Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)
Special Track on AI for Good

Dynamic Structure Learning through Graph Neural Network for Forecasting Soil Moisture in Precision Agriculture

Anoushka Vyas[1] , **Sambaran Bandyopadhyay**[2*†]
[1]Department of Computer Science, Virginia Tech, Arlington, VA, USA
[2]Amazon
anoushkav@vt.edu, samb.bandyo@gmail.com

Abstract

Soil moisture is an important component of precision agriculture as it directly impacts the growth and quality of vegetation. Forecasting soil moisture is essential to schedule the irrigation and optimize the use of water. Physics based soil moisture models need rich features and heavy computation which is not scalable. In recent literature, conventional machine learning models have been applied for this problem. These models are fast and simple, but they often fail to capture the spatio-temporal correlation that soil moisture exhibits over a region. In this work, we propose a novel graph neural network based solution that learns temporal graph structures and forecast soil moisture in an end-to-end framework. Our solution is able to handle the problem of missing ground truth soil moisture which is common in practice. We show the merit of our algorithm on real-world soil moisture data.

1 Introduction

Precision agriculture [Zhang *et al.*, 2002] is the science of observing, assessing and controlling agricultural practices such as monitoring soil, crop and climate in a field; detection and prevention of pest and disease attacks; providing a decision support system. It can help optimizing the natural resources needed for agricultural activities and thus ensures an efficient, sustainable and environment friendly development of agriculture. Remote sensing and IoT technology [Liaghat *et al.*, 2010] can be used as effective tools in precision agriculture by providing high resolution satellite imagery data with rich information about crop status, crop and water stresses, and ground truth agricultural information from local sensors and agricultural drones. Thus, availability of historical and real-time data has significantly improved the scope of applying artificial intelligence for precision agricultural practices [Jha *et al.*, 2019].

Soil moisture is an important component of precision agriculture [Zhang *et al.*, 2002] for crop health and stress management, irrigation scheduling, food quality and supply chain. Soil moisture measures the amount of water stored in various depths of soil. Forecasting high resolution and accurate soil moisture well ahead of time helps to save natural resources such as water. Soil moisture can be measured by soil sensors in the field [Hummel *et al.*, 2001] or by physics based land surface models [Rui and Beaudoing, 2011]. However, deploying soil moisture sensors on a vast region is expensive. They also cannot provide forecasts. Physics based soil moisture estimation can be quite accurate, but they need extremely rich set of input features such as different soil properties, landscape information, crop information which are difficult to obtain. These physical models are also computationally heavy, making them infeasible to scale over a larger region.

Soil moisture is also directly related to weather parameters such as temperature, precipitation, and vegetation condition of the field such as Normalized Difference Vegetation Index (NDVI) [Pettorelli, 2013]. There are data driven and machine learning approaches present for soil moisture estimation. Conventional techniques such as Bayesian estimation, random forest and support vector regression are commonly used in industry for their simplicity to model soil moisture with respect to these flexible set of input features [Dasgupta *et al.*, 2019; Ahmad *et al.*, 2010]. However, these approaches essentially treat different locations independently while modeling soil moisture and fail to exploit the rich spatial dependency that soil moisture values over a region exhibit. For example, if there is a rainfall, soil moisture values over the whole region increases. In an agricultural region, soil moisture values exhibits similar patterns with varying crop cycles. There exist methods in machine learning to capture such spatial dependencies through latent variable models [Castro *et al.*, 2012] and supervised deep learning architectures [Zhang *et al.*, 2017]. But they are complex in nature (with large number of parameters) and difficult to apply for soil moisture estimation as getting ground truth soil moisture data is expensive because of the cost of deploying physical sensors.

In this paper, we propose soil moisture forecasting as a semi-supervised learning problem on dynamic graphs (also known as temporal graphs or sequence of graphs) where the structure of the graph and node attributes change over time to capture the spatio-temporal variation of soil moisture. In contrast to existing supervised techniques, please note that semi-

*The work was done when Anoushka was an intern at IBM Research and Sambaran was affiliated to IBM Research prior to joining Amazon

