Easy Begun Is Half Done: Spatial-Temporal Graph Modeling with ST-Curriculum Dropout

Hongjun Wang[1,2§], Jiyuan Chen[1,2§], Tong Pan[3], Zipei Fan[4 †], Xuan Song[1,2†], Renhe Jiang[4,5], Lingyu Zhang[1,2,7], Yi Xie[6], Zhongyi Wang[6], and Boyuan Zhang[1,2]

[1] SUSTech-UTokyo Joint Research Center on Super Smart City, Southern University of Science and Technology
[2] Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology
[3] Department of Physics, The Chinese University of Hong Kong
[4] Center for Spatial Information Science, University of Tokyo
[5] Information Technology Center, University of Tokyo
[6] Huawei Technologies CO.LTD
[7] Didichuxing Inc

Abstract

Spatial-temporal (ST) graph modeling, such as traffic speed forecasting and taxi demand prediction, is an important task in deep learning area. However, for the nodes in graph, their ST patterns can vary greatly in difficulties for modeling, owning to the heterogeneous nature of ST data. We argue that unveiling the nodes to the model in a meaningful order, from easy to complex, can provide performance improvements over traditional training procedure. The idea has its root in Curriculum Learning which suggests in the early stage of training models can be sensitive to noise and difficult samples. In this paper, we propose ST-Curriculum Dropout, a novel and easy-to-implement strategy for spatial-temporal graph modeling. Specifically, we evaluate the learning difficulty of each node in high-level feature space and drop those difficult ones out to ensure the model only needs to handle fundamental ST relations at the beginning, before gradually moving to hard ones. Our strategy can be applied to any canonical deep learning architecture without extra trainable parameters, and extensive experiments on a wide range of datasets are conducted to illustrate that, by controlling the difficulty level of ST relations as the training progresses, the model is able to capture better representation of the data and thus yields better generalization.

Introduction

With the advances in deep learning techniques such as Convolutional Neural Network (CNN) and Graph Neural Networks (GNN), spatial-temporal graph modeling has been receiving increasing attention. The basic assumption behind it is that a node's future state is conditioned on its historical state as well as its spatial neighbors' historical state (Wu et al. 2019). Thus, the modeling target seeks to discover both the node-level spatial inter-dependent relations and the node's self-dependent temporal relations. The task has many real-world applications like traffic speed forecasting (Yu, Yin, and Zhu 2017) and human action recognition (Yan, Xiong, and Lin 2018).

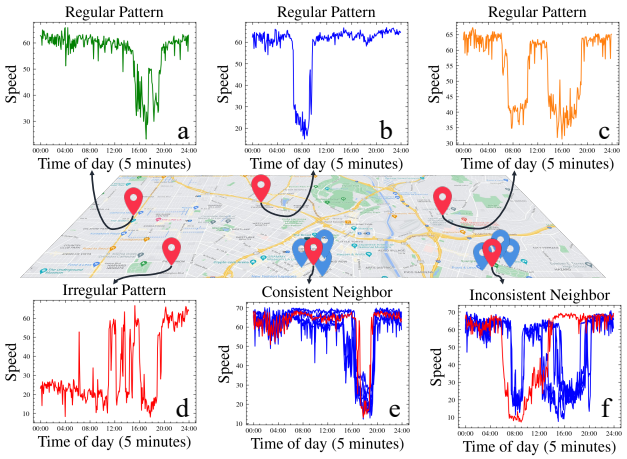


Figure 1: Speed data visualization in METR-LA dataset.

While plenty of attention has been devoted to designing more sophisticated model architectures so as to capture better ST relations, we are convinced that it's also valuable to take a step back and focus on an essential yet easily-overlooked question: *Can we simply use a more advanced training strategy to improve model performances?* Almost all the existing work in ST modeling tends to treat the nodes in the graph equally and feed them into the neural network. However, chances are that the difficulty of modeling a node's ST pattern can vary greatly. Figure 1 (a), (b) and (c) show the regular speed pattern of nodes in METR-LA dataset (Li et al. 2017), which demonstrate that most of the nodes are only significantly affected by the morning and/or evening peaks, when heavy traffic congestions might happen. These regular temporal patterns are easy for models to learn. Nevertheless, Figure 1 (d) shows an example of nodes whose speed patterns are highly disordered and unpredictable, and their data representation can be very different from the majority

[§]Equal contribution.
[†]Corresponding to: Zipei Fan (fanzipei@iis.u-tokyo.ac.jp) and Xuan Song (songx@sustech.edu.cn).

of nodes. This is an irregular one which is hard to model. For Figure 1 (e) and (f), as we mentioned before, ST graph modeling also relies on the node-level spatial inter dependency. Figure 1 (e) shows the circumstance where a node (red line) and its first-order spatial neighbors' (blue lines) speed patterns are very alike. This spatial relation without doubt can be easily captured by the model, while Figure 1 (f) shows the opposite way where the node (red line) doesn't share the same trend as its neighbors (blue lines). Although in Figure 1 (f) the temporal speed patterns of both the node and its neighbors are easy to learn, their spatial relations can be confusing for the model especially at early training stages. Extensive research has suggested that presenting the training samples in a meaningful order, from easy to hard, can benefit the training process and such strategy is referred to as Curriculum Learning (Kumar, Packer, and Koller 2010; Xu et al. 2020; Wang et al. 2021).

