DynamicEarthNet: towards FedSPD

March 25, 2025

Overview

1. Data Setting
2. Motivation for Federated Learning
3. Federated Procedure
   Riemannian local optimization
   Differential Privacy Option
   Euclidean local optimization
4. Server Aggregation
   Riemannian aggregation
   Euclidean aggregation

DynamicEarthNet dataset

Multi-spectral images

- 75 Areas of Interest (AOI)
- Satellite images collection: daily 2018/01/01 $\rightarrow$ 2019/12/31
- 730 images/AOI $\rightarrow$ $54,750$ satellite images in total
- 4 channels: RGB+NIR
- Spatial resolution: 3 meters
- Image size: 1024×1024

One time-series: $x \in \mathbb{R}^{T \times H \times W \times 4}$ with T:number of days in the month, $H = W = 1024$.

Land Use/Land Cover (LULC)

- Labels: $24 \times 75 = 1,800$
- 7 different (imbalanced) classes:

class name	%	#AOIs	color
impervious surface	7.1	70	
agriculture	10.3	37	
forest & other vegetation	44.9	71	
wetlands	0.7	24	
soil	28.0	75	
water	8.0	58	
snow & ice	1.0	2	

No single AOI contains all 7 classes.
For a given time series, label:

$$y \in \{0, \ldots, 6\}^{T \times H \times W}$$

Challenges

Issue 1 Voluminous data (hundreds of GB)

Issue 2 Slow to move all images to a single processing server

Why Federated Learning is relevant?

Solution Issue 1 client $\rightarrow$ smaller dataset & own stat. heterogeneity with clear interpretation

Solution Issue 2 No exchange of raw datasets, just each client's updates

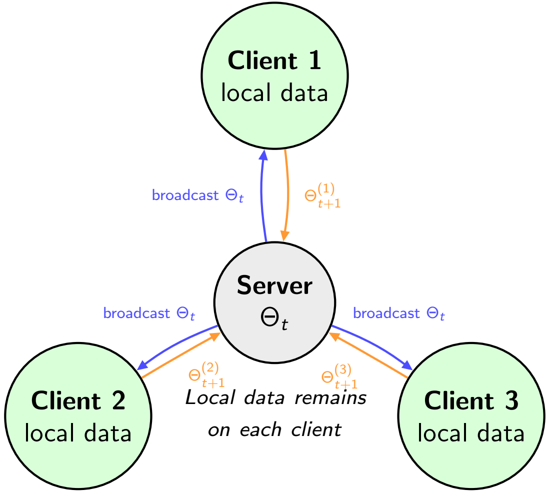
- Each AOI $\rightarrow$ local phenomena
- FL: adapt to local distrib and aggregate knowledge for better generalization

  ⋆ Idea 1 one AOI = one client $\rightarrow$ too many clients (75);

  ⋆ Idea 2 create clients based on dominant LULC categories across AOI

    1. Urban areas
    2. Agricultural areas
    3. Forest-dominated areas
    4. Water-dominated areas
    5. Mixed land-use areas

Federated Learning Overview



Framework

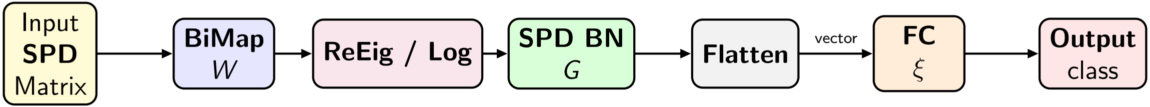
K clients. Each client k has dataset D_k of n_k pixel annotation pairs $\{x_i^{(k)}, y_i^{(k)}\}_{i=1,\ldots,n_k}$, where

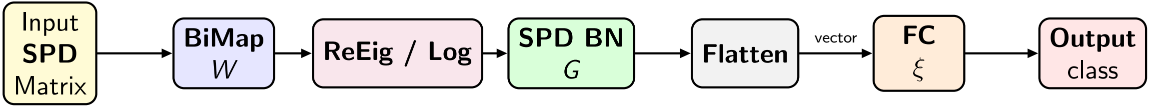
- $x_i^{(k)}$ is a covariance matrix in $\mathbb{R}^{4T\times 4T}$ (with $T = 31$),
- $y_i^{(k)} \in \{0, \ldots, 6\}$ is a semantic mask