†Both Anoushka and Sambaran contributed equally

supervised nature of our solution enables us to train the network on both labeled (data points with ground truth soil moisture) and unlabeled (data points with missing soil moisture) data. This is important because (i) obtaining ground truth soil moisture is expensive and (ii) there can be large amount of missing ground truth soil moisture values because of device and communication failures. A graph can be visualized as a connected (by edges) set of entities (nodes). Intuitively, more the correlation between the soil moisture values in two locations more should be the chance of them to be connected in the graph. Recently, graph representation learning, or more specifically graph neural networks (GNNs) [Wu *et al.*, 2019; Kipf and Welling, 2017; Bandyopadhyay and Peter, 2021] are able to achieve state-of-the-art performance on several learning tasks on graph. GNNs are also proposed to operate on dynamic graphs for traffic forecasting [Li *et al.*, 2018; Chen *et al.*, 2020] and social networks [Pareja *et al.*, 2020]. However, these GNN based approaches cannot be used directly for modeling soil moisture. Unlike social networks where the graph structure is explicitly given [Karrer and Newman, 2011], there is no ground truth graph structure given as an input for the problem of spatio-temporal data modeling. As discussed before, it is possible that the correlation between soil moisture values in two nearby locations is negligible due to multiple hidden factors. Thus, we also need to learn the graph structure of the problem along with the prediction of soil moisture. Recently, research is conducted on graph structure learning using GNNs [Zhu *et al.*, 2021]. However, those approaches are often limited to graphs without any temporal aspect and parameterized over the entries of the discrete adjacency matrix which make the optimization computationally expensive. As discussed in Section 3.3, we tackle these problems in a simple and effective way by introducing multiple regularizers. Following are the novel contributions we make in this paper:

- We pose the problem of soil moisture forecasting as a semi-supervised learning problem on dynamic graphs to jointly capture the varying degrees of spatial and temporal dependencies. As no ground truth graph structure is given as an input, we learn and update the sequence of graph structures in an end-to-end fashion.
- We propose a novel dynamic graph neural network, which is referred as DGLR (Dynamic Graph Learning through Recurrent graph neural network).
- We curate two real world soil moisture datasets. The source code of DGLR and the datasets are available at https://github.com/AnoushkaVyas/DGLR.

2 Problem Formulation

We are given a set of N locations indexed by the set $[N] = \{1, 2, \cdots, N\}$ from a geographic region. There are $[T] = \{1, 2, \cdots, T\}$ time steps. For each time step t, ground truth soil moisture (SM) values are given for a subset of locations $N_{tr}^t \subseteq [N]$. Please note that N_{tr}^{t1} may be different from N_{tr}^{t2} for $t1 \neq t2$ because of the presence of missing SM values.

We use $s_i^t \in \mathbb{R}$ to denote the soil moisture at location i and time step t. For each location i and time step t, there are some D input features (such as temperature, relative humidity, precipitation, NDVI, etc.) available which might be useful to predict soil moisture. Let us denote those feature by an attribute vector $x_i^t \in \mathbb{R}^D, \forall i \in [N], t \in [T]$. We also assume that the geographic distance between any two locations are given. Let us use d_{ij} to denote the distance between two locations i and j. Given all these information, our goal is to predict (forecast) the soil moisture at time step $T+1$ for each location $i \in [N]$, such that the forecasting model considers both past soil moisture and input feature values, and also able to learn and exploit the relation between different locations in their soil moisture values.

In practice, retraining the model for forecasting soil moisture for every time step in future is expensive. So for our experiments in Section 4, we train (including validation) the models roughly on the first 70% to 80% of the time interval and predict soil moisture on the rest.

3 Solution Approach

There are multiple components of our integrated solution DGLR. Please note that all the parameters are trained in an end-to-end fashion. We explain each component below.

3.1 Initial Dynamic Graph Formulation

As explained in Section 1, there is no explicit graph structure given for the problem of soil moisture forecasting. To apply graph neural network in the first iteration, we use the following heuristic to form an initial graph structure. First, for each location $i \in [N]$, we form a node (indexed by i) in the graph. Intuitively, if two locations are very close by, there soil moisture values can be more correlated compared to the points which are far away (there are exceptions to this and our learning algorithm is going to handle those). So, we connect any two nodes by an undirected and unweighted edge in the graph if the distance between the two corresponding locations d_{ij} is less than some pre-defined threshold θ. We also create a self-loop for every node in the graph. We assign attribute vector $x_i^t \in \mathbb{R}^F$ (as discussed in Section 2) to node i at time t. According to this construction, the link structure of the graph is same across different time steps, but node features are changing. Thus, the constructed graph is dynamic in nature[1]. Let us denote the set of graphs as $\{G^1, \cdots, G^T\}$, where $G^t = (V, E^t, X^t)$ is the graph[2] at time step t and $V = [N]$. We denote the adjacency matrix of G^t as A^t. We also row-normalize the adjacency matrix by dividing each row with the degree of the corresponding node.