Curriculum Learning (CL) follows the intuitions of humans: learning should start with the fundamental knowledge and then gradually build on that to handle more complicated ones. The idea is first proposed by (Bengio et al. 2009) and we've witnessed its successful applications in many fields like image classification (Srivastava et al. 2014), text classification (Ma et al. 2017) and object segmentation (Kumar et al. 2011). From the view of optimization, curriculum learning excludes the impacts from difficult or even noisy samples at the early stage of training when the model is vulnerable, and is thus able to prevent the model from getting stuck in bad local minima, making it converge faster and reach a better solution than the standard approach. The success of curriculum learning, especially in highly non-convex deep models, gives us the motivation to introduce it into the task of spatial-temporal graph modeling, and to the best of our knowledge, such an idea still remains unexplored.

In previous work, the smallest unit to be assigned a difficulty level is a data sample (e.g., for image classification, a data sample should be a single image), and the curriculum is built by progressively increasing the difficulty threshold for input data. However, in the context of spatial-temporal graph modeling, a data sample is a graph consisting of multiple spatially-connected nodes with their temporal features. Therefore, it is hard to rank two data samples' difficulties when one's nodes suffer from messy spatial relations, and the others' nodes suffer from terribly orderless temporal patterns. In a word, as a graph, the data sample here lacks the necessary atomicity to be analyzed as a whole. On the contrary, we find that the node in graph is atomic and itself contains sufficient ST information (spatial neighborhood relationships in spatial view and its own time series values in temporal view). Therefore, unlike previous work, we innovatively set the smallest unit as a node instead of the whole graph and define our curriculum as gradually introducing the nodes with complex ST relations to the model.

The key challenge towards utilizing curriculum learning lies in determining easy/hard nodes. In this paper, we propose to evaluate the difficulty of nodes in high dimensional feature space. Specifically, we define easy nodes as those whose data representations locate in high-density area of the feature space and are consistent with their spatial neighbors on graph (we denote the neighbors on graph as **G-neighbors** for simplicity). Conversely, difficult nodes have their data representation either solitarily far away from the majority or inconsistent with its G-neighbors. Then based on these difficulty scores, we propose our ST-Curriculum Dropout as a novel training strategy specially for spatial-temporal graph modeling. The idea is inspired by (Sinha, Garg, and Larochelle 2020) which smooths the feature space with a continuously weakened Gaussian kernel. We take a braver step by directly dropping out the nodes with sophisticated ST relations at the early stage of training, such that the model can only propagate and learn the most fundamental ST relations. We further propose a smooth curriculum arrangement function to progressively introduce harder nodes into the model, allowing at each training step, the model can focus on "interesting" ST relations, that are near its border of capability, neither too easy nor too hard, to expand the border gradually.

The main contributions in this paper are listed as follows:

- We explore and make novel adaptions of CL into spatial-temporal graph modeling. To the best of our knowledge, this is the first time curriculum learning is proven to be prospective in such task.
- We propose a novel CL training strategy, which can also be formulated as an external plug-in for canonical deep learning models, along with a smooth CL arrangement function for ST graph modeling task.
- We obtained universal performance gain on a wide range of real-world datasets, illustrating the effectiveness of our design.

Preliminaries

Definition 1 ***Graph***. *In this paper, a graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ indicates the set of edges, and A is the adjacency matrix derived from graph $\mathcal{G}$. At each time step t, the graph $\mathcal{G}$ has a dynamic feature matrix $\mathbf{X}_t \in \mathbb{R}^{|\mathcal{V}| \times \mathcal{D}}$ and $\mathcal{D}$ is the dimension of node features.*

Definition 2 ***Spatial-Temporal Graph Modeling***. *Given a graph $\mathcal{G}$ and its N step historical observation $\mathbf{X}_{t-N:t}$, the task of ST graph modeling is to learn a neural network f which is able to predict the future S step graph features $\overline{\mathbf{X}}_{t+1:t+S}$. We summarize the mapping relation as:*

$$f : \left[\mathbf{X}_{(t-N):t}, \mathcal{G}\right] \mapsto \overline{\mathbf{X}}_{(t+1):(t+S)}.$$

Methodology

In this section, we decompose our ST-Curriculum Dropout (abbreviated as STC-Dropout) into two stages: Difficulty Evaluation and Curriculum Learning. In the first stage, each node in graph will be assigned a score reflecting its difficulty of ST relations. This is done by measuring both a node's data distribution density (temporal view) and its consistency with G-neighbors (spatial view). In the second stage, based on these scores, nodes are exposed to the model in an easy-to-difficult fashion, resulting in the final curriculum that the

