Adaptive Data Augmentation on Temporal Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Temporal Graph Networks (TGNs) are powerful on modeling temporal graph data based on their increased complexity. Higher complexity carries with it a higher risk of overfitting, which makes TGNs capture random noise instead of essential semantic information. To address this issue, our idea is to transform the temporal graphs using data augmentation (DA) with adaptive magnitudes, so as to effectively augment the input features and preserve the essential semantic information. Based on this idea, we present the **MeTA** (*Memory Tower Augmentation*) module: a multi-level module that processes the augmented graphs of different magnitudes on separate levels, and performs message passing across levels to provide adaptively augmented inputs for every prediction. MeTA can be flexibly applied to the training of popular TGNs to improve their effectiveness without increasing their time complexity. To complement MeTA, we propose three DA strategies to realistically model noise by modifying both the temporal and topological features. Empirical results on standard datasets show that MeTA yields significant gains for the popular TGN models on edge prediction and node classification in an efficient manner.

1 Introduction

Many real-world graphs are not static but evolving, where every edge (or interaction) has a timestamp to denote its occurrence time. These graphs are called temporal (or dynamic) graphs [33]. Recently, temporal graph networks (TGNs) [26, 33, 17] have been proposed to support learning on temporal graphs. Advanced TGNs utilize an RNN based memory module to represent a node's history as a compact state (see Fig. 1), which is used to predict the node's activities [28]. TGNs are capable of making predictions from complex graph topology and temporal information , thanks to their advanced representational power. However, the increased representational capacity comes with higher model complexity, which can induce over-fitting and weaken their generalization ability. In particular, a trained TGN may capture random noise instead of semantic information, which is not desired [35].

To combat over-fitting, data augmentation (DA) has been demonstrated to be effective [23]. Nevertheless, DA for the temporal graphs remains under-explored, of which the main challenges lie in highly irregular dynamic topology. DA applies transformations to input features so as to model realistic noise for enriching the input data. The magnitudes of the transformations, known as *DA magnitudes*, are controlled by hyper-parameters, which is positively related to the difference between the input features before and after DA [5]. Existing work on image and text data devises *adaptive DA* methods, which apply higher DA magnitudes to the less informative parts of input features, in order to effectively augment the input features while preserving the essential semantic information [32].

In order to design such an adaptive DA method on temporal graphs, we consider the informativeness of edges to a target node in terms of both time and topology when predicting the activities of the target node. On the time axis, more recent edges tend to be more informative for predicting the target node's states than the earlier ones. For example, when a girl is going to watch a movie in the evening, the interaction between her and a cinema conductor a few minutes ago is much more informative

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

to predict her behaviors than the academic discussion between her and a student in the morning. Similarly, in terms of topology, the edges directly connecting the target node provide more important information than edges a few hops away. Overall, the edges that are closer to the target node in time or topology tend to provide more important information , as validated in the existing work [26].

To devise an efficient adaptive DA method for temporal graphs, the central idea of this paper is to generate a few graphs with different DA magnitudes, and perform the message passing between these graphs to provide adaptively augmented inputs for every prediction. We encapsulate this idea in a novel module, called **MeTA** (*Memory Tower Augmentation*): MeTA stacks a few levels of memory modules as a tower with weight sharing, where lower levels process the graphs of lower DA magnitudes (see Fig. 2). MeTA includes two message passing mechanisms across levels for adaptive DA. The first one is *cross-level propagation*, which propagates the features between nodes across levels, while the second one is *memory transition*, that transmits memory states of a node from higher levels to lower ones periodically. This design allows us to achieve the goal of adaptive DA, since for every prediction, the edges closer to the target node in time or topology are placed on lower levels corresponding to lower magnitudes , as demonstrated by the theoretical analysis (see Sec. 3.4).

To complement MeTA, we propose three DA strategies augmenting both the temporal and topology features: (i) perturbing the edge time to simulate time shifts, (ii) removing edges, and (iii) adding edges to modify the topology. Our strategies effectively enrich the input data by modeling realistic noise intuitively and theoretically. MeTA is a general module that can be applied for training popular TGNs to enhance their performance. MeTA improves the effectiveness of TGNs without increasing their time complexity during training, as analyzed in Sec. 3.4. Note that our methods do not induce any extra inference cost since DA is not applied during inference.

We evaluate our MeTA on edge prediction and node classification tasks using the standard temporal graph datasets: Reddit [3], Wikipedia [20], MOOC [17], and LastFM [11]. We measure its performance through the metrics: test accuracy, average precision (AP), and the area under the ROC accuracy curve (AUC), under inductive and transductive settings. Overall, MeTA achieves substantial improvements for popular TGN models [26, 33] and enhances them to outperform the baseline methods. The source code of MeTA will be released soon.

2 Related Work

Although many methods have been proposed for representation learning on static graphs [24, 9, 15, 16, 10, 29], the work on representation learning in temporal graphs is much sparser. There exist two main classes of temporal graphs: discrete-time dynamic graphs (DTDG) [18] and continuous-time dynamic graphs (CTDG) [26]. DTDG are sequences of static graph snapshots taken at intervals in time, while CTDG can be represented as timed lists of events. Representation learning on CTDG is more flexible, general and challenging, which is the focus of this paper. Early models for temporal graph learning focus on DTDGs [12, 18, 1, 8, 34]. Some recent work addresses the learning on CTDGs [17, 28, 26, 33, 30] based on the increased model complexity. Our method is orthogonal to these work in the sense that we do not propose a new TGN model, but acts as a general DA module that can be flexibly incorporated into the training of popular TGNs to improve their effectiveness.