3.2 Temporal Graph Neural Network

Given a dynamic graph, we develop a temporal graph neural network which can generate embedding of a node in each time step and also use that to forecast soil moisture. Each layer of the proposed temporal graph neural network has two major components. There is a self-attention based GNN (with shared parameters) similar to [Veličković *et al.*, 2018] which

[1] A dynamic or temporal graph is a sequence of (correlated) graphs over time.

[2] Initially, the edge set is same for all the graphs, but they will change during the course of learning.

works on each graph, thus capturing the spatial dependency between the nodes. The updated node embeddings from the GNN is fed to a RNN which connects graphs over different time steps to capture the temporal dependency of soil moisture. The layer is formally discussed below.

For a graph G^t, we use a trainable parameter matrix W (shared over graphs $\forall t \in [T]$) to transform the initial feature vector x_i^t as Wx_i^t. As different neighboring locations may have significantly different impact on the soil moisture of a location, we use a trainable attention vector $a \in \mathbb{R}^{2K}$ to learn the importance α_{ij} for any two neighboring nodes in a graph as follows:

$$\alpha_{ij}^t = \frac{exp\Big(\text{LeakyReLU}\Big(a \cdot [Wx_i^t || Wx_j^t]\Big)\Big)}{\sum\limits_{j' \in \mathcal{N}_{G^t}(i)} exp\Big(\text{LeakyReLU}\Big(a \cdot [Wx_i^t || Wx_{j'}^t]\Big)\Big)} \tag{1}$$

where $\mathcal{N}_{G^t}(i)$ is the neighboring nodes of i (including i itself) in the graph G^t, $\cdot$ represents dot product between the two vectors and $||$ is the vector concatenation. These normalized importance parameters are used to update the node features as:

$$h_i^t = \sigma\Big(\sum_{j \in \mathcal{N}_{G^t}(i)} \alpha_{ij}^t A_{ij}^t W x_j^t\Big) \tag{2}$$

where A_{ij}^t is the $(i,j)$the element of A^t, i.e., weight of the edge (i,j) in G^t.

The sequence of updated node embeddings h_i^t of a node i over different time steps are fed to a Gated Recurrent Unit (GRU) [Chung *et al.*, 2014], which is a popularly used recurrent neural network. The GRU unit for ith node at time step t takes the GNN output h_i^t (from Equation 2) and the GRU output h_i^{t-1} to update h_i^t, as shown below[3].

$$U_i^t = \sigma(W_U^i h_i^{t-1} + P_U^i h_i^{t-1} + B_U^i); \tag{3}$$

$$R_i^t = \sigma(W_R^i h_i^t + P_R^i h_i^{t-1} + B_R^i); \tag{4}$$

$$\tilde{h}_i^t = \tanh(W_H^i h_i^t + P_H^i (R_i^t \circ h_i^{t-1}) + B_H^i); \tag{5}$$

$$h_i^t = (1 - U_i^t) \circ h_i^{t-1} + U_i^t \circ \tilde{h}_i^t \tag{6}$$

where are update and reset gates of GRU at time t, respectively. $W_U^i, W_R^i, W_H^i, P_U^i, P_R^i, P_H^i, B_U^i, B_R^i, B_H^i$ are trainable parameters of GRU for the node i. Please note that we have not shared the parameters of GRUs for different nodes. This helps to boost the performance as the temporal trend of soil moisture values are quite different in different locations. The above completes the description of a GNN+GRU layer of our temporal graph neural network. The output of the GRU of a layer is fed as input to the next layer of GNN. Two such layers of GNN-GRU pair are used for all our experiments. We denote $H^t = [h_1^t \cdots h_N^t]^{Trans}, \forall t \in [T]$.

For soil moisture prediction at time t, node embeddings from the final layer GRU are used. The node embeddings from $t-w$ to $t-1$ are used where w is a window of time steps. We concatenate $h_i^{t-w}, \cdots, h_i^{t-1}$ and pass the vector to a fully connected layer with ReLU activation to predict the soil moisture of i^{th} node at time t, which is denoted as $\hat{s}_i^t$. The window w is a sliding window with stride 1. The loss of soil moisture prediction is calculated as:

$$\mathcal{L}_{STSM} = \sum_{t=w+1}^{T} \sum_{i \in N_{tr}^t} (s_i^t - \hat{s}_i^t)^2 \tag{7}$$

3.3 Updating the Dynamic Graph Structures

In Section 3.1, we use a simple heuristic to form the initial graph which may not always reflect the actual spatial dependency of soil moisture values. In this section, we try to learn the link structure between different nodes to improve the quality of the graph such that it facilitates soil moisture prediction. Graph structure learning has received attention in recent literature. But a wide varieties of approaches are only limited to static graphs without temporal aspects and parameterize all the entries of the adjacency matrix (i.e., learning edges between the nodes) which makes the optimization combinatorial in nature [Fatemi *et al.*, 2021; Selvan *et al.*, 2020; Shang *et al.*, 2021]. Our approach is quite feasible as we learn the temporal graph structures via node embeddings by introducing simple regularizers as discussed below.

