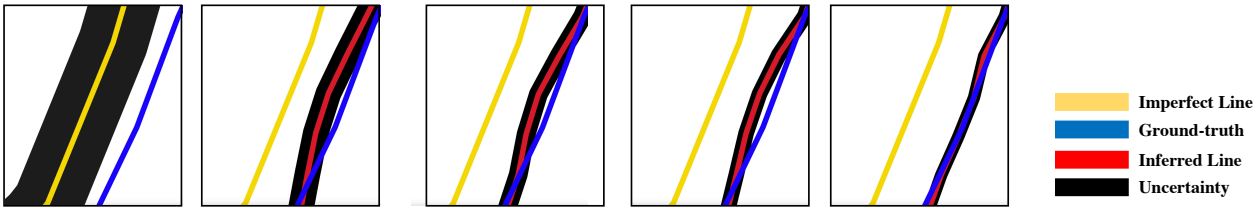
Uncertainty quantification evaluation metrics: The quantitative evaluation metrics for uncertainty estimations performance is Accuracy versus Uncertainty (AvU)[13]. We set an uncertainty threshold T_u to group uncertainty estimation into certain and uncertain. Then the sample predictions are grouped into four catetgories as Table 1 shows. $n_{\mathrm{AC}}, n_{\mathrm{AU}}, n_{\mathrm{IC}}, n_{\mathrm{IU}}$ represent the number of samples in the categories AC, AU, IC, IU, respectively. As Equation 9 shows, AvU measures the percentage of two categories AC and IU . A reliable model should provide higher AvU measure ($AvU \in [0, 1]$).

Table 1: Accuracy versus Uncertainty (AvU)

		Uncertainty	
		Certain	Uncertain
Accuracy	Accurate	Accurate Certain (AC)	Accurate Uncertain (AU)
	Inaccurate	Inaccurate Certain (IC)	Inaccurate Uncertain (IU)

$$AvU = \frac{n_{\mathrm{AC}} + n_{\mathrm{IU}}}{n_{\mathrm{AC}} + n_{\mathrm{AU}} + n_{\mathrm{IC}} + n_{\mathrm{IU}}} \tag{9}$$

However, AvU is usually biased by the accuracy of the model. Since model tend to have high confidence for accurate prediction.



(a) Initialization (IoU: 0.18) (b) Iteration 1 (IoU: 0.26) (c) Iteration 2 (IoU: 0.30) (d) Iteration 3 (IoU: 0.32) (e) Iteration 4 (IoU: 0.36)

Figure 5: Refined polyline labels and its uncertainty: the uncertainty of the inferred line is reduced during iterations.

We propose an evaluation metric that evaluate uncertainty performance for accurate and inaccurate prediction separately. Specifically, we compute AvU_A for accurate predictions and AvU_I for inaccurate predictio with the following equations:

$$AvU_A = \frac{n_{\text{AC}}}{n_{\text{AC}} + n_{\text{AU}}}, AvU_I = \frac{n_{\text{IU}}}{n_{\text{IC}} + n_{\text{IU}}} \tag{10}$$

In our evaluation, we compute the harmonic average of AvU_A and AvU_I: $AvU = \frac{2*AvU_A*AvU_I}{AvU_A+AvU_I}$ to penalize the extreme cases instead of the arithmetic average .

4.1 Comparison on Classification Performance

We first compared the overall classification performance between the base model, weakly supervised learning based on self-training, patch-based translation and spatial deep learning. The setup was the same as described at the beginning of this section. The results on two datasets were summarized in the left column of Table 2 and Table 3 (Appendix). 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 the pre-trained U-Net model from the imperfect ground truth label had very poor precision and recall in the streamline class (the overall F1 score was 0.42). Adding self-training strategy slightly improved the test F1 from 0.42 to 0.47 (largely due to a better recall). Patch-based translation improved the recall from 0.44 to 0.69 but the precision dropped to 0.31. The reason for the worse precision is that patch-based translation cannot reduce the uncertainty of the polyline label. The weakly-supervised spatial deep learning model perform well, because the method explicitly captures the spatial registration error. However, the limitation of spatial deep learning is that it cannot estimate the refined label uncertainty, which will be discussed in subsection 4.2. Our proposed RuaDL framework based on iterative training improved the precision from 0.39 to 0.65 and improved the recall from 0.44 to 0.70. The overall F1-score in our method is 0.67. The confusion matrix shows that our method reduces the number of false positives from 79801 to 51599 (by 61%) and reduces the number of false negatives from 79672 to 40303 (by 56%). The metrics confirmed that our method significantly enhanced the baseline model when the training labels were imperfect. We see similar results on the second dataset (see Table 3). The results with DeepLab are provided in supplementary materials