Data augmentation plays a central role in training neural networks, of which the effectiveness has been validated on image data [37, 27, 7]. On static graph data, [31] proposes NodeAug to augment and utilize the unlabeled data in semi-supervised learning. DropEdge [25] and GAUG [36] perturb edges as data augmentation for static graphs. Different from them, our work focuses on the DA for temporal graphs. Our MeTA considers the edge importance related to the temporal and topology information and adaptively augment temporal graphs in an efficient manner, while our DA strategies effectively augment both the temporal and topological features.

3 Methodology

In this section, we propose MeTA (*Memory Tower Augmentation*), an adaptive data augmentation (DA) approach for improving the temporal graph learning. A MeTA module can be applied to popular TGN models, and trains them with DA to improve its inference performance. MeTA includes two message passing mechanisms: *cross-level propagation* and *memory transition*, to provide adaptively augmented inputs for every prediction, as demonstrated by the theoretical analysis.

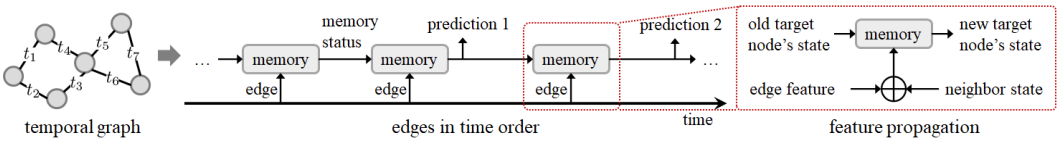

Figure 1: TGNs feed edges in time order into an RNN-based memory module, which updates nodes' states for predictions. Every time an edge connecting the node i happens, the memory module takes the i's state, edge feature, and the connected neighbor's state as the inputs, and update the state of node i (see Eq. (1)). In this way, the memory state of node i represents i's history.

To complement MeTA, we propose three DA strategies for augmenting temporal graphs by modifying the temporal and topological features, which effectively model the realistic noise to enrich the input data intuitively and theoretically. Using our methods to enhance the popular TGNs does not increase their time complexity and does not induce any inference cost.

3.1 Preliminaries

In a temporal graph $\mathcal{G}$, an interaction between nodes i and j is denoted as a temporal edge $\mathbf{e}_{ij}(t), t > 0$, where $\mathbf{e}$ is the edge attributes [26], and t is the edge time. $t = 0$ is the time of starting observations.

Advanced TGN models utilize a memory module to store and update the (memory) state $\mathbf{s}_i(t)$ of node i (see Fig. 1). The state of node i is expected to represent i's history in a compressed format. Given the memory updater as mem, when an edge $\mathbf{e}_{ij}(t)$ connecting node i is observed, node i's state is updated as:

$$\mathbf{s}_i(t) = \mathsf{mem}\big(\mathbf{s}_i(t^-), \mathbf{e}_{ij}(t) \| \mathbf{s}_j(t^-)\big), \tag{1}$$

where $\mathbf{s}_i(t^-)$ is the memory state of node i just before time t, $\|$ is the concatenation operator, and node j is i's neighbor connected by $\mathbf{e}_{ij}(t)$. $\mathbf{s}_i(t)$ is initialized to 0 values and not changed until the next edge involving node i happens. Edge feature $\mathbf{e}_{ij}(t)$ and neighbor state $\mathbf{s}_j(t^-)$ are concatenated as the input to update $\mathbf{s}_i(t)$. All edges are fed to the memory modules in time order (see Fig. 1).

The implementation of mem varies slightly for different TGN models, of which a common component is a recurrent neural network (RNN) such as LSTM [13] or GRU [4]. TGNs make the prediction for node i at time t using $\mathbf{s}_i(t^-)$, instead of $\mathbf{s}_i(t)$, since when making predictions for time t, the interactions at time t have not happened and cannot be observed.

The number of hops is used to denote the distance on the topology [22]. From Eq. (1), we observe that the edges connecting node i directly contributes to i's state, while the edges, that are multiple hops away from i, have their features propagated to node i through i's neighbors' states.

3.2 Memory Tower for Adaptive Data Augmentation on Temporal Graphs

Data augmentation (DA) aims at creating novel training data by applying transformation to input features [32]. The magnitudes of the transformations, known as *DA magnitudes*, are controlled by the hyper-parameters, which is positively related to the difference between the input features before and after DA [5]. To effectively augment the input features while preserving the essential semantic information, *adaptive DA* methods apply low DA magnitudes to the informative parts of the input features and higher magnitudes to the less informative parts [32]. For example, TF-IDF based word replacement, a DA method for natural language processing, is designed to remove less important keywords with higher probability [32]. This helps the model to generalize better to changes in the uninformative features, and effectively encode the essential features into the representations.

In a temporal graph, when predicting the activities of a target node, different edges are of different informativeness for the target node. For example, an engineer is busy doing a project in the evening, the interaction between him and his colleague a few minutes ago is much more important for predicting his working status than the discussion between him and his daughter on a new movie in the morning. Similarly, the edges directly connecting the target node tend to be more informative than the ones that are multiple hops distant since the former directly happen on the target node. Overall,

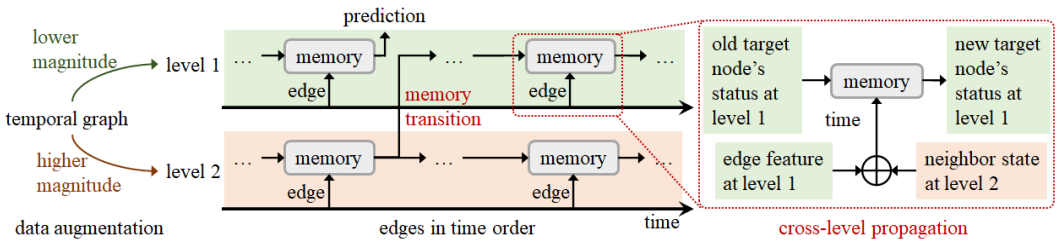

Figure 2: Our MeTA method with two levels of memory modules. The colors denote the DA magnitudes, i.e., 'green' means lower, and 'yellow' means higher. With the memory transition from level 2 to level 1, for every prediction, the earlier edges hold higher DA magnitudes. By propagating the neighbors' states from higher levels, more distant edges on topology hold higher DA magnitudes.

if an edge is more close to the target node in terms of time or topology, it is more informative, as validated in [26].

Applying adaptive DA magnitudes to edges of different informativeness is not trivial, since an edge's informativeness changes with different predictions, i.e., when making predictions for different target nodes, the distance from an edge to the target node changes. A simple method for adaptive DA is to separately augment the graph with adaptive magnitudes for every prediction, which leads to different input graphs for different predictions. Therefore, this method repeats feeding the graph edges to the model from the beginning time for every prediction, and is named the *refeed* method.

Existing work [26, 28, 17] trains the memory module in a recursive manner for efficiency, i.e., the predictions at time t take the former output at time τ ($\forall \tau < t$) as inputs, as shown in Fig. 1, since there is no DA and different predictions share the same input graph. The refeed method is much less efficient than recursive training, since in the former one, the memory states before the prediction time have to be recomputed for every prediction, while the number of predictions is generally the same as the number of edges during training, which is a large value in practice [17].

We propose the Memory Tower Augmentation (MeTA), which offers appropriate DA magnitudes for every prediction in an efficient manner. MeTA stacks L levels of memory modules as a tower with weight sharing. For level l ($l = 1, \ldots, L$), we augment the graph $\mathcal{G}$ as $\mathcal{G}^{(l)}$ with the magnitude p_l. p_l increases as l increases, so that the DA magnitude is higher at a higher level. Denote the state of node i at time t on level l as $\mathbf{s}^{(l)}(t)$, and the edge connecting nodes i and j at time t on level l, i.e., belonging to $\mathcal{G}^{(l)}$, as $\mathbf{e}_{ij}^{(l)}(t)$. We feed edges at all levels in time order to the memory module, and always make the prediction using the memory states at the first level, i.e., $\mathbf{s}_i^{(1)}(t)$.

MeTA includes two message passing mechanisms across levels to offer adaptive DA for every prediction: *cross-level propagation* and *memory transition*. With *cross-level propagation*, when observing an edge $\mathbf{e}_{ij}^{(l)}(t)$ involving node i at level l, node i's state at level l is updated as:

$$\mathbf{s}_i^{(l)}(t) = \mathsf{mem}\big(\mathbf{s}_i^{(l)}(t^-), \mathbf{e}_{ij}^{(l)}(t) \| \mathbf{s}_j^{(\min(l+1,L))}(t^-)\big), \tag{2}$$

In this way, the edges that are more distant to the target node i in terms of the topology are augmented at a higher magnitude, since their features are propagated to the target node through the neighbors' states $\mathbf{s}_j^{(\min(l+1,L))}(t^-)$ that are located on higher levels (see Proposition 1).

In addition to the distances on the topology, the time distances also influence the edge's informativeness. Hence, the second message passing mechanism we design is the *memory transition* for nodes' states from higher levels to lower ones, i.e., replacing the level-l states with the level-$l+1$ ones:

$$\mathbf{s}_i^{(l)}(t) \leftarrow \mathbf{s}_i^{(l+1)}(t), \forall l = 1, \ldots, L-1. \tag{3}$$

The transition is made in the memory tower periodically, so that when we make a prediction at level 1, the more recent edges contributing to the prediction are at lower levels corresponding to lower DA magnitudes (see Proposition 2). We determine the transition time of node i by implementing a counter for node i, which is initialized to 0 and increments when the model makes a prediction related to node i. If the counter reaches T, we reinitialize the counter and perform this memory transition for node i, where T is a hyper-parameter for the transition period.

Overall, we visualize our MeTA method in Fig. 2 and summarize it in Alg. 1 (see Appendix). How to obtain the predictions from memory states and calculate the training loss differ slightly for the specific TGN models and downstream tasks, of which the details can be found in the corresponding work [17, 28, 26].

3.3 Data Augmentation Strategies

We propose three DA strategies for the temporal graphs: (i) perturbing time; (ii) removing edges; (iii) adding edges with perturbed time. The first one augments the temporal features; the second is for the graph topology, while the last augments both the topology and temporal features. The search space for the hyper-parameters of DA magnitudes is large if we set the parameters separately for different DA techniques. Hence, for the convenience of setting the hyper-parameters, we use a unified hyper-parameter p to control the magnitude of all the three DA strategies. A higher value of p implies a larger magnitude. This design is inspired by the method RandAugment for augmenting images [5], where only one parameter is used to control the magnitude of more than ten image transformations without the loss of effectiveness. Next, we introduce our DA strategies in detail.

Perturb Time Each edge has a timestamp to denote its happening time. In practice, these timestamps may not be precise. For instance, a courier can report delivery time earlier or later than the actual time. These time shifts are widespread in the real world but do not necessarily affect the semantic information. Hence, we perform time perturbation to model this kind of noise as data augmentation. Denote the average intervals between interactions as $\bar{t}$, and the time range of training data as $[0, T_{\max}]$. We add Gaussion noise following $\tau \sim \mathcal{N}(0, \sigma^2)$ [2] to the edge time t as $t_{\text{new}} = \tau + t$, where t_{new} is the augmented timestamp. We set the standard deviation of Gaussion noise σ as $p\bar{t}$. When p is 0, all edges' time is not changed, and the noise's standard deviation grows as p increases. The augmented time t_{new} can exceed the time range $[0, T_{\max}]$. This is not practical since only the observations in the range $[0, T_{\max}]$ can be observed. Therefore, we set $t_{\text{new}} = t$ if $\tau + t > t_{\max}$ or $\tau + t < 0$, so that all the augmented time t_{new} all in the valid range $[0, T_{\max}]$. This design also helps us to retain the original distribution of edges' happening time after DA (see Theorem 1).

Remove Edges Removing edges models the realistic noise that some edges may be missing in the data collection. For example, a researcher can communicate with hundreds of other researchers one year. Collection of these interactions may have omissions, and it is unnecessary to consider all interactions to predict the researcher's state. A satisfactory TGN model is desired to effectively encode the essential semantic information even in the presence of this noise. We set the probability of removing an edge as p. With a higher p, edges are more likely to be removed and the augmented graphs tend to be more incomplete.

Add Edges with Perturbed Time A node can repeat interacting with its neighbor. An engineer, for example, repeats communicating with his colleague to check the project progress, and he talks with a waiter several times to order more food during dinner. We add the edges with the perturbed time to model noise from interaction repetition. Denote the edge set of the original graph $\mathcal{G}$ as $\mathcal{E}$. We uniformly sample the edges from $\mathcal{E}$ with replacement with the budget $p\#\mathcal{E}$ to form $\hat{\mathcal{E}}$ and then perturb the time of edges in $\hat{\mathcal{E}}$ as introduced earlier , where $\#\mathcal{E}$ is the cardinality of $\mathcal{E}$. We add the edges in $\hat{\mathcal{E}}$ for DA. Higher p corresponds to more edges being added.

3.4 Analysis and Discussion

We analyze how our MeTA and DA techniques work for temporal graph learning. Recall that if an edge is more distant to a target node on either topology or time, it is less important or informative to predict the target node's properties and thus we are expected to apply a higher DA magnitude to it. Next, we provide theoretical analysis to show that our methods meet this expectation. The proof is provided in Appendix.

Proposition 1 (Effect of topology)**.** *If edge $\mathbf{e}_{km}^{(l)}(\tau)$ contributes to the node state $\mathbf{s}_i^{(1)}(t)$, and the distances from nodes k and m to i are equal to or larger than h hops in graph $\mathcal{G}$, we have the level of edge $\mathbf{e}_{km}^{(l)}(\tau)$: $l \geq \min(h+1, L)$ and the time $\tau < t$, where L is the number of levels.*

Proposition 1 implies that the edges more distant to the target node on the topology must be on levels of higher l, and thus of higher DA magnitudes. In addition to the topology, we provide the following proposition to illustrate the effects of the temporal distances.

Proposition 2 (Effect of time)**.** *If edges $\mathbf{e}_{ij}^{(l_1)}(\tau_1)$ and $\mathbf{e}_{ik}^{(l_2)}(\tau_2)$ connect node i and directly contribute to the node state $\mathbf{s}^i(t)$ as Eq.* (2)*, we have $l_1 \geq l_2, \tau_1 < \tau_2 < t$ when $\tau_1 < \tau_2$ holds, and $l_1 \leq l_2, \tau_2 < \tau_1 < t$ holds with $\tau_1 > \tau_2$.*

Proposition 2 shows that the more earlier edges tend to be on levels of higher l, and thus of higher DA magnitudes.

Next, we analyze how our DA strategies retain the original temporal graph characteristics to model realistic noise. Many social and natural activities follow the Poisson process [19, 14, 6]. Denote the time range of the training data as $[0, T_{\max}]$. Our data augmentation includes the perturbations on the edge time. If nodes' interactions follows a Poisson process, we provide the following theorem to show that our DA techniques do not change the distribution of edge time.

Theorem 1. *If the edges before data augmentation follow a homogeneous Poisson process, our data augmentation does not change the distribution of any edge's occurrence time.*

Theorem 1 illustrates the benefits of our strategies for time perturbation. Otherwise, if the original distribution is broken, the augmented edges cannot reflect the realistic condition and may degrade the TGNs' generalization. In addition to the time distribution, we provide the following proposition to show the expected edge number does not change after our DA. Denote the edge set of the original graph $\mathcal{G}$ as $\mathcal{E}$ and that of the augmented graph $\mathcal{G}^{(l)}$ as $\mathcal{E}^{(l)}$.

Proposition 3. *After our data augmentation, the expected edge number is the same as that before data augmentation for any data augmentation magnitude, i.e., $\mathbb{E}[\#\mathcal{E}^{(l)}] = \#\mathcal{E}, \forall p_l \in [0, 1]$ holds.*

This proposition shows that our data augmentation does not change the density of interactions in expectation, which meets the realistic condition and does not induce extra computation load.

We analyze the time complexity of training TGNs and MeTA with respect to the input data. Denote the number of interactions as $\#\mathcal{E}$, the node number $\#\mathcal{V}$, and the dimensionality of the input features as d. Since the RNN based memory module updates memory states with every edge and make predictions in a recursive manner, the time complexity of the memory module is $\mathcal{O}(d \cdot \#\mathcal{E})$. The refeed method introduced in Sec. 3.1 achieves adaptive DA by feeding the input data from the beginning time to the prediction time into the memory module for every prediction. The corresponding time complexity is:

$$\mathcal{O}(\sum_{e=0}^{\#\mathcal{E}-1} de) = \mathcal{O}\big(d((\#\mathcal{E})^2 - \#\mathcal{E})/2\big) = \mathcal{O}\big(d(\#\mathcal{E})^2\big), \tag{4}$$

which is much higher than the complexity of original TGNs.

Our MeTA method stacks L levels of memory modules. On level l, the time complexity is $\mathcal{O}(d \cdot \#\mathcal{E}^{(l)})$, where $\mathcal{E}^{(l)}$ is the edge set of the augmented graph $\mathcal{G}^{(l)}$ on level l. Summing the complexity of all levels up leads to $\sum_{l=1}^{L} d \cdot \#\mathcal{E}^{(l)}$, of which the expectation is

$$\mathcal{O}\Big(\mathbb{E}\big[\sum_{l=1}^{L} d \cdot \#\mathcal{E}^{(l)}\big]\Big) = \mathcal{O}\Big(\sum_{l=1}^{L} d \cdot \mathbb{E}[\#\mathcal{E}^{(l)}]\Big) = \mathcal{O}\Big(\sum_{l=1}^{L} d \cdot \#\mathcal{E}\Big) = \mathcal{O}(d\#\mathcal{E}), \tag{5}$$

which is same as the original TGNs. The second equation holds because of Proposition 3, while the last equation holds because L is a small constant value invariant to the input data, and we find that $L = 2$ generally exhibits satisfactory performance. Compared to the refeed method, our MeTA achieves the adaptive DA with much lower complexity during training. During inference, our MeTA method does not lead to any extra computation load since DA is not applied then.

4 Experiments

In this section, we present the performance of TGN models implemented with our adaptive DA method. We compare our method against a variety of strong baselines on the task of edge prediction and node classification on temporal graphs. Our experimental settings closely follow those of the previous work [26, 33, 30] to ensure fair comparison. To avoid violating temporal constraints, we make predictions that strictly take place after all observations. As for the evaluation metrics, we follow [33, 30] to use average precision (AP) and accuracy in the edge prediction tasks and employ the area under the ROC accuracy curve (AUC) for node classification [17].

Table 1: Statistics of the datasets used in our experiments. Edges refer to temporal edges.

Dataset	#Nodes	#Edges	#Nodes in val./test.	#Edges in val./test.	#Nodes with dynamic labels	Nodes' label type
MOOC	7,144	411,749	2,599/2,412	61,762/61,763	4,066	course dropout
LastFM	1,980	1,293,103	1,714/1,753	193,965/193,966	-	-
Reddit	10,984	672,447	9,839/9,615	100,867/100,867	366	posting ban
Wikipedia	9,227	157,474	3,256/3,564	23,621/23,621	217	editing ban

Table 2: Test accuracy and average precision (AP) of transductive edge prediction. We conduct 100 trials with random weight initialization. Mean (%) and standard deviations are reported. The best results in each column are highlighted in **bold** font.

Method	MOOC		LastFM		Reddit		Wikipedia	
	Accuracy	AP	Accuracy	AP	Accuracy	AP	Accuracy	AP
CTDNE [21]	65.34 ± 0.7	74.29 ± 0.6	59.43 ± 0.5	68.29 ± 0.4	73.76 ± 0.5	91.41 ± 0.3	79.42 ± 0.4	92.17 ± 0.5
JODIE [17]	76.45 ± 0.6	83.87 ± 0.4	63.58 ± 0.6	72.04 ± 0.5	90.91 ± 0.3	97.11 ± 0.3	87.04 ± 0.4	94.62 ± 0.5
TGAT [33]	75.20 ± 0.5	82.66 ± 0.4	65.76 ± 0.5	72.97 ± 0.5	92.92 ± 0.3	98.12 ± 0.2	88.14 ± 0.2	95.34 ± 0.1
DyRep [28]	73.36 ± 0.4	81.75 ± 0.3	71.16 ± 0.3	79.82 ± 0.3	92.11 ± 0.2	97.98 ± 0.1	87.77 ± 0.2	94.59 ± 0.2
TGN [26]	81.38 ± 0.6	89.79 ± 0.5	69.35 ± 0.2	78.41 ± 0.2	92.56 ± 0.2	98.70 ± 0.1	89.51 ± 0.4	98.46 ± 0.1
TGN [26] + GAUG [36]	81.50 ± 0.5	89.84 ± 0.4	69.64 ± 0.3	78.59 ± 0.3	92.60 ± 0.2	98.71 ± 0.2	89.54 ± 0.3	98.48 ± 0.1
DyRep + MeTA (Ours)	76.21 ± 0.4	84.18 ± 0.3	$\mathbf{74.29 \pm 0.3}$	$\mathbf{82.15 \pm 0.2}$	93.04 ± 0.3	98.62 ± 0.1	88.92 ± 0.2	95.63 ± 0.2
TGN + MeTA (Ours)	$\mathbf{83.84 \pm 0.5}$	$\mathbf{92.03 \pm 0.3}$	72.15 ± 0.2	80.89 ± 0.2	$\mathbf{94.19 \pm 0.2}$	$\mathbf{99.08 \pm 0.1}$	$\mathbf{91.34 \pm 0.3}$	$\mathbf{98.87 \pm 0.1}$

Table 3: Test accuracy and average precision (AP) of inductive edge prediction. We conduct 100 trials with random weight initialization. Mean (%) and standard deviations are reported. The best results in each column are highlighted in **bold** font.

Method	MOOC		LastFM		Reddit		Wikipedia	
	Accuracy	AP	Accuracy	AP	Accuracy	AP	Accuracy	AP
JODIE [17]	75.79 ± 0.5	83.44 ± 0.6	57.51 ± 0.6	65.24 ± 0.8	88.34 ± 0.9	94.36 ± 1.1	84.32 ± 0.4	93.11 ± 0.4
TGAT [33]	74.02 ± 0.3	80.84 ± 0.5	59.67 ± 0.4	66.58 ± 0.5	90.73 ± 0.2	96.62 ± 0.3	85.35 ± 0.2	93.99 ± 0.3
DyRep [28]	72.92 ± 0.4	80.36 ± 0.4	63.96 ± 0.5	72.12 ± 0.4	89.60 ± 0.2	95.68 ± 0.2	83.46 ± 0.3	92.05 ± 0.3
TGN [26]	80.73 ± 0.2	89.21 ± 0.3	64.52 ± 0.3	72.40 ± 0.2	91.62 ± 0.1	97.55 ± 0.1	88.60 ± 0.2	97.81 ± 0.1
TGN [26] + GAUG [36]	80.82 ± 0.3	89.27 ± 0.3	64.58 ± 0.3	72.44 ± 0.2	91.64 ± 0.2	97.56 ± 0.1	88.63 ± 0.2	97.82 ± 0.2
DyRep + MeTA (Ours)	75.89 ± 0.4	82.56 ± 0.3	65.94 ± 0.3	73.61 ± 0.3	90.52 ± 0.2	96.59 ± 0.2	85.67 ± 0.3	94.13 ± 0.2
TGN + MeTA (Ours)	$\mathbf{83.47 \pm 0.2}$	$\mathbf{90.85 \pm 0.2}$	$\mathbf{67.22 \pm 0.3}$	$\mathbf{74.19 \pm 0.2}$	$\mathbf{92.96 \pm 0.1}$	$\mathbf{98.17 \pm 0.1}$	$\mathbf{90.82 \pm 0.2}$	$\mathbf{98.26 \pm 0.1}$

We use four standard temporal graph datasets: MOOC [17], LastFM [11], Reddit [3], and Wikipedia [17] for evaluation. The statistics of these datasets are shown in Table 1. We use a chronological train-validation-test split with a ratio of 70%-15%-15% following [26, 33].

For the hyper-parameters of baseline methods, e.g., the number of hidden units, the optimizer, the number of sampled neighbors, and the learning rate, we set them as suggested by their authors. For the hyper-parameters of our MeTA, we set the number of memory levels as $L = 2$, the data augmentation magnitudes as $p_1 = 0.1, p_2 = 0.8$, and the memory transition period as $T = 10$, by default. Note that these settings are fixed for all experiments unless specifically indicated to be changed. Setting more levels with $L > 2$ offers higher granularity for DA magnitudes, but we find that $L = 2$ generally offers satisfactory performance.

4.1 Edge Prediction

Following [33, 26], we conduct experiments for edge prediction under both transductive and inductive settings for a comprehensive evaluation. In the former one, we predict edges connecting the nodes observed during training, and in the latter we predict edges for the unseen nodes [26].

For transductive edge prediction , we take the state-of-the-art approaches for representation learning on temporal graphs: CTDNE [21], JODIE [17], DyRep [28], TGAT [33], and TGN [26], and the advanced DA method for static graphs GAUG [36] as the baselines for comparison [33]. We conduct the experiments for 100 trials with random weight initialization. We implement MeTA with the TGN models DyRep and TGN. Table 2 reports the results. Our MeTA improves the test accuracy of DyRep by 3.9% on MOOC, 4.4% on LastFM, 1.0% on Reddit, 1.3% on Wikipedia, and TGN by 3.0% on MOOC, 4.0% on LastFM, 1.8% on Reddit, 2.0% on Wikipedia. As a result, our MeTA achieves substantial improvements for DyRep and TGN, which is higher than that given by the advanced DA method for static graphs GAUG [36].

In the inductive setting, we keep the baselines which support inductive learning for comparison. We conduct the experiments for 100 trials with random weight initialization. The results are reported in Table 3. We implement our MeTA with DyRep and TGN to study whether MeTA can improve the performance of TGNs under the inductive setting. We observe that MeTA improves the test accuracy of DyRep by 4.1% on MOOC, 3.1% on LastFM, 1.0% on Reddit, 2.6% on Wikipedia, and TGN by 3.4% on MOOC, 4.2% on LastFM, 1.5% on Reddit, 2.5% on Wikipedia. As a result, our MeTA method enhances DyRep and TGN to outperform the baseline methods in the inductive task.

Given the TGN models DyRep and TGN, our MeTA achieves consistent and substantial improvements on all datasets, thanks to the adaptive data augmentation offered by our MeTA method. MeTA performs better than GAUG [36], since the latter does not consider the time features in temporal graphs for DA and thus cannot provide augmented inputs adaptive to the time and topology as our MeTA. MeTA effectively augments the input features while preserving the essential semantic information. Overall, the results above indicate that our methods are effective in improving the effectiveness of the popular TGN models in both transductive and inductive settings.

4.2 Node Classification

The task of node classification on temporal graphs is to predict the time-varying labels of nodes [17]. Table 4 reports the test ROC AUC. The LastFM dataset is not used here since it does not have node labels. We observe that MeTA improves ROC AUC of DyRep by 2.1% on MOOC, 2.3% on Reddit, 2.4% on Wikipedia, and TGN by 2.2% on MOOC, 2.0% on Reddit, 2.5% on Wikipedia. As a result, our MeTA method enhances DyRep and TGN to outperform the baseline methods in the temporal node classification. This validates the importance of adaptive data augmentation and our DA strategies on enriching the input data for temporal graph learning. Ou MeTA adaptively augments the input features with appropriate magnitudes, and thus effectively enhance TGNs on node classification.

Table 4: ROC AUC (%) on the test set for temporal node classification. We conduct 100 trials with random weight initialization. Mean (%) and standard deviations are reported.

Method	MOOC	Reddit	Wikipedia
CTDNE [21]	67.54 ± 0.7	59.43 ± 0.6	75.89 ± 0.5
JODIE [17]	76.31 ± 1.6	61.83 ± 2.7	84.84 ± 1.2
TGAT [33]	74.25 ± 0.9	65.56 ± 0.7	83.69 ± 0.7
DyRep [28]	75.32 ± 1.3	62.91 ± 2.4	84.59 ± 2.2
TGN [26]	77.73 ± 0.7	67.06 ± 0.9	87.81 ± 0.3
TGN [26] + GAUG [36]	77.82 ± 0.8	67.09 ± 0.9	87.85 ± 0.5
DyRep + MeTA (Ours)	76.88 ± 1.1	64.36 ± 2.0	86.65 ± 1.9
TGN + MeTA (Ours)	$\mathbf{79.41 \pm 0.8}$	$\mathbf{68.37 \pm 0.9}$	$\mathbf{90.03 \pm 0.3}$

4.3 Efficiency with MeTA

As analyzed in Sec. 3.4 , MeTA enhance TGNs without increasing their time complexity. We evaluate the efficiency of TGN with MeTA using the Reddit dataset, which holds the largest number of nodes and second largest number of edges among the used datasets. We follow the experimental setting of the transductive edge prediction in Sec. 4.1.

The methods we evaluate include: TGN, our MeTA method of the level number $L = 1, 2$, and the refeed method to achieve adaptive DA introduced in Sec. 3.2. The number of memory updates (# Memory Updates) processed per epoch by different methods and the performance is reported in Table 5, where 'Time' is the training time until convergence using a Linux Server with an Intel(R) Xeon(R) E5-1650 v4 @ 3.60GHz CPU and a GeForce GTX 1080 Ti GPU. We notice that, compared with the refeed method, our MeTA significantly reduces the number of memory updates per epoch. As a result, our MeTA method takes much less training time and memory updates than the refeed method, and exhibits similar efficiency to the original TGNs without DA, while the running time and test accuracy of refeed are unavailable (denoted as N.A.), since it takes too much time to converge. This agrees with the theoretical results in Sec. 3.4, since TGNs' time complexity is $\mathcal{O}(d \cdot \#\mathcal{E})$, which is the same as that of TGN with MeTA, while the time complexity of refeed is $\mathcal{O}(d(\#\mathcal{E})^2)$, much higher than our MeTA. Here, $\#\mathcal{E}$ is the number of edges, which is generally a large value in practice. Both MeTA ($L = 1$) and MeTA ($L = 2$) lead to substantial improvements over TGNs.

Table 5: Training time of TGN and TGN with DA on the Reddit dataset. The task is transductive edge prediction. We denote the refeed method for DA (see Sec. 3.2) as refeed. L is the level number of our MeTA method. We report mean values over 100 trials with random weight initialization.

Method	**# Memory Updates**	**Time**	**Accuracy (%)**
TGN [26]	4.7×10^5	4625s	92.56
TGN + refeed	1.1×10^{11}	N.A.	N.A.
TGN + MeTA ($L = 1$)	4.7×10^5	**4513s**	93.07
TGN + MeTA ($L = 2$)	9.4×10^5	7241s	**94.19**

4.4 Ablation Study

We investigate the effects of the adaptive DA applied by our MeTA module. With and without MeTA, we implement the DyRep model with our DA strategies. We follow the experimental settings of transductive edge prediction introduced in Sec. 4.1. Fig. 3 presents the results on the LastFM dataset. Our DA techniques without MeTA enhances DyRep to perform better than the original DyRep without DA, since it enriches the input data with diverse DA strategies. Note that with extremely large DA magnitudes the performance degrades, since the essential semantic information of some nodes at some time may not be retained. Furthermore, our MeTA leads to more significant improvements than only the DA techniques without MeTA, which validates the importance of adaptive DA achieved by MeTA. Our MeTA applies higher DA magnitudes to less important features, so as to effectively augment the input features and preserve the essential semantic information.