First, we reconstruct the adjacency matrix $\hat{A}^t$ of the graph G^t by the similarity of the node embeddings. Formally, we obtain $\hat{A}^t$ from the node embedding matrix H^t as:

$$\hat{A}^t = \text{ReLU}\Big(H^t {H^t}^{Trans}\Big) \in \mathbb{R}^{N \times N} \tag{8}$$

Thus, A_{ij}^t (weight of an edge) is the dot product of the two corresponding node embeddings $h_i^t = H_{i:}^t$ and $h_j^t = H_{j:}^t$. The element-wise use of activation function $\text{ReLU}(\cdot)$ ensures that reconstructed edge weights are non-negative and thus can be used in the message passing framework of GNN. Similar to A^t, we also row-normalize $\hat{A}^t$ by degrees of the nodes. However, the reconstructed graph can be noisy if the initially obtained node embeddings are erroneous. To avoid that, we use following regularization terms.

Graph Closeness. Initially constructed graph with adjacency structure $A^t, \forall t \in [T]$, though not perfect, is easy to explain and intuitive as soil moisture values in close by regions tend to be correlated. So we want the reconstructed graph $\hat{A}^t$ not to move very far away from A^t. To ensure that, we use the following graph closeness regularization:

$$\mathcal{L}_{GC} = \sum_{t=1}^{T} \text{Distance}(A^t, \hat{A}^t) \tag{9}$$

One can use any distance function between the matrices such as KL divergence or any matrix norms. We use binary cross entropy between A^t and $\hat{A}^t$ as that produces better results.

Feature Smoothness. As explained in Section 1, features like weather parameters, vegetation type of the field have significant impact on soil moisture. So, locations having similar features often exhibit similar soil moisture values. It is not necessary that these locations have to be close to each other through geographic distance. Thus, we aim to connect such

[3]We use the same notation h_i^t as the output of both GNN and GRU. But they update it sequentially.

locations in the link structure of the graphs by using the feature smoothness regularizer as shown below:

$$\mathcal{L}_{FS} = \sum_{t=1}^{T} \sum_{\substack{i,j \in [N] \\ i \neq j}} \hat{A}_{ij}^{t} ||x_i^t - x_j^t||_2^2 \tag{10}$$

Minimizing above ensures that for any time step $t \in [T]$, if feature vectors for two nodes i and j are similar, i.e., $||x_i^t - x_j^t||_2^2$ is less, the optimization would try to assign higher values to $\hat{A}_{ij}^t$. Similarly, if $||x_i^t - x_j^t||_2^2$ is high for two locations i and j at time step t, feature smoothness would try to lower the weight of the edge (i, j) even if their geographic distance d_{ij} is low.

Target Smoothness. Finally, if two nodes have similar target variable (which is soil moisture for this work), message passing between them improves the performance of graph neural network, as observed in [Hou *et al.*, 2020]. For example in a graph dataset, if nodes which are directly connected by an edge also share same node labels, it helps the performance of node classification via message passing GNNs. As soil moisture is continuous in nature, we use the square of the difference of two soil moisture values to find their similarity. We calculate this regularization term only on the training set N_{tr}^t for each time step t, where ground truth soil moisture is available.

$$\mathcal{L}_{TS} = \sum_{t=1}^{T} \sum_{\substack{i,j \in N_{tr}^t \\ i \neq j}} \hat{A}_{ij}^{t} (s_i^t - s_j^t)^2 \tag{11}$$

Again, minimizing target smoothness ensures that less edge weights are assigned to a node pair where nodes have very different soil moisture values. Thus, during the application of GNN, it avoids mixing features which lead to different types of soil moisture values.