The polyline quality and uncertainty value during different iterations: To understand the polyline refinement process and uncertainty reduction over iterations, we selected a smaller patch of the training area and plotted the inferred polyline (red) and its uncertainty (black) after each iteration in Figure 5. In each iteration, we train a new U-Net from scratch based on the current refined polyline labels and their uncertainty. We initialized the Gaussian Process kernel with a large variance value. We can see that in the following iteration, the polyline label IoU is improved and its uncertainty value is reduced. After convergence, the polyline IoU was improved from 0.18 to 0.36 in the whole training area.

4.2 Comparison on Uncertainty Quantification Performance

We aim to evaluate the performance of uncertainty quantification for our proposed RuaDL framework. We expect the model to be certain about the refined label prediction when it is accurate and provides higher uncertainty estimates when making inaccurate label refinement. We choose two cases to visualize the uncertainty estimation performance as Figure 6 shows. We can see Figure 6 (b) show higher uncertainty on the inferred polyline label and the label quality is worse than 6 (a). The uncertainty estimations can reflect their label refinement quality.

We also evaluate the uncertainty estimation performance quantitatively of our model versus baseline methods: deep learning base model, weakly-supervised learning based on self-training, patch-based translation, and spatial deep learning. The quantitative evaluation metric is AvU in Equation 10 and the results were summerized in the right column of Table 2 and Table 3. The numbers in the table correspond the number of samples in four categories: $n_{\text{AC}}, n_{\text{AU}}, n_{\text{IC}}, n_{\text{IU}}$. We can see the base U-Net has good uncertainty estimation for accurate predictions (97% are certain) but for the inaccurate predictions, the uncertainty estimation tend to be overconfident, 14% are uncertain. U-Net with self-training and patch-based model show worse AvU because the iterative training strategy make the model more certain about its prediction during iteration, especially for the inaccurate predictions. Weakly-supervised spatial deep learning model improve the overall classification accuracy dramatically. However, for the inaccurate prediction, the model only give high uncertainty to 14% samples. The overall AvU is 0.25. In contrast, our model improve the overall AvU to 0.54, significantly higher than the baselines, because it improves the uncertainty estimations for inaccurate prediction to 38%. We can observe similar results in the second dataset. The baselines give worse uncertainty estimations for inaccurate predictions (around 5% $\sim$ 8% are uncertain). In contrast, our model has 58% uncertain samples for the inaccurate prediction, dramatically improves the overall uncertainty

Table 2: Comparison on classification and uncertainty estimation performance on dataset 1

	Classification Performance						Uncertainty Estimation Performance				
Method	Class	Confusion Matrix		Precision	Recall	F1 score	Accuracy	Uncertainty		AvU_A/AvU_I	AvU
		Negative	Positive					Certain	Uncertain		
BaseDL	Non-stream	9818176	79801	0.99	0.99	0.99	Accurate	248182	18295	0.93	
	Stream	79672	57551	**0.39**	**0.44**	**0.42**	Inaccurate	460083	90005	0.14	**0.24**
		Negative	Positive					Certain	Uncertain		
SelfDL	Non-stream	9747792	150185	1.00	0.98	0.99	Accurate	275847	16439	0.94	
	Stream	48750	88473	**0.37**	**0.64**	**0.47**	Inaccurate	505295	53411	0.10	**0.18**
		Negative	Positive					Certain	Uncertain		
PatchDL	Non-stream	9686619	211358	1.00	0.98	0.99	Accurate	293279	20592	0.93	
	Stream	41302	95921	**0.31**	**0.69**	**0.43**	Inaccurate	577582	73527	0.11	**0.19**
		Negative	Positive					Certain	Uncertain		
WssDL	Non-stream	9867813	30164	0.99	0.99	0.99	Accurate	785916	44648	0.95	
	Stream	57614	79609	**0.71**	**0.58**	**0.64**	Inaccurate	420379	61823	0.14	**0.25**
		Negative	Positive					Certain	Uncertain		
RuaDL	Non-stream	9846378	51599	1.00	0.99	0.99	Accurate	742436	53236	0.93	
	Stream	40552	96671	**0.65**	**0.70**	**0.67**	Inaccurate	397575	237981	0.38	**0.54**

estimations performance. The results with DeepLab base model are provided in supplementary materials.