The related work of this paper is included in Appendix A.1

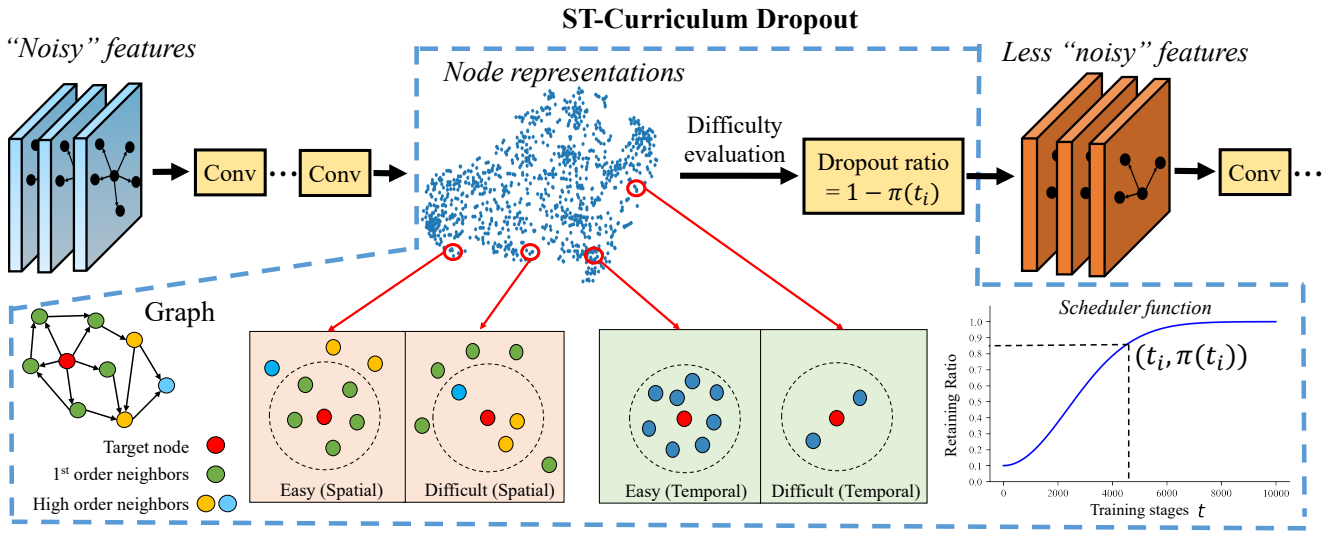


Figure 2: The schematic of ST-Curriculum Dropout is shown above as a plug-in to deep learning models. In high-level feature space, nodes with difficult ST-relations will be dropped out at a certain ratio to "denoise" the data, ensuring a smoother objective for the model to optimize at early training stages. The ratio to retain difficult nodes will be progressively increased during training until the overall utilization.

model will be trained on. The schematic of our method is shown in Figure 2 as a plug-in and the details will be elaborated next.

Difficulty Evaluation

At the core of our idea is to develop a general difficulty evaluation justified for all possible models in spatial-temporal graph modeling tasks, even though in this paper only GNNs are conducted as an example. We analyze the nodes' high-level representations retrieved from previous layers in the model. Here, one might question why we don't directly analyze the original data itself rather than its high-level representation. However, this is non-trivial since models always encode the data through advanced non-linear transformations, so what might seem reasonable to a human in the original data might not be considered the same way by the model in its high-dimension feature space. Therefore, we argue that the difficulty scores should be derived from the high-level data representation inside the model so that we can "eliminate" what the model itself thinks difficult or disturbing to its training. This idea shares a consensus with Self-Paced-Learning (Kumar, Packer, and Koller 2010), where the author claims the models should take charge in deciding the difficulties of samples.

With the node's high-level representation which infers the specific semantic information perceived by the model, we can now evaluate the difficulties. For the sake of clarity, let's first remind the basic assumption of ST graph modeling: a node's future state is conditioned on its historical state as well as its G-neighbors' historical state. Intuitively, the easier nodes should be consistent with G-neighbors (spatial view) and have their high-level data representations in the high density area of the distribution (temporal view). On the contrary, more difficult nodes should be the ones either have data representation solitarily far away from the majority or are inconsistent with its G-neighbors.

Formally, let $h^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d^{(l)}}$ indicate the $d^{(l)}$-dimensional data representation of all nodes, which is generated by the $(l-1)^{th}$ layer $f^{(l-1)}$ with $h^{(l-1)}$ as

$$h^{(l)} = f^{(l-1)}(h^{(l-1)}). \tag{1}$$

For node v_i and its representation $h_i^{(l)} \in \mathbb{R}^{d^{(l)}}$, we build a $d^{(l)}$-dimensional ball with radius R centered at $h_i^{(l)}$:

$$\mathcal{B}\left(h_i^{(l)}, R\right) = \left\{ h_j^{(l)} \mid D(h_j^{(l)}, h_i^{(l)}) \leq R \right\}, \tag{2}$$

where D is the distance function. We normalize the vectors here to eliminate the effect of variances along different directions and apply Cosine Distance to alleviate the curse of dimensionality in high-level feature space. Then, in order to measure the node's consistency with G-neighbors, we here further introduce the concept of k order neighbors.

Definition 3 *k order neighbors. If node v_j is a k order neighbor of node v_i on graph $\mathcal{G}$, there exists a path $P_{v_i \to v_j}$: $v_i \to v_m \to \cdots \to v_j$ which satisfies $|P_{v_i \to v_j}| - 1 \leq k$.*

Let $\Omega_{v_i}^k = \{v_j \mid |P_{v_i \to v_j}| - 1 \leq k\}$ denotes the set of nodes being the k order neighbors of v_i, an indicator function $\mathcal{I}$ can be further defined as

$$\mathcal{I}_{v_i}(v_j, \Omega_{v_i}^k) = \begin{cases} 1, & v_j \in \mathcal{B}\left(h_i^{(l)}, R\right) \wedge v_j \in \Omega_{v_i}^k \\ 0, & \text{otherwise} \end{cases}$$

We can now count the portion of node v_i's k order neighbors which are wrapped by the ball $\mathcal{B}\left(h_i^{(l)}, R\right)$ as:

$$\frac{1}{|\,\Omega_{v_i}^k\,|} \sum_{v_j \in \mathcal{V}} \mathcal{I}_{v_i}(v_j, \Omega_{v_i}^k), \tag{3}$$

Intuitively, the higher the fraction of node v_i's k order neighbors captured by the ball, the more consistent v_i should be with its G-neighbors, which indicates an easier spatial relation for the model to capture. However, for different nodes, their data distribution density (i.e., if a node's high-level data representation are intensively surrounded by many of others) can vary by orders of magnitudes, which the simple ratio in Formula (3) can not properly capture. Take node v_i as an example, if ball $\mathcal{B}\left(h_i^{(l)}, R\right)$ wraps massive nodes inside, which means $h_i^{(l)}$ is located in the high density area of the feature space, then v_i is a normal node with easier temporal patterns and has many peers similar to itself. If the nodes wrapped by the ball further account for a large portion of v_i's k order neighbor, then the pattern of v_i should be easy to learn in both spatial and temporal aspects. Conversely, if v_i's high-level representation h_i^l stays solitarily away from others (i.e., in low density area), then v_i is an outlier in the dataset. In this case, v_i should be more difficult to learn in temporal aspect regardless of its consistency with G-neighbors.