3.4 Joint Optimization and Training

There are multiple loss components in the overall solution of DGLR. So, we form the final loss function of DGLR by taking a linear combination of all as shown below.

$$\min_{W_G, \Theta_R} \mathcal{L}_{total} = \alpha_1 \mathcal{L}_{STSM} + \alpha_2 \mathcal{L}_{GC} + \alpha_3 \mathcal{L}_{FS} + \alpha_4 \mathcal{L}_{TS} \tag{12}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{R}_+$ are hyperparameters. W_G and Θ_R contain the trainable parameters of GNN and RNN layers respectively. In contrast to existing works which solve graph structure learning as an expensive approximation of a combinatorial optimization problem [Jin *et al.*, 2020; Selvan *et al.*, 2020], the variables in the optimization problem in Equation 12 are only the parameters of GNN and RNNs. Please note that we do not minimize Equations 9-11 directly with respect to the reconstructed adjacency matrix $\hat{A}$, rather it is always calculated as $\hat{A}^t = \text{ReLU}\left(H^t H^{t^{Trans}}\right)$ (Equation 8) (followed by row normalisation). Thus, instead of solving task learning (which in this case is predicting soil moisture) and graph structure learning as alternating minimization, we jointly solve both in the framework of neural networks using ADAM [Kingma and Ba, 2014].

Once we obtain the reconstructed adjacency matrix $\hat{A}^t$ after optimizing Equation 12, we use $\hat{A}^t$, along with node feature matrices X^t, $\forall t \in [T]$ for the next iteration of DGLR. The pseudo code of DGLR is presented in Algorithm 1.

Please note that DGLR is semi-supervised in nature. For example, there can be examples in the training where soil moisture value is not present due to some loss of information or faulty sensors, but other feature information is present. In a conventional supervised setup, such unlabeled examples (without ground truth) are ignored in the training. But in DGLR, both labeled and unlabeled examples are used to propagate information through the graph neural networks (Equation 2). Unlabeled examples can also contribute in the training loss function components such as Equations 9 and 10. Because of the semi-supervised nature, DGLR can work well even with lesser amount of labeled data or missing data, as shown in Section 4.3.

Computation of the GNNs takes $O(MTDK)$ time, where M is the average number of edges in the initial and reconstructed graphs over time. GRU takes another $O(NTK)$ time to generate embeddings. Finally, the loss components collectively takes $O(N^2TK)$ time. Hence, each iteration of DGLR takes $O((N^2 + MD)TK)$ time. One can reduce the runtime of graph update regularizers by adding restriction on two nodes to get connected by an edge if their physical distance is more than some threshold, particularly for larger datasets. Besides, forming a graph across locations which are very far from each other does not make sense because of the significant change of input features, landscape, soil types, etc. Hence, DGLR can be used for real-world soil moisture prediction.

4 Experimental Section

4.1 Datasets and Experimental Setup

In this section, we summarize the datasets, baselines and the setup used for experiments[4]. Obtaining ground truth soil moisture is expensive due to the cost of physical sensors and most of the existing works from agriculture use their private and often interpolated datasets. So, we have curated two soil moisture datasets and the associated features from multiple sources. We refer them as **Spain** and **USA**. We use 6 input features to each location for each time step for Spain and 15 input features for USA. For Spain, they are NDVI obtained from MODIS, SAR back scattering coefficients VV and VH from Sentinel 1, weather parameters consisting of temperature, relative humidity and precipitation. Locally sensed soil moisture is collected from REMEDHUS [Sanchez *et al.*, 2012] for Spain. For USA, the features (soil temperature, weather data, etc.) and soil moisture data are obtained from the SCAN network.

We have used a diverse set of baselines algorithms like **SVR** [Drucker *et al.*, 1997] and **SVR-Shared** (for all the Shared models the parameters being shared for all the stations

[4]More details can be found in the longer version of the work at https://arxiv.org/abs/2012.03506