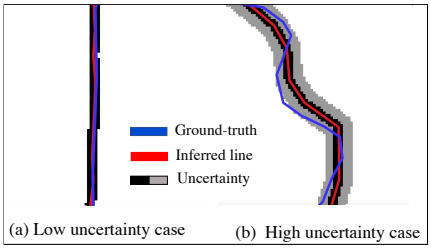


Figure 6: Uncertainty visualization

4.3 Sensitivity analysis

We also conducted a sensitivity analysis of our model to different hyper-parameters, including both the Gaussian Process kernel function parameters and sample point distances. Gaussian process kernel includes initial standard deviation σ_0, kernel length scale l, and the number of sample points k in one segment (the size of the covariance matrix). The results are summarized in Figure 7. First, we change the kernel initial standard deviation σ_0 from 2 to 8 and used the default setting in all other hyper-parameters. The corresponding test F1 scores are in the Figure 7(a). We can see that when the prior uncertainty (standard deviation) increases, the test F1 score first increases and then become stable. The reason is that if the initial uncertainty is too small, it cannot cover the feature around the true polyline, which will mislead the following iteration. Second, we change the kernel length scale from 10 to 70. The results in Figure 7(b) show that the performance is best when the kernel length scale is larger than 30. This is because when the length scale is larger, Gaussian Process can capture more correlation between neighbors. In the dataset, the registration error in one point location is highly correlated with near neighbors. Figure 7(c) shows that when we increase the block size k of each segment, the test F1 score first increases and then drops. This is because we assume independence between different segments, and small block sizes will ignore the correlation between points in different blocks. On the other hand, too large block size is computationally inefficient and can cause round-off errors. We also conducted a sensitivity analysis of our model to the distance between the sample points along the imperfect polyline λ as summarized in Figure 7(d). We changed λ from 5, 10, 15 to 20. The best performance was achieved when the candidate point interval size was small (10 meters). The reason is that a small sample point interval provided a high spatial precision in line refinement. The sensitivity analysis of the perpendicular candidates distance are provided in the supplementary material.

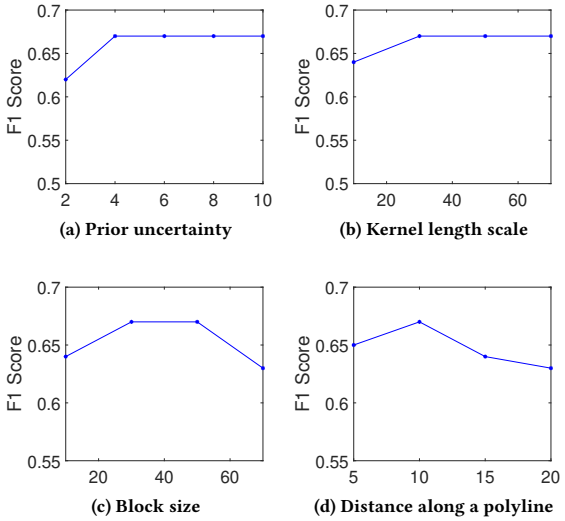


Figure 7: Parameters sensitivity analysis.

4.4 Analysis of computational costs

We evaluated the computational efficiency of our framework. The time costs between iterations were summarized in Figure 8. The blue and orange bars showed the time costs of polyline refinement in the CPU and model re-training in the GPUs, respectively. We can see that the posterior estimation (polyline refinement) part took far less time than the model training. 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 were highly dependent on the hardware platform.

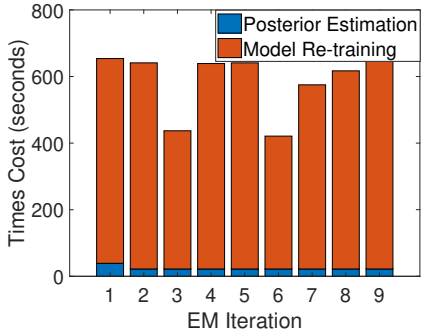


Figure 8: Time cost analysis of training process.