Therefore, in addition to Formula (3) which serves as a difficulty measurer in spatial aspect, we further propose Formula (4) to measure the node's difficulty from temporal aspect as follows:

$$\frac{|\mathcal{B}\left(h_i^{(l)}, R\right)|}{|\mathcal{B}\left(h_i^{(l)}, R\right)| + \epsilon_R}, \tag{4}$$

where $\epsilon_R = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}|\mathcal{B}\left(h_i^{(l)}, R\right)|$ acts as a penalty term for those "lonely" nodes. Specifically, the functionality of Formula (4) mainly relies on three important properties: 1) The value of Formula (4) increases monotonically as the node's distribution density becomes larger (i.e., $|\mathcal{B}\left(h_i^{(l)}, R\right)|$ gets larger). 2) The value of ϵ_R represents the average data distribution density and serves as a penalizer. When $|\mathcal{B}\left(h_i^{(l)}, R\right)| \gg \epsilon_R$, we have $\frac{|\mathcal{B}\left(h_i^{(l)}, R\right)|}{|\mathcal{B}\left(h_i^{(l)}, R\right)|+\epsilon_R} \approx 1$. Conversely, when $|\mathcal{B}\left(h_i^{(l)}, R\right)| \ll \epsilon_R$, we have $\frac{|\mathcal{B}\left(h_i^{(l)}, R\right)|}{|\mathcal{B}\left(h_i^{(l)}, R\right)|+\epsilon_R} \approx 0$. The penalty strength of ϵ_R actually grows as the node's data distribution density decreases. 3) The range of Formula (4) is constrained to $[0, 1]$, which is consistent with the range of Formula (3). Therefore, by integrating Formula (3) and Formula (4), we propose our difficulty evaluation function $Diff(v_i)$ as:

$$2 - \underbrace{\frac{1}{|\ \Omega_{v_i}^k\ |}\sum_{v_j \in \mathcal{V}} \mathcal{I}_{v_i}(v_j, \Omega_{v_i}^k)}_{\text{Spatial difficulty}} - \underbrace{\frac{|\mathcal{B}\left(h_i^{(l)}, R\right)|}{|\mathcal{B}\left(h_i^{(l)}, R\right)| + \epsilon_R}}_{\text{Temporal difficulty}}. \tag{5}$$

Moreover, to avoid excessive hyperparameters, we here heuristically define the radius R in Eq. (5) as $R = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\mathcal{Q}_\rho(\{D(h_i^{(l)}, h_j^{(l)}) : j \in \{1, \cdots, |\mathcal{V}|\}\})$, where $\mathcal{Q}$ is a quantile function that return the $\rho\%$ quantile values expanding from small to large.

Curriculum Learning

In this section, we start from the formalization of curriculum learning (Bengio et al. 2009). Let $\lambda \in [0, 1]$ denotes the training time such that the training begins at $\lambda = 0$ and finishes at $\lambda = 1$. Suppose $Q_\lambda(x)$ is the training distribution at time λ, where the training sample x is drawn. The concept of curriculum learning requires that the training samples from Q_λ should be easier than the samples from $Q_{\lambda+\delta}$, $\delta > 0$. Mathematically, curriculum learning holds the assumption that

$$Q_\lambda(x) \propto W_\lambda(x)P(x), \tag{6}$$

where $P(x)$ denotes the target training distribution including both easy and hard samples, and $0 \leq W_\lambda(x) \leq 1$ is the difficult criterion for every training time λ and sample x. In the meanwhile, the choice of $W_\lambda(x)$ should satisfy such property:

$$H(Q_\lambda(x)) < H(Q_{\lambda+\delta}(x)), \tag{7}$$

where H is the Shannon's entropy. Eq. (7) expounds the fact that the diversity and information in the training set should gradually increase with respect to the training time λ, which is in accordance with our idea to progressively exposing nodes with more complicated ST relations to the model.

To control the change of difficulties of the training set during training (the pace of curricula), we follow the theory in (Bengio et al. 2009) and introduces our curriculum scheduler $\pi(t)$, which indicates the retain rate of nodes at a certain training stage. Let $t \in \{0, 1, 2, \dots\}$ denotes the training stages, measured by gradient updates, the scheduler should smoothly level up the difficulty of training corresponding to t by gradually increasing the retain rate of nodes in a given layer. More straightforwardly, we want the model to start from a small fraction of easy nodes ($\pi(0) = \bar{\alpha}$) and gradually increases the availability of nodes to complicate the task until eventually recovering to normal training ($\lim_{t\to\infty}\pi(t) = 1$). The scheduler function $\pi(t)$ can be formulated as

$$\pi(t) = 1 - (1 - \bar{\alpha})\exp(-\beta t), \quad \beta > 0 \tag{8}$$

With the previous derivations, we can now formally introduce our ST-Curriculum Dropout strategy as, given a training stage t and nodes' high-level representation h^l, we keep the top $\pi(t)\%$ easiest nodes and dropout (set to zero) the others to control the level of difficulty of the task. The procedure can be mathematically written as:

$$S_t = \mathcal{Q}_{\pi(t)}(\{Diff(v_j) : v_j \in \mathcal{V}\}) \tag{9}$$

$$\mathcal{M} = \mathbb{1}\{Diff(v_i) \leq S_t \mid v_i \in \mathcal{V}\} \tag{10}$$

$$\widetilde{h}^{(l)} = \mathcal{M} * h^{(l)}. \tag{11}$$

Some might argue that the choice of β in Eq. (8) can be annoying since it requires cross-validation. However, as suggested in (Morerio et al. 2017), this value can actually be heuristically decided. In order to reserve all the nodes to recover normal training in later stages, we should have