Algorithm 1 DGLR

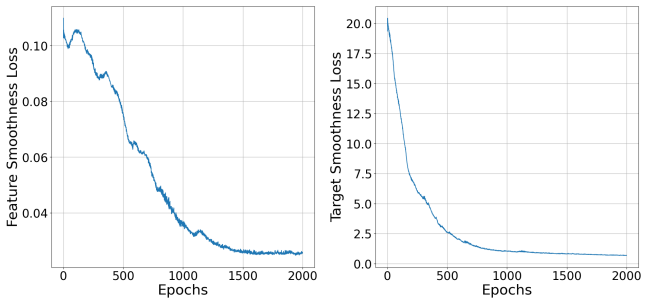
Input: Soil moisture locations $[N]$ and distances $d_{ij}, \forall i, j \in [N]$ between them. Temporal attribute matrix $X^t = \{x_i^t \mid \forall i \in [N]\}$ for each time step $t \in [T+L]$. Ground truth soil moisture values for (a subset of) locations from time $1, \cdots, T$. Window length w.
Output: Predict soil moisture values of all the locations from $T+1, \cdots, T+L$.
Training
Construct an initial attributed graph G^t with normalized adjacency matrix A^t and node attribute matrix X^t for each time $t \in [T]$ where each node represents one location (Section 3.1).
Set $\tilde{A}^t = A^t, \forall t \in [T]$
for pre-defined number of iterations **do**
    Pass $\tilde{A}^t$ and X^t to two layers of GNN+RNN pairs (Section 3.2) to generate all the node embeddings, $\forall t \in [T]$.
    Concatenate $h_i^{t-w}, \cdots, h_i^{t-1}$ and pass it to a fully connected layer to predict soil moisture $\hat{s}_i^t$, $\forall i$ and $\forall t \in \{w+1, \cdots, T\}$
    Update the parameters of GNN and RNNs by minimizing the losses at Equation 12.
    Obtain $\hat{A}^t$ by Equation 8, followed by row normalization and set $\tilde{A}^t = \hat{A}^t, \forall t \in [T]$
end for
Soil Moisture Prediction
Pass the reconstructed graph from the last time step of training, along with attribute matrices $X^{t-w} \cdots X^{t-1}$ to the trained GNN+RNN model to predict soil moisture at time t, $t = T+1, \cdots, T+L$.

in a dataset), **Spatial-SVR** where spatial information in SVR is used by concatenating a node's feature with the features of its nearest k neighbors, **ARIMA** [Hannan and Rissanen, 1982], **RNN** [Chung *et al.*, 2014] and **RNN-Shared**, popular spatio-temporal GNN algorithms like **DCRNN** [Li *et al.*, 2018] and **EvolveGCN** [Pareja *et al.*, 2020].

Hyperparameters are tuned by using the validation set and test results are reported at the best validation epoch. The embedding dimension is set to 10 for Spain and 20 for USA. We run the algorithms for 2000 epochs on Spain and 500 epochs on USA. We used three different metrics during the test period of each time series. They are Root Mean Square Error (RMSE), Symmetric Mean Absolute Percentage Error (SMAPE) and Correlation Coefficient.

4.2 Results and Analysis

We run each algorithm 10 times on each dataset and reported the average metrics over all the locations. Table 1 shows the performance of all the baselines, along with DGLR and its variants. First, we can see that algorithms with non-shared parameters across the locations in a dataset performs better than their counterparts with shared parameters. This is because soil moisture patterns in a dataset varies significantly from one location to other. Next, the performance of dynamic GNN based baselines like DCRNN and EvolveGCN are not always better than a deep sequence modeling technique like RNN. This shows that using spatial information (via graph) may not always lead to a better performance, especially when the initially constructed graph is noisy. Finally, we find that DGLR is able to achieve better performance than all the baseline algorithms considered.



(a) Feature Smoothness (b) Target Smoothness

Figure 1: 1a and 1b show loss components to depict the evolution of the graph structure of Spain over the iterations of DGLR (best seen in color).

Comparing the performance of DGLR with its variants, we can see the importance of different components of DGLR such as non-shared parameters of the GRU layers across locations, the importance of learning the graph structure and the role of feature and target smoothness regularizers.

4.3 Sensitivity to Missing Soil Moisture Values

The datasets that we used here do not have any missing soil moisture values. But for real-world applications, it is possible to have missing soil moisture values in between due to problem in the local sensors or in the communication networks. DGLR, being a semi-supervised approach due to use of graph neural networks, can be trained on labeled and unlabeled data points together. To see the impact of missing ground truth values in the training, we randomly remove $p\%$ soil moisture values from the training data, where $p \in \{10, 20, 30\}$. Table 2 shows the test set performance of DGLR with different proportions of missing soil moisture values in the training. Please note that we still use the input features from those locations with missing soil moisture values and they do participate in the message passing framework of GNN. From Table 2, the performance of DGLR drops very slowly with more amount of missing soil moisture values in the training. In fact, its performance with 20% missing values is still better than most of the baselines in Table 1 with no missing values. So, it is clear that DGLR is quite robust in nature, and thus can be applicable for real-world soil moisture prediction.