5 CONCLUSION AND FUTURE WORKS

This paper investigates a novel weakly supervised learning framework that explicitly models the uncertainty of vector label registration errors based on a Gaussian process. We propose an uncertainty-aware loss function and design an iterative uncertainty reduction method. Evaluations show that the proposed framework outperforms several baseline methods.

One limitation is we only evaluated the idea for only Earth image segmentation application. We plan to evaluate the proposed approach on other applications such as medical image analysis. We will also generalize the approach from polyline vector labels to polygon vector labels, from binary classes to multiple classes.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. IIS-2147908, IIS-2207072, CNS-1951974, OAC-2152085, and the National Oceanic and Atmospheric Administration (NOAA), UFII Seed Grant, Microsoft AI for Earth Grant and the Extreme Science and Engineering Discovery Environment (XSEDE).

REFERENCES

[1] 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.

[2] Blake Anderson and David McGrew. 2017. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*. 1723–1732.

[3] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. 2019. AutoCorrect: Deep Inductive Alignment of Noisy Geometric Annotations. *arXiv preprint arXiv:1908.05263* (2019).

[4] 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.

[5] 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.

[6] 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.

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

[8] Grant Haskins, Uwe Kruger, and Pingkun Yan. 2020. Deep learning in medical image registration: a survey. *Machine Vision and Applications* 31, 1 (2020), 1–18.

[9] Zhe Jiang, Wenchong He, Marcus Kirby, Sultan Asiri, and Da Yan. 2021. Weakly Supervised Spatial Deep Learning based on Imperfect Vector Labels with Registration Errors. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 767–775.

[10] 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 2017*.

[11] 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.

[12] Simon AA Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus H Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. 2018. A probabilistic u-net for segmentation of ambiguous images. *arXiv preprint arXiv:1806.05034* (2018).

[13] Ranganath Krishnan and Omesh Tickoo. 2020. Improving model calibration with accuracy versus uncertainty optimization. *Advances in Neural Information Processing Systems* 33 (2020), 18237–18248.

[14] 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

[15] Jie Luo, Sarah Frisken, Duo Wang, Alexandra Golby, Masashi Sugiyama, and William Wells III. 2020. Are Registration Uncertainty and Error Monotonically Associated?. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 264–274.

[16] David JC MacKay and David JC Mac Kay. 2003. *Information theory, inference and learning algorithms*. Cambridge university press.

[17] JB Antoine Maintz and Max A Viergever. 1998. A survey of medical image registration. *Medical image analysis* 2, 1 (1998), 1–36.

[18] 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.

[19] 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.

[20] Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1825–1834.

[21] rishizek. 2018. DeepLab-v3-plus Semantic Segmentation. https://github.com/rishizek/tensorflow-deeplab-v3-plus.

[22] 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.

[23] 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.

[24] 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.

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

[26] ZFTurbo. 2018. ZF_UNET_224 Pretrained Model. https://github.com/ZFTurbo/ZF_UNET_224_Pretrained_Model.

[27] Zhen-Yu Zhang, Peng Zhao, Yuan Jiang, and Zhi-Hua Zhou. 2019. Learning from incomplete and inaccurate supervision. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1017–1025.

[28] Barbara Zitova and Jan Flusser. 2003. Image registration methods: a survey. *Image and vision computing* 21, 11 (2003), 977–1000.

6 SUPPLEMENTARY MATERIAL

6.1 Model and Dataset

Model Architecture: 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.

We used the DeepLabv3+ model [4] 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 (ASPP) with 4 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 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 4 double-convolution layers and 4 max-pooling layers. We initialized the backbone model randomly and trained it with the DeepLab model together.

Model Hyper-paramterers: For deep learning base model:The distance interval between two sample points along an imperfect polyline is 10 meters. This value is determined to both maintain a decent sample rate and avoid interference between two consecutive sample points. We set the maximum registration error distance as 25 meters for the Rowan Creek dataset and 75 meters for the Panther Creek dataset. For self-training: The confidence threshold $t = 0.95$. The convergence threshold is 0.001. For patch-based translation: patch size is 16 by 16 and translation distance is 5. The convergence threshold is 0.001. For RuaDL: The convergence threshold is 0.001.

Dataset Description: Training and validation windows were sampled from the same region but were carefully selected to avoid overlapping. Test windows were sampled from a different scene to have independent testing. We used stratified sampling to balance the number of windows that contain the stream class with those that do not. The numbers of training and validation windows are after data augmentation (horizontal and vertical flipping, rotation).