$\pi(T) > 1 - \frac{1}{|\mathcal{V}|}$, where T refers to the total number of gradient updates. Surprisingly, we observe that $\beta = \frac{10^3}{T \times |\mathcal{V}|}$ naturally implies $|\pi(T) - 1| < 10^{-2}|\mathcal{V}|^{-1} \equiv \pi(T) > 1 - \frac{1}{100|\mathcal{V}|}$ and such principle can fit any situation in our framework.

So far, the nodes with difficulty level under the threshold of scheduler function $\pi(t)$ will be sampled and treated equally in model training stages. However, this could lead to a somewhat counterintuitive phenomenon: those easy nodes (e.g., $Diff(v_i) < 0.1$) are likely to be repeatedly used in model training, which deviates from the general learning principle of human. For example, college students who study calculus no longer need to review the simple addition and subtraction of numbers, but can still keep such capability when learning harder courses. From the view of optimization, we prefer at each iteration, the model can focus more on the difficult targets so as to better extend its border of capacity, while the repeatedly used easy samples are paid less attention. To this end, we introduce a simple and dynamic weighting policy into our design, so that the model can further benefit from the learning of curriculum. Formally, at training step t, we define the weight of node v_i as:

$$w_i^{(t)} = \begin{cases} 1 & \text{if} \quad Diff(v_i) \leq S_{t-1} \quad (12a) \\ 1 + \pi(t) & \text{if} \quad S_{t-1} < Diff(v_i) \leq S_t \quad (12b) \\ 0 & \text{if} \quad Diff(v_i) > S_t. \quad (12c) \end{cases}$$

We do not consider weights for those nodes which haven't been included in training (i.e., $Diff(v_i) > S_t$) and set them as 0. And our scheduler function $\pi(t)$ increases monotonically with its range in $[0, 1]$. Therefore, at each training step, the loss of newly included nodes will be mildly emphasized with more weights, while others are applied normally with $w_i^{(t)} = 1$. There are three things worth noting about the weighting design: 1) The weights of newly added nodes, which stand as the border of model's capacity at training step t, are positively correlated with the value of $\pi(t)$, since intuitively we consider, as $\pi(t)$ becomes larger, the newly added nodes can be treated as a harder capacity bound for the model to breakthrough, and therefore requires larger weights. 2) When the curriculum is over, all the nodes will be assigned the same weights (i.e., 1) to recover normal training. 3) In addition to Eq. (11) which aims to prevent the unwanted feature fusion between easy and difficult nodes (i.e., $Diff(v_i) > S_t$) in forward propagation, Eq. (12c) seeks to block such noise from model in backward propagation. Finally, we combine the weighting $w^{(t)} \in \mathbb{R}^{\mathcal{V}}$ with the objective function and formulate them as follows:

$$\mathcal{L}(\theta) = \left\| w^{(t)} \left(\mathbf{X}^{(t)}_{(n+1):(n+\beta)} - \overline{\mathbf{X}}^{(t)}_{(n+1):(n+\beta)} \right) \right\|_2^2, \quad (13)$$

where $\mathbf{X}^{(t)}_{(n+1):(n+\beta)}$ and $\overline{\mathbf{X}}^{(t)}_{(n+1):(n+\beta)}$ indicate the ground truth and model prediction at training step t, and θ is the set of model parameters.

In our curriculum design, easier nodes are first fed to the model, helping it learn the fundamental ST relations while prevent it from being disturbed by the "noisy" data in early training stages, which is the most important time for training DNNs (Jastrzebski et al. 2020). After that, more complex samples are exposed to the model, allowing it to learn the higher-level abstraction in ST relations. The newly added nodes, which serve as a manner of regularization, both improve the generalization capability of the model and allow the model to avoid over-fitting over the easy ST relations. As a result, at each training stage, the difficulty of samples can stand at the border of the model's capacity, neither too easy nor too hard, so that the model can expand its border gradually. This is very similar to the idea of Continuation Methods (Allgower and Georg 1980), which suggests to first optimize a smoothed objective and gradually consider less smoothing, with the intuition that a smooth version of the problem can reveal the global picture.

Experiment

In this section, we evaluate our proposed STC-Dropout on a wide range of spatial-temporal datasets so as to demonstrate its general applicability: METR-LA (Li et al. 2017), PEMSD7M (Yu, Yin, and Zhu 2017), Covid-19 (Panagopoulos, Nikolentzos, and Vazirgiannis 2021) and NYC-Crime (Xia et al. 2021). Since this is the first attempt to introduce CL into ST modeling task, we validate the superiority of our strategy by comparing with the strong CL strategies from other fields: 1) **C-Dropout** (curriculum dropout) (Morerio et al. 2017) , 2) **C-Smooth** (curriculum by smoothing) (Sinha, Garg, and Larochelle 2020) and 3) **Anti-CL** (Shrivastava, Gupta, and Girshick 2016) on state-of-the-art models : 1) **STGCN** (Yan, Xiong, and Lin 2018), 2) **Graph WaveNet** (Wu et al. 2019), 3) **ASTGCN** (Guo et al. 2019) and 4) **Z-GCNETs** (Chen, Segovia, and Gel 2021). The four models we choose range from the classic ones to the latest ones, and they are all designed for ST tasks. *Please refer to the appendix A.2-A.9 for more detailed descriptions of the datasets, baselines, experiment setup and extra results.*