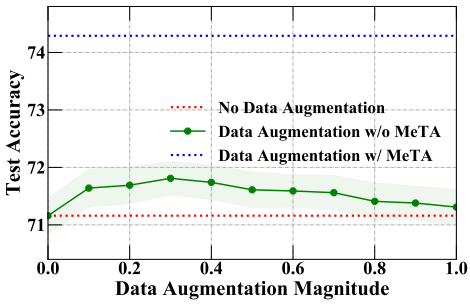

Figure 3: The test accuracy of DyRep on the LastFM dataset enhanced by our DA strategies with and without our MeTA method.

Next, we evaluate the contributions of our data augmentation strategies. We apply different DA strategies sequentially with our MeTA on the DyRep model. The results are presented in Table. 6. Our data augmentation strategies improve the effectiveness of DyRep by modeling the realistic noise to enrich the training data. Among different DA strategies, adding edges with time perturbation leads the largest benefits, since it augments both the temporal and topology features. Perturbing time gives higher improvements than removing edges, because augmenting the temporal information meets the characteristics of the temporal graph data better.

Table 6: Effects of our different DA strategies on the inductive edge prediction of MOOC.

Technique	Accuracy (%)	Δ	Cumu Δ
DyRep	72.92	0	0
+ Remove Edges	73.64	+0.72	+0.72
+ Perturb Time	74.68	+1.04	+1.76
+ Add Edges	**75.89**	**+1.21**	**+2.97**

Finally, we analyze the sensitivity of MeTA to the hyper-parameter: p_1, p_2 to control the DA magnitudes. The result is visualized in Fig. 4. We alter p_1 among $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and p_2 among $\{0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. The performance of TGN with MeTA is relatively smooth when parameters are within certain ranges. However, extremely large values of p_1 and small p_2 results in poor performances. Too large DA magnitudes on level 1 with large p_1 increase the risks of breaking the essential semantic information, while too small magnitudes on level 2 with small p_2 cannot augment the input features sufficiently. Moreover, only a poorly set hyper-parameter does not lead to significant performance degradation, which demonstrates that our MeTA framework is able to provide appropriate DA magnitudes for different parts of input features. We provide additional experiments about the message passing mechanisms and the hyper-parameter T to control the memory transitions in Appendix.

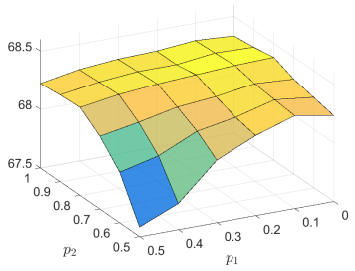

Figure 4: The ROC AUC (% in z-axis) of TGN with MeTA on node classification of Reddit with different hyper-parameters p_1 and p_2.

5 Conclusion

In this paper, we study the problem of exploring data augmentation to strengthen the TGN models. We propose a novel methodology named MeTA, which provides adaptively augmented inputs for temporal graph learning. MeTA is a generic framework that can use any DA strategies to enhance popular TGN models. In our work, considering the characteristics of temporal graph data, we propose three DA strategies to augment the temporal graphs by modifying both temporal and topology features. Our experimental results show that MeTA, with our DA strategies, yields significant gains for both edge prediction and node classification in an efficient manner. One limitation of our methods is that our DA strategies cannot model all realistic noise on temporal graphs in practice. Therefore, future work could explore other advanced DA strategies to enhance our approach.

References

[1] Nahla Mohamed Ahmed, Ling Chen, Yulong Wang, Bin Li, Yun Li, and Wei Liu. Sampling-based algorithm for link prediction in temporal networks. *Information Sciences*, 374:1–14, 2016.

[2] Mohammad Ahsanullah, BM Golam Kibria, and Mohammad Shakil. Normal distribution. In *Normal and Student´st Distributions and Their Applications*, pages 7–50. Springer, 2014.

[3] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 830–839, 2020.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[6] Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

[7] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[8] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5 (2):1–27, 2011.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[10] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[11] Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer, 2012.

[12] Ryohei Hisano. Semi-supervised graph embedding approach to dynamic link prediction. In *International Workshop on Complex Networks*, pages 109–121. Springer, 2018.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[14] Tomoharu Iwata, Amar Shah, and Zoubin Ghahramani. Discovering latent influence in online social activities via shared cascade poisson processes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–274, 2013.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[17] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019.

[18] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[19] Tahmida Mahmud, Mahmudul Hasan, Anirban Chakraborty, and Amit K Roy-Chowdhury. A poisson process model for activity forecasting. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3339–3343. IEEE, 2016.

[20] Meta. Data dumps — meta, discussion about wikimedia projects, 2021. URL `https://meta.wikimedia.org/w/index.php?title=Data_dumps&oldid=20922289`. [Online; accessed 1-February-2021].

[21] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018.

[22] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020.

[23] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[25] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

[26] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[27] Krishna Kumar Singh, Hao Yu, Aron Sarmasi, Gautam Pradeep, and Yong Jae Lee. Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond. *arXiv preprint arXiv:1811.02545*, 2018.

[28] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[30] Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. Apan: Asynchronous propagate attention network for real-time temporal graph embedding. *arXiv preprint arXiv:2011.11545*, 2020.

[31] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–217, 2020.

[32] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[33] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

[34] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Haifeng Chen, and Wei Wang. Link prediction with spatial and temporal consistency in dynamic networks. In *IJCAI*, pages 3343–3349, 2017.

[35] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[36] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. *arXiv preprint arXiv:2006.06830*, 2020.

[37] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]