6.2 Comparison on Classification Performance

We also compared the overall classification performance between the DeepLabv3+ model, weakly supervised learning based on self-training, patch-based translation, and our proposed model on the two dataset. The setup was the same as described. The results on second datasets were summarized in Table 4 and Table 5. We can observe that our model significantly improve both the uncertainty estimation performance and classification performance than baselines.

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

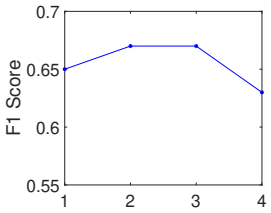


Figure 9: Distance in perpendicular to a polyline sensitivity analysis.

6.3 Theorem proof

In this section, we provide proof for the Theorem 1.

Theorem 6.1. *Assume that the point shift direction is perpendicular to the segment between two points, denote $N_{B(L')}$ as the total pixel number after rasterizing polyline $\mathbf{L}'$, r_i is the shift distance of pixel i along the perpendicular direction of two consecutive points on $\mathbf{L}'$, the projection of pixel i to the polyline $\mathbf{L}'$ is $\mathbf{p}_j$, the variance of point $\mathbf{p}_j$ is σ_j^2, and denote the cumulative density function of $\mathcal{N}(0, \sigma_j^2)$ as Φ_j, then the expected likelihood of polyline $P(\mathbf{L}|\mathbf{X}, \mathbf{\Theta})$ over the polyline prior distribution $\mathbb{E}_{\mathbf{L}\sim P(\mathbf{L}|\mathbf{L}')} \log P(\mathbf{L}|\mathbf{X}, \mathbf{\Theta}) = \sum_i [\Phi_j(r_i + b) - \Phi_j(r_i - b)] \frac{\log p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}}$.*

Proof.

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{L}\sim P(\mathbf{L}|\mathbf{L}')} P(\mathbf{L}|\mathbf{X}, \mathbf{\Theta}) \\
&= \int_{\mathbf{L}} P(\mathbf{L}|\mathbf{L}') \frac{\sum_{(u_i, v_i)\in B(\mathbf{L})} p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}} d\mathbf{L} \\
&= \int_{\mathbf{L}} P(\mathbf{L}|\mathbf{L}') \frac{\sum_i I(u_i \in B(\mathbf{L})) p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}} d\mathbf{L} \\
&= \sum_i \int_{\mathbf{L}} P(\mathbf{L}|\mathbf{L}') \frac{I(u_i \in B(\mathbf{L})) p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}} d\mathbf{L} \\
&= \sum_i \int_{\mathbf{L}} P(\mathbf{L}|\mathbf{L}') I(u_i \in B(\mathbf{L})) d\mathbf{L} \frac{p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}} \qquad (11) \\
&= \sum_i \int_{b-r_i}^{b+r_i} N(\delta_j; 0, \sigma_j) d\delta_j \frac{p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}} \\
&= \sum_i [\Phi_j(r_i + b) - \Phi_j(r_i - b)] \frac{p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}} \\
&= \Phi(\frac{r_j - \mu(s_0) + b}{\sigma(s_0)}) - \Phi(\frac{r_j - \mu(s_0) - b}{\sigma(s_0)}) \frac{p(y_i|\mathbf{X}, \mathbf{\Theta})}{N_{B(\mathbf{L})}}
\end{aligned}
$$

□

6.4 Sensitivity analysis

We also conducted a sensitivity analysis of our model to the distance between two candidate true point locations along the perpendicular direction to a polyline λ_2. The results are summarized in Figure 9. We changed the spatial precision of discretized error distance from 1, 2, 3 to 4 while keeping the same maximum range of shifting (25 meters). We can see that when the distance λ_2 increases (the sample points precision decrease), the test F1 first increases and then drops lower. This is because a small shifting interval provided a high spatial precision in line refinement. But the too small distance between candidate points can be sensitive to the noise of the model prediction.