Experimental Results

We conduct extensive experiments on a wide range of ST datasets to demonstrate the superiority of STC-Dropout, which are listed in Table 1. We can observe that generally, SOTA models still have much room for improvement via the utilization of STC-Dropout. We summarize the average improvements as follows: *MAE: 0.21 ↑, MAPE: 0.36 ↑, RMSE: 0.34 ↑* in METR-LA , *MAE: 0.47 ↑, MAPE: 0.63 ↑, RMSE: 0.61 ↑* in Covid-19, *MAE: 0.12 ↑, MAPE: 0.19 ↑, RMSE: 0.23 ↑* in PeMSD7M and *MAE: 0.06 ↑, MAPE: 0.10 ↑, RMSE: 0.08 ↑* in Crime. Especially, according to recent traffic benchmarks (Li et al. 2021; Jiang et al. 2021), our performance gains on METR-LA and PeMSD7M can even match the performance improvements between two generations of models (e.g., DCRNN and GW-Net), and we achieve this by just using a simple plug-in to change the learning strategy (An intuitive reasoning in Appendix A.7).

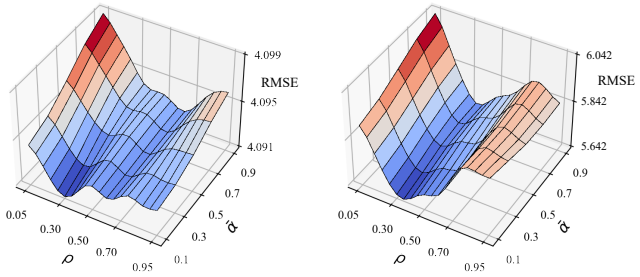
We further horizontally compare STC-Dropout with several CL methods from other fields and the results are reported in Table 2 (and Appendix A.8). It's observed that both C-Dropout and C-Smooth enhance the baseline models by a certain margin, which again validates the idea of CL on ST modeling task. However, Anti-CL fails to outperform baselines. Therefore, we here discuss some reasons behind their

Model	METR-LA (15/30/60 min)			PeMSD7M (15/30/60 min)		
	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
ST-GCN	2.89/3.50/4.62	7.64/9.59/12.70	5.75/7.28/9.40	2.12/2.72/3.35	4.99/6.92/8.97	4.09/5.53/6.62
+STC-Dropout	**2.67/3.28/4.37**	**7.37/9.22/12.22**	**5.54/6.83/9.06**	**2.03/2.61/3.27**	**4.83/6.85/8.64**	**3.85/5.30/6.38**
GW-Net	2.70/3.05/3.53	6.88/8.38/10.05	5.17/6.24/7.37	2.08/2.66/3.23	4.91/6.69/8.48	3.94/5.31/6.35
+STC-Dropout	**2.51/2.83/3.27**	**6.45/8.06/9.68**	**4.79/5.91/6.99**	**1.97/2.52/3.11**	**4.71/6.54/8.33**	**3.76/5.17/6.11**
ASTGCN	4.87/5.45/6.54	9.24/10.14/11.62	9.25/10.62/12.50	2.19/2.77/3.52	5.42/7.31/9.18	4.15/5.37/6.54
+STC-Dropout	**4.70/5.26/6.40**	**8.93/9.74/11.32**	**9.02/10.12/12.26**	**2.02/2.68/3.40**	**5.30/7.25/8.90**	**4.06/5.15/6.20**
Z-GCNETs	2.62/2.99/3.32	6.79/8.45/9.97	5.09/6.04/7.15	1.91/2.35/2.98	4.86/6.52/8.18	3.78/5.18/6.17
+STC-Dropout	**2.50/2.78/3.05**	**6.48/8.03/9.56**	**4.79/5.69/6.82**	**1.85/2.21/2.84**	**4.61/6.22/8.01**	**3.50/4.81/5.98**
Model	Covid-19 (3/5/7 days)			Crime (3/5/7 days)		
	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
ST-GCN	3.51/3.68/3.88	2.29/2.36/2.53	6.22/6.57/7.18	2.15/3.22/3.86	1.40/2.10/2.52	3.21/5.04/6.10
+STC-Dropout	**3.12/3.24/3.37**	**1.57/1.69/1.98**	**5.65/6.04/6.49**	**2.07/3.14/3.80**	**1.31/2.05/2.40**	**3.15/4.97/5.98**
GW-Net	3.40/3.55/3.76	2.26/2.30/2.37	6.06/6.42/6.92	2.17/2.80/3.38	1.42/1.83/2.21	3.48/4.60/5.69
+STC-Dropout	**2.89/3.18/3.26**	**1.49/1.65/1.81**	**5.56/5.81/6.24**	**2.11/2.77/3.33**	**1.36/1.72/2.07**	**3.42/4.55/5.61**
ASTGCN	3.54/3.60/3.86	2.31/2.33/2.52	6.23/6.45/7.15	2.31/2.96/3.39	1.78/1.99/2.28	3.45/4.46/5.29
+STC-Dropout	**2.94/3.17/3.24**	**1.60/1.77/2.02**	**5.53/5.75/6.32**	**2.24/2.91/3.32**	**1.65/1.91/2.17**	**3.35/4.37/5.21**
Z-GCNETs	3.27/3.42/3.59	2.21/2.28/2.33	5.95/6.22/6.74	2.10/2.68/3.19	1.31/1.85/2.12	3.10/4.38/5.43
+STC-Dropout	**2.82/3.07/3.12**	**1.43/1.65/1.86**	**5.54/5.69/6.18**	**2.06/2.61/3.12**	**1.22/1.78/2.04**	**3.08/4.29/5.30**

Table 1: Performances of STC-Dropout on a wide range of spatial-temporal datasets.