4.4 Sensitivity to Initially Constructed Graph Structure

We use a simple heuristic to construct an initial graph in Section 3.1 on which we apply the proposed dynamic GNN to forecast soil moisture and update the graph structures over the iterations. In this section, we check the sensitivity of DGLR on different initially constructed graph. To do this, we vary the threshold parameter θ (Section 3.1). As the value of θ increases, nodes with larger distance will get connected in the initial graph, increasing the overall density. Interestingly, Table 3 shows that the performance of DGLR does not change

Algorithms	Spain			USA		
	RMSE (↓)	SMAPE % (↓)	Correlation (↑)	RMSE (↓)	SMAPE % (↓)	Correlation (↑)
SVR-Shared	0.061 ± 0.004	33.654 ± 1.236	0.020 ± 0.001	12.332 ± 0.848	39.013 ± 3.395	0.287 ± 0.031
SVR	0.052 ± 0.005	23.840 ± 2.453	0.083 ± 0.002	10.461 ± 0.718	27.377 ± 3.314	0.354 ± 0.031
Spatial-SVR	0.052 ± 0.007	23.752 ± 1.562	0.150 ± 0.013	10.245 ± 1.567	26.710 ± 4.049	0.351 ± 0.049
ARIMA	0.041 ± 0.008	19.002 ± 0.872	0.010 ± 0.001	9.290 ± 1.110	27.785 ± 1.306	0.159 ± 0.016
RNN-Shared	0.039 ± 0.003	23.443 ± 1.996	0.585 ± 0.048	7.814 ± 0.442	30.880 ± 2.355	0.191 ± 0.041
RNN	0.039 ± 0.003	21.722 ± 1.385	0.529 ± 0.049	7.338 ± 0.346	27.833 ± 1.812	0.273 ± 0.046
DCRNN	0.061 ± 0.003	31.832 ± 2.363	0.588 ± 0.062	8.161 ± 0.762	25.213 ± 1.762	0.393 ± 0.015
EvolveGCN	0.061 ± 0.004	31.789 ± 2.554	0.731 ± 0.022	8.526 ± 0.519	27.717 ± 1.919	0.415 ± 0.038
DGLR (Shared)	0.037 ± 0.003	17.533 ± 1.000	0.752 ± 0.020	7.526 ± 0.519	20.700 ± 1.210	0.493 ± 0.031
DGLR (w/o SL)	0.035 ± 0.003	15.599 ± 0.940	0.753 ± 0.017	6.970 ± 0.356	18.360 ± 0.987	0.517 ± 0.023
DGLR (w/o Sm)	0.033 ± 0.002	19.835 ± 1.005	0.759 ± 0.018	6.721 ± 0.176	17.991 ± 1.211	0.533 ± 0.013
DGLR (full model)	$\mathbf{0.031 \pm 0.001}$	$\mathbf{15.583 \pm 1.008}$	$\mathbf{0.764 \pm 0.017}$	$\mathbf{6.454 \pm 0.111}$	$\mathbf{17.002 \pm 0.987}$	$\mathbf{0.566 \pm 0.008}$

Table 1: Performance of soil moisture forecasting in test interval of different datasets. DGLR, as it exploits both spatial and temporal nature of the problem along with learns the graph structure, is able to perform better in most of the cases.

Missing SM %	Spain		
	RMSE (↓)	SMAPE % (↓)	Correlation (↑)
0%	0.031 ± 0.001	15.583 ± 1.008	0.764 ± 0.017
10%	0.035 ± 0.004	18.121 ± 1.430	0.750 ± 0.033
20%	0.039 ± 0.002	18.864 ± 1.516	0.738 ± 0.027
30%	0.039 ± 0.006	18.962 ± 1.112	0.661 ± 0.047

Table 2: Test set performance of DGLR with different proportions of missing soil moisture values in the training set.

Distance Threshold (km)	Spain		
	RMSE (↓)	SMAPE % (↓)	Correlation (↑)
8	0.034 ± 0.002	16.418 ± 0.810	0.752 ± 0.042
12	0.032 ± 0.003	16.441 ± 1.106	0.752 ± 0.028
16	0.031 ± 0.001	15.583 ± 1.008	0.764 ± 0.017
20	0.032 ± 0.004	16.556 ± 1.099	0.754 ± 0.054

Table 3: Test set performance of DGLR with different distance thresholds in initialising the graph structure.

much with different values of θ. This happens because DGLR also learns the optimal graph structure for soil moisture prediction. Thus, it is less sensitive to the heuristic-based initially constructed graph.

4.5 Evolution of Graph Structure

One key component of DGLR is to learn and update the graph structure as it predicts the soil moisture values (Section 3.3). Feature Smoothness (FS) loss (Eq. 10) measures if nodes having dissimilar features are connected. Target Smoothness (TS) loss (Eq. 11) measures if nodes having dissimilar soil moisture values are connected. For Spain, these losses over different iterations (Step 3 of Alg. 1) of DGLR are shown in Figure 1. Both FS loss and TS loss decrease significantly after the initial iterations and stabilize during the end of training. Thus, DGLR is able to connect nodes with similar features and similar soil moisture values by edges over the training such a way that it also helps to predict the soil moisture better.

5 Discussion and Conclusion

In this work, we have addressed the problem of soil moisture modeling and prediction which is of immense practical importance. We propose a semi-supervised dynamic graph neural network which also learns the temporal graph structures iteratively to predict soil moisture. Our solution is robust in nature with respect to missing ground truth soil moisture values and also able to achieve state-of-the-art performance for data driven soil moisture prediction on real-world soil moisture datasets. We hope that our solution along with the publicly available source code and datasets would help further AI research in this direction. In future, we want to apply this solution for other types of spatio-temporal data with different types of domain specific challenges. We will also check the impact of our solution in broader sustainability aspects such as improving production and quality of crops, preserving natural resources, etc.

References

[Ahmad *et al.*, 2010] Sajjad Ahmad, Ajay Kalra, and Haroon Stephen. Estimating soil moisture using remote sensing data: A machine learning approach. *Advances in Water Resources*, 33(1):69–80, 2010.

[Bandyopadhyay and Peter, 2021] Sambaran Bandyopadhyay and Vishal Peter. Unsupervised constrained community detection via self-expressive graph neural network. In *Uncertainty in Artificial Intelligence*, pages 1078–1088. PMLR, 2021.

[Castro *et al.*, 2012] Marisol Castro, Rajesh Paleti, and Chandra R Bhat. A latent variable representation of count data models to accommodate spatial and temporal dependence: Application to predicting crash frequency at intersections. *Transportation research part B: methodological*, 46(1):253–272, 2012.

[Chen *et al.*, 2020] Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. Multi-range attentive bicomponent graph convolutional network for traffic

forecasting. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 3529–3536, 2020.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[Dasgupta *et al.*, 2019] Kalyan Dasgupta, Kamal Das, and Manikandan Padmanaban. Soil moisture evaluation using machine learning techniques on synthetic aperture radar (sar) and land surface model. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5972–5975. IEEE, 2019.

[Drucker *et al.*, 1997] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[Fatemi *et al.*, 2021] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[Hannan and Rissanen, 1982] Edward J Hannan and Jorma Rissanen. Recursive estimation of mixed autoregressive-moving average order. *Biometrika*, 69(1):81–94, 1982.

[Hou *et al.*, 2020] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations*, 2020.

[Hummel *et al.*, 2001] John W Hummel, Kenneth A Sudduth, and Steven E Hollinger. Soil moisture and organic matter prediction of surface and subsurface soils using an nir soil sensor. *Computers and electronics in agriculture*, 32(2):149–165, 2001.

[Jha *et al.*, 2019] Kirtan Jha, Aalap Doshi, Poojan Patel, and Manan Shah. A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, 2:1–12, 2019.

[Jin *et al.*, 2020] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '20, page 66–74, 2020.

[Karrer and Newman, 2011] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[Li *et al.*, 2018] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[Liaghat *et al.*, 2010] Shohreh Liaghat, Siva Kumar Balasundram, et al. A review: The role of remote sensing in precision agriculture. *American journal of agricultural and biological sciences*, 5(1):50–55, 2010.

[Pareja *et al.*, 2020] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, pages 5363–5370, 2020.

[Pettorelli, 2013] Nathalie Pettorelli. *The normalized difference vegetation index*. Oxford University Press, 2013.

[Rui and Beaudoing, 2011] Hualan Rui and Hiroko Beaudoing. Readme document for global land data assimilation system version 2 (gldas-2) products. *GES DISC*, 2011, 2011.

[Sanchez *et al.*, 2012] Nilda Sanchez, José Martínez-Fernández, Anna Scaini, and Carlos Perez-Gutierrez. Validation of the smos l2 soil moisture data in the remedhus network (spain). *IEEE Transactions on Geoscience and Remote Sensing*, 50(5):1602–1611, 2012.

[Selvan *et al.*, 2020] Raghavendra Selvan, Thomas Kipf, Max Welling, Antonio Garcia-Uceda Juarez, Jesper H Pedersen, Jens Petersen, and Marleen de Bruijne. Graph refinement based airway extraction using mean-field networks and graph neural networks. *Medical Image Analysis*, page 101751, 2020.

[Shang *et al.*, 2021] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[Zhang *et al.*, 2002] Naiqian Zhang, Maohua Wang, and Ning Wang. Precision agriculture—a worldwide overview. *Computers and electronics in agriculture*, 36(2-3):113–132, 2002.

[Zhang *et al.*, 2017] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[Zhu *et al.*, 2021] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 2021.