Total number of samples $n := \sum_{k=1}^{K} n_k$

Loss function ℓ (CE)

Forward pass SPD-Net $f_\Theta(\cdot)$





Each client's SPD-net with L layers has learnable parameters

$$\Theta = (\underbrace{W_1, \ldots, W_L}_{\textit{BiMap}}, \underbrace{G_1, \ldots, G_L}_{\textit{BN}}, \underbrace{\xi_1, \ldots, \xi_d}_{\textit{FC}}) \in \mathcal{M}$$

with

$$\mathcal{M} := \left(\prod_{l=1}^{L} \mathrm{St}(d_l, d_{l+1}) \right) \times \left(\prod_{l=1}^{L} \mathrm{SPD}_{d_l} \right) \times \mathbb{R}^d.$$

Local objective functions For each client $k = 1, \ldots, K$,

$$F_k(\Theta) := \frac{1}{n_k} \sum_{i=1}^{n_k} \ell\big(f_\Theta(x_i^{(k)}),\, y_i^{(k)}\big).$$

Global objective function

$$\arg\min_{\Theta \in \mathcal{M}} F(\Theta) := \sum_{k=1}^{K} \frac{n_k}{n} F_k(\Theta).$$

Federated procedure

Algorithm

$\star$ **Initialization** Θ_0 to all clients (random)

$\star$ **Outer loop** For $t = 0, \ldots, T-1$ (rounds):

1. A subset $\mathcal{S}_t$ of $r \leq K$ clients is sampled (random)
2. Server broadcasts Θ_t to clients.
3. Each client $k \in \mathcal{S}_t$ runs E_k local updates on Θ with initialization Θ_t.
4. Aggregation of each client's local updated parameters via barycenters $\rightarrow \Theta_{t+1}$

$\star$ **Global output** Final aggregated param. Θ_T or $\mathrm{Unif}(\{\Theta_0, \ldots, \Theta_T\})$

If each local function F_k is L_k-smooth and geodesically convex, then the algorithm converges in $O(1/T)$ (provided some curvature/manifold additional assumptions).

Riemannian local updates at client $k \in \mathcal{S}_r$

Let θ denote a Riemannian parameter (either one of the W_l or G_l).

- Initialize $\theta_0^{(k)} = \theta_t$
- For $s = 1, \ldots, E_k$:

$$\theta_{s+1}^{(k)} \;=\; R_{\theta_s^{(k)}}\Big(-\eta^{(k)}\, g^{(k)}\Big),$$

  where the local Riemannian gradient is

$$g^{(k)} \;=\; \operatorname{grad} F_k(\theta_s^{(k)}) \;-\; \mathcal{T}_{\theta_t \to \theta_s^{(k)}}\Big(\operatorname{grad} F_k(\theta_t) \;-\; \operatorname{grad} F(\theta_t)\Big),$$

  - The blue term is the optional **SVR correction** (variance reduction).

Differential Privacy as a model variant

Besides the variance-reduction approach, we can also inject **DP noise** on each local gradient to protect sensitive data.

Key idea: Add tangent-space Gaussian noise before retraction/exponential update.

- **DP** can be combined with or without the SVR term.
- Each client k ensures local DP by perturbing its gradient:

$$g_{\mathrm{DP}}^{(k)} = g^{(k)} + \varepsilon_t, \quad \text{where} \quad \varepsilon_t \sim \mathcal{N}_{\theta_s^{(k)}}(0, \sigma^2).$$

- This noise is drawn from a *tangent-space Gaussian* distribution at $\theta_s^{(k)}$.

Privacy Guarantee: With an appropriate choice of σ and composition rules, the entire FL pipeline can provide an (ε, δ)-DP guarantee.

Riemannian local updates with optional DP

- Initialize $\theta_0^{(k)} = \theta_t$
- For $s = 1, \ldots, E_k$:

$$g^{(k)} \;=\; \operatorname{grad} F_k(\theta_s^{(k)}) \;-\; \mathcal{T}_{\theta_t \to \theta_s^{(k)}}\Big(\operatorname{grad} F_k(\theta_t) \;-\; \operatorname{grad} F(\theta_t)\Big),$$

$$g_{\mathrm{DP}}^{(k)} \;=\; g^{(k)} \;+\; \varepsilon_s, \qquad \varepsilon_s \;\sim\; \mathcal{N}_{\theta_s^{(k)}}(0, \sigma^2).$$

Then update via retraction (or exponential map):

$$\theta_{s+1}^{(k)} \;=\; R_{\theta_s^{(k)}}\Big(-\eta^{(k)}\, g_{\mathrm{DP}}^{(k)}\Big).$$

- **Red** term is the **DP noise** addition (optional)
- **Blue** term is still the **SVR correction** (optional).

Local updates for Euclidean FC weights (SVR + DP)

Let $\xi \in \mathbb{R}^{d_{\mathrm{FC}}}$ be the parameter vector for the final fully-connected layer. During each local epoch on client k:

$$\xi_0^{(k)} = \xi_t, \quad \text{and for } s = 0, \ldots, E_k - 1 :$$

$$\xi_{s+1}^{(k)} = \xi_s^{(k)} - \eta^{(k)} \Big(\nabla F_k\big(\xi_s^{(k)}\big) - \big[\nabla F_k(\xi_t) - \nabla F(\xi_t)\big] + \varepsilon_s \Big),$$

where:

- $\nabla F_k(\xi_s^{(k)})$: local Euclidean gradient on client k.
- $\big[\nabla F_k(\xi_t) - \nabla F(\xi_t)\big]$ is the **SVR correction** term.
- $\varepsilon_s \sim \mathcal{N}(0, \sigma^2 I)$ is the **DP noise** term.
- After E_k steps, client k sends $\xi_{E_k}^{(k)}$ to the server.

Server aggregation (Karcher flow) for $\theta = G$ SPD

After each client $k \in \mathcal{S}_r$ sends back $\theta_{E_k}^{(k)}$, we compute the Riemannian (Fréchet) mean on the manifold:

$$\Theta_{t+1} \;=\; \arg\min_{\Theta \in \mathcal{M}} \sum_{k \in \mathcal{S}_r} p_k \, d\big(\Theta,\ \theta_{E_k}^{(k)}\big)^2.$$

- In practice, we use a gradient-descent-based procedure (Karcher flow):

$$\Theta_{t+1} \;\leftarrow\; \mathrm{Exp}_{\Theta_t}\Big(-\gamma \sum_{k \in \mathcal{S}_r} p_k \, \mathrm{Log}_{\Theta_t}(\theta_{E_k}^{(k)})\Big),$$

  where γ is a step size for the Karcher flow.

Server aggregation (projection-based) for $\mathbf{W} \in \mathrm{St}(d_l, d_{l+1})$

For the BiMap parameters W_l living on the Stiefel manifold, we can use:

- **Karcher flow** or Riemannian gradient-based aggregator:

$$W_{t+1} \;=\; \arg\min_{W \in \mathrm{St}(d_l, d_{l+1})} \sum_{k \in S_t} p_k \, d^2\big(W,\; W_\tau^{(k)}\big),$$

  which can be iteratively approximated (exp/log maps).

- **Projection-based aggregator**:

$$W_{t+1} \;=\; \mathrm{uf}\Big(\underbrace{\sum_{k \in S_t} p_k \, W_\tau^{(k)}}_{\text{arithmetic mean}}\Big) \quad \text{or} \quad W_{t+1} \;=\; \mathrm{qf}\Big(\sum_{k \in S_t} p_k \, W_\tau^{(k)}\Big).$$

  No iterative scheme needed, computationally cheaper (simple retractions/liftings).

Server aggregation of FC weights

At round t, we receive each client's updated $\xi_{E_k}^{(k)}$ for $k \in \mathcal{S}_t$. To get ξ_{t+1}, standard weighted average (FedAvg):

$$\xi_{t+1} \;=\; \sum_{k \in \mathcal{S}_t} \frac{n_k}{\sum_{j \in \mathcal{S}_t} n_j} \, \xi_{E_k}^{(k)}$$

No need for any Riemannian retraction or manifold-based aggregator.

References