Model	CL Strategy	MAE	MAPE(%)	RMSE
METR-LA	ST-GCN	2.89	7.64	5.75
	+Anti-CL	3.04	7.73	5.91
	+C-Dropout	2.75	7.52	5.69
	+C-Smooth	2.77	7.60	5.72
	+STC-Dropout	**2.67**	**7.37**	**5.54**
PeMSD7M	GW-Net	2.08	4.91	3.94
	+Anti-CL	2.16	5.07	4.11
	+C-Dropout	2.03	4.91	3.92
	+C-Smooth	2.01	4.82	3.86
	+STC-Dropout	**1.97**	**4.71**	**3.76**
Covid-19	ASTGCN	3.54	2.31	6.23
	+Anti-CL	3.55	2.30	6.23
	+C-Dropout	3.23	1.83	6.06
	+C-Smooth	3.27	1.78	5.92
	+STC-Dropout	**2.94**	**1.60**	**5.53**
Crime	Z-GCNETs	2.10	1.31	3.10
	+Anti-CL	2.12	1.34	3.12
	+C-Dropout	2.09	1.25	3.09
	+C-Smooth	2.07	1.33	3.09
	+STC-Dropout	**2.06**	**1.22**	**3.08**

Table 2: Different curriculum learning strategies.



(a) PeMSD7M (b) METR-LA

Figure 3: Grid search of hyper-paramerters on ST-GCN.

performances and STC-Dropout's overwhelming improvements over others as follows:

- C-Dropout is a curriculum method proposed regarding the model itself, which progressively reduces the model's capacity in object classification tasks. Completely adopting such an idea into spatial-temporal modeling might be hard since the task is more complicated and requires a large network capacity to model high-level ST relations, especially at later training stages. The benefit of C-Dropout in experiment might come from the early stages of training, which offers better parameter initialization (Morerio et al. 2017).
- C-Smooth is first proposed in CNN using Gaussian kernel filters to "blur" the data and reduce noises. On the one hand, this might be too arbitrary without consideration of the ST relations. On the other hand, although we modify it to run on GNNs, this could arouse another problem called over-smoothing (Rong et al. 2019). Conversely, STC-Dropout is specially developed for ST graph modeling and considers the ST relation at the stage of difficulty evaluation.
- Anti-CL is another popular data selection strategy opposite to CL (i.e. hard sample first). A well-accepted insight (Chang, Learned-Miller, and McCallum 2017) about whether to choose CL or Anti-CL points out that CL is more preferable for noisy datasets to improve model robustness and Anti-CL is more beneficial for cleaner datasets to accelerate convergence. In light of this, we attribute the inferior performances of Anti-CL to the chaotic nature of ST data.

Level	Models	METR-LA		
		MAE	MAPE(%)	RMSE
Hidden	ST-GCN	2.89	7.64	5.75
	+C-LoF	2.73	7.48	5.66
	+C-IF	2.78	7.58	5.71
	+STC-Dropout	**2.67**	**7.37**	**5.54**
Input	+C-LR	2.88	7.67	5.77
	+C-LoF	2.76	7.53	5.70
	+C-IF	2.84	7.63	5.74
	+STC-Dropout	**2.69**	**7.42**	**5.61**

Table 3: Comparison with heuristic curriculum strategy

Models	METR-LA		
	MAE	MAPE(%)	RMSE
ST-GCN	2.89	7.64	5.75
+Dropout	2.87	7.63	5.74
+STC-Dropout-Mean	2.79	7.60	5.71
+SC-Dropout	2.76	7.51	5.66
+TC-Dropout	2.73	7.49	5.68
+w/o. weight	2.70	7.44	5.58
+STC-Dropout	**2.67**	**7.37**	**5.54**

Table 4: Ablation Study

Ablation Study

Comparison with Heuristic Curriculum Strategy. To validate the effectiveness of STC-Dropout, we compare it with multiple heuristic curriculum designs (without ST correlation assumptions) to forecast future 15 min's traffic speed using ST-GCN. We perform the curriculum both in the high-level hidden space (as STC-Dropout does) and directly at the input (corresponding to the discussion in the first paragraph of **Difficulty Evaluation**). Two types of heuristic curriculum schemes are used here: one defines the difficulty of nodes via model predicting loss (Guo et al. 2018) and the other seeks the complex samples through outlier detection (Zhang et al. 2019). For predicting loss we apply a basic time series prediction models: **LR** (Linear Regression) as the difficulty indicator, while for outlier detection, we use the popular corresponding algorithms: **LoF** (Local Outlier Factor) (Breunig et al. 2000) and **IF** (Isolation Forest) (Liu, Ting, and Zhou 2008). These above-mentioned methods will rank the difficulty of nodes either in the hidden dimension or at the input space, and then they follow the same schema of STC-Dropout to dynamically dropout difficult nodes at early stages. We report the results in Table 3, where we can discern that STC-Dropout, which we designed with ST relation assumptions, yields the best outcome, and the curriculum performed in hidden states is generally better than those in the input space.