Table 3: Comparison on classification and uncertainty estimation performance on dataset 2

	Classification Performance						**Uncertainty Estimation Performance**				
Method	Class	Confusion Matrix		Precision	Recall	F1 score	Accuracy	Uncertainty		AvU_A/AvU_I	AvU
		Negative	Positive					Certain	Uncertain		
BaseDL	Non-stream	13482315	236571	0.97	0.99	0.98	Accurate	2417699	78322	0.97	
	Stream	507022	826892	**0.77**	**0.61**	**0.68**	Inaccurate	1297638	49638	0.04	**0.08**
SelfDL		Negative	Positive					Certain	Uncertain		
	Non-stream	13575265	143621	0.98	0.99	0.98	Accurate	2050127	41337	0.98	
	Stream	383744	950170	**0.86**	**0.71**	**0.78**	Inaccurate	1297638	49638	0.04	**0.08**
PatchDL		Negative	Positive					Certain	Uncertain		
	Non-stream	13479015	239871	0.97	0.98	0.98	Accurate	2249534	52361	0.98	
	Stream	407656	926258	**0.80**	**0.70**	**0.75**	Inaccurate	1583466	75639	0.05	**0.09**
WssDL		Negative	Positive					Certain	Uncertain		
	Non-stream	13519788	199098	0.99	0.99	0.99	Accurate	2220900	33411	0.99	
	Stream	137803	1196111	**0.86**	**0.89**	**0.87**	Inaccurate	953853	83013	0.08	**0.15**
RuaDL		Negative	Positive					Certain	Uncertain		
	Non-stream	13544126	174760	0.99	0.99	0.99	Accurate	2569523	152704	0.94	
	Stream	160705	1173209	**0.87**	**0.88**	**0.87**	Inaccurate	576503	799703	0.58	**0.71**

Table 4: Comparison on classification and uncertainty estimation performance on dataset 1 with DeepLabv3+ base model

	Classification Performance						**Uncertainty Estimation Performance**				
Method	Class	Confusion Matrix		Precision	Recall	F1 score	Accuracy	Uncertainty		AvU_A/AvU_I	AvU
		Negative	Positive					Certain	Uncertain		
BaseDL	Non-stream	9827625	70352	0.99	0.99	0.99	Accurate	319911	17474	0.95	
	Stream	89703	47520	**0.40**	**0.35**	**0.38**	Inaccurate	456510	50822	0.10	**0.18**
SelfDL		Negative	Positive					Certain	Uncertain		
	Non-stream	9795675	102302	1.00	0.98	0.99	Accurate	239415	15056	0.94	
	Stream	65988	71235	**0.41**	**0.52**	**0.46**	Inaccurate	460747	33442	0.07	**0.13**
PatchDL		Negative	Positive					Certain	Uncertain		
	Non-stream	9784982	112995	1.00	0.98	0.99	Accurate	265472	41645	0.86	
	Stream	66010	71213	**0.39**	**0.51**	**0.44**	Inaccurate	552875	84551	0.13	**0.22**
RuaDL		Negative	Positive					Certain	Uncertain		
	Non-stream	9844549	53428	1.00	0.99	0.99	Accurate	892608	70084	0.93	
	Stream	47234	89989	**0.63**	**0.64**	**0.64**	Inaccurate	417870	222527	0.26	**0.40**

Table 5: Comparison on classification and uncertainty estimation performance on dataset 2 with DeepLabv3+ base model

	Classification Performance						**Uncertainty Estimation Performance**				
Method	Class	Confusion Matrix		Precision	Recall	F1 score	Accuracy	Uncertainty		AvU_A/AvU_I	AvU
		Negative	Positive					Certain	Uncertain		
BaseDL	Non-stream	13585949	132937	0.98	0.99	0.98	Accurate	2023551	69454	0.97	
	Stream	831849	502065	**0.79**	**0.38**	**0.51**	Inaccurate	1678632	134386	0.07	**0.13**
SelfDL		Negative	Positive					Certain	Uncertain		
	Non-stream	17586361	192079	0.98	0.99	0.98	Accurate	1955324	52139	0.98	
	Stream	877258	1414702	**0.86**	**0.61**	**0.71**	Inaccurate	1438856	64533	0.04	**0.08**
PatchDL		Negative	Positive					Certain	Uncertain		
	Non-stream	13536240	182646	0.97	0.98	0.98	Accurate	2142776	44561	0.98	
	Stream	637782	696132	**0.80**	**0.52**	**0.63**	Inaccurate	1765745	95442	0.05	**0.09**
RuaDL		Negative	Positive					Certain	Uncertain		
	Non-stream	17428621	349819	0.99	0.99	0.99	Accurate	2166744	177856	0.92	
	Stream	515688	1776272	**0.84**	**0.77**	**0.80**	Inaccurate	632465	694478	0.52	**0.70**