Further, we consider several variants of STC-Dropout including the origin **Dropout** without curriculum (Hinton et al. 2012), **STC-Dropout-Mean**: set the dropout values to its neighbor's mean instead of zeros, **SC-Dropout**: only utilizes spatial difficulty measurer (Formula 3), and **TC-Dropout**: only uses temporal difficulty measurer (Formula 4). The weighting design is also included in the ablation. The comparison results are shown on Table 4. Regarding the variants, there is no improvement over the standard STC-Dropout on ST-GCN.

Hyper-parameter Analysis. Last but not least, we explore the hyper-parameters sensitivity to ρ, which controls the radius of the $d^{(l)}$-dimensional ball, and $\bar{\alpha}$, which specifies the initial retain rate of nodes. The results of hyper-parameter search from $\rho \times \bar{\alpha} \in [0.05, 0.10, \cdots, 0.95] \times [0.1, 0.3, \cdots 0.9]$ on ST-GCN using both PeMSD7M and METR-LA datasets are depicted in Figure 3. As we can see, the model performance is negatively related to the value of $\bar{\alpha}$. Thus, it is recommended to set its value to be relatively small to ensure only the easy samples can be retained in the early model training. Concerning ρ, we encourage maintaining $\rho \in [0.25, 0.4]$ for desirable performances.

Conclusion

This paper designs a novel curriculum strategy called ST-Curriculum Dropout, especially for spatial-temporal graph modeling. By defining the complex score of nodes based on their G-neighbor consistency (representing the difficulty of spatial relations) and data distribution density (representing the difficulty of temporal patterns), we gradually expose the complex nodes to the network to form a curriculum. Experimental results show the effectiveness of our design. One potential problem of our method could be the computation of pairwise distance. Although the cost is acceptable, this can be further alleviated by using Faiss[1] to accelerate computing. In future work, we would like to extend the STC-Dropout into further applications such as multi-variate time series.

Acknowledgments

We are grateful to anonymous reviewers for their helpful comments. We additionally would like to thank Lei Huang for his help. This work was partially supported by the grants of National Key Research and Development Program of China (No. 2018AAA0101100), National Key Research and Development Project (2021YFB1714400) of China and Guangdong Provincial Key Laboratory (2020B121201001), and Grant-in-Aid for Scientific Research B (22H03573) of Japan Society for the Promotion of Science (JSPS).

References

Allgower, E.; and Georg, K. 1980. Simplicial and continuation methods for approximating fixed points and solutions to systems of equations. *Siam review*, 22(1): 28–85.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 93–104.

[1] https://github.com/facebookresearch/faiss

Chang, H.-S.; Learned-Miller, E.; and McCallum, A. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30.

Chen, Y.; Segovia, I.; and Gel, Y. R. 2021. Z-GCNETs: time zigzags at graph convolutional networks for time series forecasting. In *International Conference on Machine Learning*, 1684–1694. PMLR.

Guo, S.; Huang, W.; Zhang, H.; Zhuang, C.; Dong, D.; Scott, M. R.; and Huang, D. 2018. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 135–150.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 922–929.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Jastrzebski, S.; Szymczak, M.; Fort, S.; Arpit, D.; Tabor, J.; Cho, K.; and Geras, K. 2020. The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*.

Jiang, R.; Yin, D.; Wang, Z.; Wang, Y.; Deng, J.; Liu, H.; Cai, Z.; Deng, J.; Song, X.; and Shibasaki, R. 2021. DL-Traff: Survey and Benchmark of Deep Learning Models for Urban Traffic Prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 4515–4525.

Kumar, M. P.; Packer, B.; and Koller, D. 2010. Self-Paced Learning for Latent Variable Models. In *NIPS*, volume 1, 2.

Kumar, M. P.; Turki, H.; Preston, D.; and Koller, D. 2011. Learning specific-class segmentation from diverse data. In *2011 International Conference on Computer Vision*, 1800–1807. IEEE.

Li, F.; Feng, J.; Yan, H.; Jin, G.; Yang, F.; Sun, F.; Jin, D.; and Li, Y. 2021. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data (TKDD)*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*, 413–422. IEEE.

Ma, F.; Meng, D.; Xie, Q.; Li, Z.; and Dong, X. 2017. Self-paced co-training. In *International Conference on Machine Learning*, 2275–2284. PMLR.

Morerio, P.; Cavazza, J.; Volpi, R.; Vidal, R.; and Murino, V. 2017. Curriculum dropout. In *Proceedings of the IEEE International Conference on Computer Vision*, 3544–3552.

Panagopoulos, G.; Nikolentzos, G.; and Vazirgiannis, M. 2021. Transfer graph neural networks for pandemic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4838–4845.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*.

Shrivastava, A.; Gupta, A.; and Girshick, R. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 761–769.

Sinha, S.; Garg, A.; and Larochelle, H. 2020. Curriculum by smoothing. *arXiv preprint arXiv:2003.01367*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021. CurGraph: Curriculum Learning for Graph Classification. In *Proceedings of the Web Conference 2021*, 1238–1248.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *The 28th International Joint Conference on Artificial Intelligence (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization.

Xia, L.; Huang, C.; Xu, Y.; Dai, P.; Bo, L.; Zhang, X.; and Chen, T. 2021. Spatial-Temporal Sequential Hypergraph Network for Crime Prediction with Dynamic Multiplex Relation Learning. In *IJCAI*, 1631–1637.

Xu, B.; Zhang, L.; Mao, Z.; Wang, Q.; Xie, H.; and Zhang, Y. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6095–6104.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*.

Yu, B.; Yin, H.; and Zhu, Z. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Zhang, M.; Li, T.; Shi, H.; Li, Y.; Hui, P.; et al. 2019. A decomposition approach for urban anomaly detection across spatiotemporal data. In *IJCAI International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence.