TG-GEN: A DEEP GENERATIVE MODEL FRAMEWORK FOR TEMPORAL GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have recently emerged as popular methods for learning representations of non-euclidean data often encountered in diverse areas ranging from chemistry and biology to social and financial networks. More recently, research has focused specifically on learning on *temporal graphs*, wherein the nodes and edges of a graph, and their respective features, may change over time. However, existing work in the temporal graph space has largely focused on *discriminative* models. In this work, we present TG-Gen, a generic *generative* framework for temporal graph data, which combines an encoder module that creates temporal embeddings of nodes from raw interaction data, with a decoder module that uses the learned temporal embeddings to create a deep probabilistic model of interaction data. We show that TG-Gen is able to generate robust and accurate synthetic data for temporal graphs for two traditional benchmark data and a novel dataset. Additionally, we demonstrate that TG-Gen is able to learn generalizable representations of temporal graphs and outperforms the previous state-of-the-art method in the *discriminative* regime, such as for dynamic link prediction. Finally, we perform comprehensive ablation studies which show the effects of specific modules and configurations of our model.

1 INTRODUCTION

Graph representation learning has been an area of significant research interest in the past several years. Specifically, graph neural networks (GNNs), a generalization of other neural network architectures typically based on the message passing mechanism (Hamilton et al., 2017a; Battaglia et al., 2018), have demonstrated to be powerful tools that can learn robust representations of graphs, and other graph-like data structures, such as point clouds, natural networks, and manifolds(Wu et al., 2020a;b). GNNs have proven to significantly improve downstream performance in diverse applications ranging from particle physics (Shlomi et al., 2020; Ju et al., 2020; Duarte & Vlimant, 2022) and drug discovery (Bongini et al., 2021; Jiang et al., 2021), to recommender systems (Fan et al., 2019; Yin et al., 2019) and financial fraud detectionXu et al. (2021); Cheng et al. (2020); Dou et al. (2020); You et al. (2022). Specifically, by learning useful embedding of nodes in graph structured data, downstream tasks such as node classification and regression, edge prediction, and graph classification and regression is possible. However, the majority of early efforts in graph representation learning focused on static graphs(Wu et al., 2020b). Only more recently has research extended to learning on *dynamic* graphs, which change over time(Barros et al., 2021; Kazemi et al., 2020).

Dynamic graphs are necessary in order to model many interesting real world phenomena, such as social networks or knowledge graphs. Using the formulation described by previous work (e.g.Rossi et al. (2020)), we classify dynamic graphs as either discrete-time dynamic graphs (DTDGs), wherein a dynamic graph is represented as a set of snapshots of the graph at different points in time, or continuous-time dynamic graphs (CTDGs), wherein nodes and edges can be added, changed, or deleted at any point in time. Many existing approaches for DTDGs represent such graphs as a time series of static graphs and apply static learning methods to learn representations (Liben-Nowell & Kleinberg, 2003; Dunlavy et al., 2011; Yu et al., 2019).

More interesting are CTDGs, as they are able to model several real world phenomena not possible by DTDGs, such as knowledge graphs and social networks. Recently, several methods have been proposed to explicitly deal with representations of CTDGs (Xu et al., 2020; Bastas et al., 2019; Ma et al., 2020; Nguyen et al., 2018a; Kumar et al., 2019a). Perhaps most significantly, Rossi et al. (2020) proposed a generic framework for learning representations of CTDGs, and demonstrated state-of-the-art performance on several CTDG datasets.

Simultaneous to work in graph representation learning, significant effort has focused on developing generative models for graph structured data, with applications ranging from *de-novo* drug discovery (Popova et al., 2019; You et al., 2018) and semantic parsing of natural language graphs (Chen et al., 2018; Wang et al., 2018). However, the overwhelming majority of these methods are limited to static graphs. Recent work in temporal graph generation are either limited to DTDGs (Holme, 2013; Perra et al., 2012; Vestergaard et al., 2014), or lack support for the *inductive* regime (i.e. generalization to unseen graphs) (Zeno et al., 2021; Zhou et al., 2020). Thus, they are unsuitable for the real world datasets used in this work. To the best of our knowledge, there is no framework for generative modeling of CTDGs in inductive regime.

Contributions In this work, we present TG-Gen, a generic *generative framework* for CTDGs which combines an encoder module that creates a temporal latent space embedding from raw interaction data, with a decoder module that uses said embeddings to cereate a deep probabilistic model of the interaction data. We show that TG-Gen is able to generate robust and accurate synthetic data for CTDGs on three diverse datasets. Additionally, we demonstrate that the embeddings learned by TG-Gen are able to outperform previous state-of-the-art learning methods, even in the *discriminative* regime. Finally, we perform comprehensive analysis of several different instantiations of TG-Gen in the form of ablation studies in order to show the effects of specific modules and configurations of our model.

2 Background and Related Work

Representation Learning for Dynamic Graphs Work on graph representation has focused on learning embeddings for graphs that dynamically change over time. Specifically, we can define *discrete-time dynamic graphs* (DTDGs) as time sequences of static graphs. A generalization of DTDGs are *continuous-time dynamic graphs* (CTDGs), which are a timed list of events that can include node and edge addition and deletion as well as node and edge feature evolution at any point in time (Rossi et al., 2020). In this work, we focus on CTDGs, as they can be used to model many real world data (e.g. social networks, knowledge graphs) not possible by DTDGs. Given a static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{1, ..., n\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, we represent a CTDG as a sequence of events over time, $\mathcal{G} = \{x(t_1), x(t_2), ..., x(t_l)\}$ where each timestep $t_i \in \{1, ..., l\}$ represents a node or edge event (that is, addition, deletion, feature transformation). Thus, we can denote the nodes at a particular time T as $\mathcal{V}(T) = \{i : \exists \mathbf{v}_i(t) \in \mathcal{G}, t \in T\}$ and edges as $\mathcal{E}(T) = \{src, dst : \exists \mathbf{e}_{src,dst}(t) \in \mathcal{G}, t \in T\})$, where $\mathbf{v}_i(t)$ and $\mathbf{e}_{src,dst}(t)$ are a node and edge, represented by their feature vector, at time t.

Representation learning for CTDGs are a new, albeit rapidly growing area of research. Many earlier approaches, such as Nguyen et al. (2018b); Bastas et al. (2019) use random walk approaches, wherein temporal information about the graph is incorporated into transition probabilities of the graph. Other approaches (Kumar et al., 2019a; Trivedi et al., 2017; Ma et al., 2020) use recurrent neural networks (RNNs) (Rumelhart et al., 1985) in order to incorporate temporal information into the graph representation. Perhaps one of the most significant advances in representation learning for CTDGs is TGN (Rossi et al., 2020), which proposes a generic framework for inductive learning on such dynamic graphs. TGN uses a memory module of seen nodes and edges over time, and iteratively updates unseen nodes and interactions (edges) using based on this memory. Additionally, TGN incorporates optimizations such as schemes to update the feature representation of "stale" nodes, that is, nodes that have not seen a feature transformation over a set time period. Rossi et al. (2020) shows that many other modern CTDG representation learning approaches such as Trivedi et al. (2019); Kumar et al. (2019b); Xu et al. (2020) are specific instances of this framework. Recent research You et al. (2022) successfully used static GNN architectures to obtain temporal representation of nodes in CTDG by recurrently update their state over time.

Generative Models for Graph Data Generating models for (static) graphs consists of learning a probability distribution $p(G)$ for a given set of input graphs $\Theta = \{G_1, ...\}$ which can be used to generate graphs with similar properties to Θ. Learning such a probability distribution can be difficult due to the vast search space involved (indeed, similar graphs may have several arrangements/orders of nodes and edges). Early approaches such as Karoński & Ruciński (1997); Watts & Strogatz (1998); Albert & Barabási (2002) rely on strong inductive assumptions about the distribution of $p(G)$. These approaches are able to generate realistic synthetic graphs when the underlying distribution falls under the assumed structure, but generally fail to extend to complex real world graphs. Later work (Gamage et al., 2020; Perozzi et al., 2014; Simonovsky & Komodakis, 2018) rely on schemes used in non-graph domains such as generative adversarial networks and variational autoencoders in order to learn graph representations.

Generative models for dynamic graphs is a far newer research area. These models are required to learn a probability function of the data that evolves over time. That is, $p(G, t) \forall t \in \{1, ..., l\}$ Existing approaches in the area of DTDGs include Holme (2013) which uses a so-called exponential threshold network to capture essential temporal information for a set of static graphs and Perra et al. (2012); Vestergaard et al. (2014) which utilize specialized statistially driven functions to derive temporal information from DTDGs. Other approaches extend to the CTGN domain. These include Zeno et al. (2021) which graph motif structures to generate temporal graphs and Zhou et al. (2020) which uses self-attention to generate temporal random walks that can then be used to generate new graphs. However, to the best of our knowledge, there is no existing framework for generative modeling of GTDGs in the inductive regime.

3 The Temporal Graph Generator Network

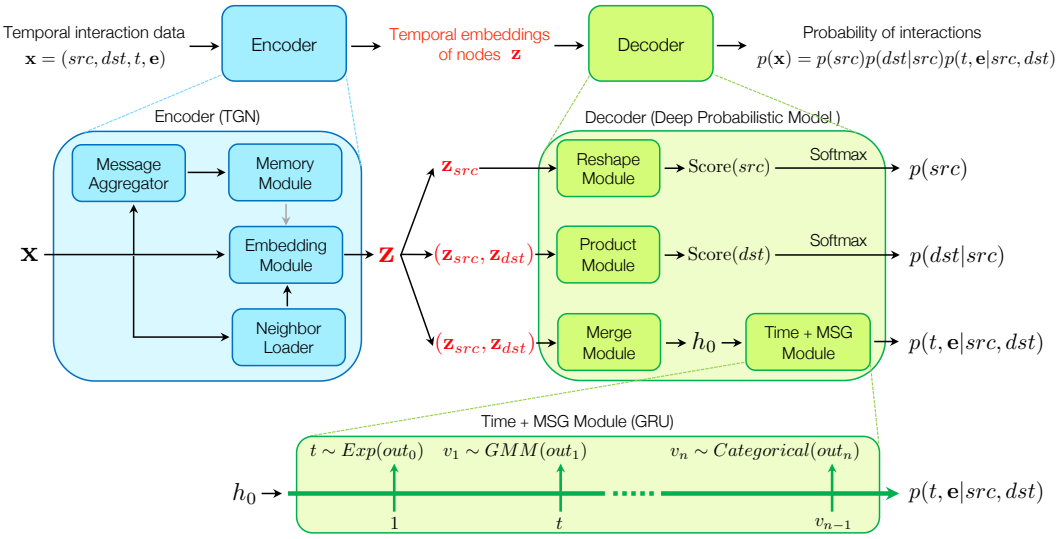

Figure 1: Overview of TG-Gen's Encoder-Decoder architecture and their internal modules, described in Section 3.

TG-Gen is a generative model for temporal graphs that allows to compute the probability of any temporal edge (interaction) and to sample edges from this probability distribution. It has an encoder-decoder architecture, where the encoder takes raw interaction data as input and produces temporal embeddings for the nodes, and the decoder uses the dynamic embeddings of the nodes to build a probabilistic model for the raw interactions (see Figure 1). Specifically, a Temporal Graph Network (TGN) Rossi et al. (2020) model is used as encoder and a deep probabilistic model for tabular data is used as decoder.

3.1 ENCODER

The Encoder processes raw temporal interaction data and creates the temporal embeddings of the nodes. A temporal embedding is a dense vector representation of the state of a node at a particular time, which may depend on the node's current and past interactions and on the state of the other nodes in the graph, such as its temporal neighbors. The specific Encoder model used in our experiments is TGNs Rossi et al. (2020), although other models to create dynamic node representations, like ROLAND You et al. (2022), can also be used. TGN is composed by two main modules: the Memory Module and the Embedding Module. The Memory Module is used to update the memory of each node, which is a vector that stores all relevant information extracted from the node's past interactions in a dense representation. The Memory Module receives as input an aggregation of the node's interactions in the previous temporal batch, combines it with the node's current memory and produces a new, updated memory. The Embedding Module is a Transformer convolution model (Vaswani et al., 2017) that combines the memories of a node's temporal neighborhood together with the corresponding edge features and temporal encodings, to produce the updated node's embedding.

3.2 DECODER

The TG-Gen Decoder uses the temporal embeddings produced by the Encoder to create a probabilistic model of the raw interactions. In temporal graphs, a raw interaction is specified by the following attributes: the ID of the source node (src) the ID of the destination node (dst) the time of the interaction (t) and the message (edge) feature vector ($\mathbf{e}$) of size n. The probability of a single edge is thus $p(src, dst, t, \mathbf{e}) = p(src)p(dst|src)p(t, \mathbf{e}|dst, src)$, where we write the joint probability as the product of three terms: the probability of a node to be a source, $p(src)$, the conditional probability of a node to be a destination of an interaction with a given source node, $p(dst|src)$, and the conditional probability that an interaction between a given source and destination nodes happens at time t and has message features $\mathbf{e}$, $p(t, \mathbf{e}|dst, src)$. The probability of the time and message features given the source and destination nodes can further be decomposed as $p(t, \mathbf{e}|dst, src) = p(t|dst, src)p(v_1|t, dst, src)...p(v_n|v_{n-1}, ..., v_1, t, dst, src)$. All these probabilities are defined using appropriate statistical distributions whose parameters are estimated using neural networks: $p(src)$ is a Categorical distribution over the possible source nodes, where the probability of each node is estimated from the temporal embeddings using the Reshape Module (defined below) followed by a Softmax. $p(dst|src)$ is a Categorical distribution over the possible destination nodes given a source node, where the probability of each node is estimated from the temporal embeddings combined with the embedding of the (known) source node using the Product Module (defined below) followed by a Softmax. $p(t|dst, src)$ is an Exponential distribution for the *inter-event time*, that is the time difference between the current interaction (between the specified source and destination nodes) and the previous interaction. The actual (absolute) time is then obtained by adding the inter-event time to the absolute time of the previous interaction, i.e. performing a cumulative sum of the chronologically-ordered inter-event times. The parameter of the exponential distribution is obtained as the first output of the Time+MSG Module (defined below), a sequence model that is initialized combining the source's and destination's node embeddings via the Merge Module (defined below). $p(e_i|dst, src)$ is a Categorical or Gaussian Mixture Model distribution for the i-th edge (message) feature, depending on whether e_i is a categorical or numerical variable, respectively. The distribution parameters (logits or means, standard deviations and mixture weights) are obtained as the i-th output of the Time+MSG Module. The following paragraphs describe the above-mentioned Modules.

Reshape Module The Reshape Module takes a node embedding vector, $\mathbf{z}_{src}$ as input and returns a scalar representing the score of that node. In our experiments it is defined as a linear transformation followed by a ReLU nonlinearity and another linear transformation to a one-dimensional output: $\text{Reshape}(\mathbf{z}) = \mathbf{W}_f \cdot ReLU(\mathbf{W}_{src} \cdot \mathbf{z})$

Product Module The Product Module takes two embedding vectors as input, $\mathbf{z}_{src}$ and $\mathbf{z}_{dst}$, and returns a scalar representing the score of an interaction between the nodes with the two input embeddings. In our experiments the score is obtained performing a linear transformation of the two input embeddings, adding the resulting vectors, applying an element-wise ReLU nonlinearity and performing a final linear transformation to a one-dimensional output: $\text{Product}(\mathbf{z}_{src}, \mathbf{z}_{dst}) = \mathbf{W}_f \cdot ReLU(\mathbf{W}_{src} \cdot \mathbf{z}_{dst} + \mathbf{W}_{dst} \cdot \mathbf{z}_{dst})$

Merge Module The merge module takes two embedding vectors as input, $\mathbf{z}_{src}$ and $\mathbf{z}_{dst}$, and returns a vector that combines the two embeddings and is used as input to the Time+MSG Module. In our experiments it is obtained performing a linear transformation of the two input embeddings, applying an element-wise ReLU nonlinearity, adding the resulting vectors, applying a second ReLU nonlinearity and performing a final linear transformation to obtain the vector $\mathbf{h}_0$: $\text{Merge}(\mathbf{z}_{src}, \mathbf{z}_{dst}) = \mathbf{W}_f \cdot ReLU(ReLU(\mathbf{W}_{src} \cdot \mathbf{z}_{dst}) + ReLU(\mathbf{W}_{dst} \cdot \mathbf{z}_{dst}))$

Time+MSG Module The Time+MSG Module is a sequence model that takes the output of the Merge Module as input, $\mathbf{h}_0$, and produces $n + 1$ outputs, where the dimension of the i-th output is equal to the number of parameters of the distribution describing the i-th edge feature e_i (for $i > 0$) or the time t (for $i = 0$). In particular, for $i = 0$ the output of the sequence model is passed through a linear layer to obtain a one-dimensional scalar and using a Softplus function is converted to a positive number that represents the parameter of an Exponential distribution for the inter-event time of the interaction. For $i > 0$, the output of the sequential model is converted to a one-dimensional score if the edge feature e_i is a categorical variable described by a Categorical (Multinomial) distribution, whereas it is converted to $3 \cdot m$ numbers corresponding to the mean, standard deviation and weight of the m components of a Gaussian Mixture Model distribution, if e_i is a numerical variable. In our experiments, the sequence model used is a GRU (Chung et al., 2014) recurrent neural network and the Merge Module's output $\mathbf{h}_0$ is used as the initial hidden state of the network.

3.3 Training

TG-Gen is trained using the Adam optimizer (Kingma & Ba, 2014) in order to minimize the negative log-likelihood of the observed interaction data. The model's log-likelihood can be computed exactly as the sum of the log-likelihoods of all the conditional probabilities parametrized by the various modules of the Decoder. In order to accelerate training for large graphs, a random subset of nodes (usually of the order of two times the batch size or larger) is sampled when computing the scores of sources and destinations. In order to improve numerical stability during training of the Time+MSG Module, a Gaussian random noise with zero mean and small standard deviation (~ 0.1) is added to the numerical variables to mitigate the potential instability caused by discontinuous distributions, such as those for the edge features of the Wikipedia dataset (see Figure 2). The standard deviation of this stabilizing noise is reduced to very small values (~ 0.001) during training in order to gradually recover the original shapes of the feature distributions.

3.4 Generation

TG-Gen can generate synthetic dynamic graphs by creating interactions and sampling their attributes from the distribution $p(src, dst, t, \mathbf{e}) = p(src)p(dst|src)p(t, \mathbf{e}|dst, src)$ as follows. First, a batch of source nodes is sampled from $p(src)$ based on the nodes' current embeddings; second, a destination is sampled for each source using $p(dst|src)$; third, the embeddings of each source-destination pair are combined to instantiate the parameters and sample from the probability distributions of the $(t, \mathbf{e})$ variables. Initially, we start with an empty graph and all nodes have an empty memory. After a batch of synthetic interactions is generated, the memories and the list of neighbors are updated and new node embeddings are computed, before generating a new batch of interactions.

4 Experiments

Datasets and Experimental Setup We test our model using three diverse datasets: Reddit, Wikipedia (Kumar et al., 2019a) and Bikeshare. Reddit and Wikipedia are estabished datasets in dynamic graph representation learning while Bikeshare is a novel dataset (see Appendix A.1 for more information regarding these datasets). Our experimental setup is split into two major parts. In the first, we demonstrate that TG-Gen is able to learn generalizable representations of these temporal graphs and thus focuses on the *inductive* domain, where the embeddings created by TG-Gen are used to predict the destination nodes of future links in a link prediction task. We leave evaluation on *transductive* link prediction and dynamic node classification for future work. For all datasets, we follow Rossi et al. (2020) and use a 75%-15%-15% train-eval-test split. Models are trained 5 times across varying random seeds and error bars are reported in our results. The second experiment involves temporal graph generation, which aims at generating synthetic graphs with statistical

	Wikipedia		Reddit		Bikeshare	
	RMSE	K-div	RMSE	K-div	RMSE	K-div
Outdegree	0.0023	0.0086	0.0023	0.0042	0.0009	0.0068
Indegree	0.0019	0.0158	0.0004	0.0014	0.0008	0.0073
Edge weight	0.0013	0.0037	0.0031	0.008	0.0151	0.0381
Interevent Time	0.0004	0.0002	0.0146	0.0227	0.0571	0.1441
Features	0.02 ± 0.01	0.08 ± 0.11	0.11 ± 0.09	1.34 ± 1.96	0.31 ± 0.43	0.22 ± 0.20

Table 1: Values of Root Mean Square Error and K-divergence metrics quantifying the distance between real and synthetic distributions characterizing various properties of the temporal graphs. The distances for the features distributions (last row) are averaged over all features considered in the experiments.

	Wikpedia		Reddit		Bikeshare	
	AP	AUC	AP	AUC	AP	AUC
GAT* †	91.27 ± 0.4	—	95.37 ± 0.3	—	—	—
GraphSAGE*†	91.09 ± 0.3	—	96.27 ± 0.2	—	—	—
Jodie†	93.11 ± 0.4	—	94.36 ± 1.1	—	—	—
TGAT †	93.99 ± 0.3	—	96.62 ± 0.3	—	—	—
DyRep †	92.050.3	—	95.68 ± 0.2	—	—	—
TGN-attn	96.58 ± 0.6	96.20 ± 0.5	98.31 ± 0.1	98.26 ± 0.1	$\mathbf{88.96 \pm 0.5}$	$\mathbf{91.31 \pm 0.6}$
TG-Gen (ours)	$\mathbf{99.9 \pm 0.1}$	$\mathbf{99.9 \pm 0.1}$	$\mathbf{99.71 \pm 0.2}$	$\mathbf{99.71 \pm 0.2}$	$\mathbf{89.68 \pm 2.4}$	$\mathbf{92.16 \pm 1.3}$

Table 2: Average Precision $(\%)$ and AUC $(\%)$ for future edge prediction in the inductive setting. Best performing model is **bolded**. * Static graph method. † Results taken from Rossi et al. (2020).

properties similar to those of real dynamic graphs seen during training. To this end, we evaluate the synthetic data using several statistical measures detailed in the following section.

Evaluation Metrics For the inductive link prediction task, we report standard metrics: Average Precision (AP) and Area Under the ROC Curve (AUC). This is consistent with metrics reported by baseline models. Evaluation for the quality of generated graph data is more nuanced and complex. For the generation task, we report Root Mean Square Error (RMSE $= \sqrt{\frac{\sum_{i=1}^{N}(P_i - Q_i)^2}{N}}$) and K-divergence (K-div $= \sum_{i=1}^{N} P_i \ln \frac{2P_i}{(P_i+Q_i)}$), to quantify the distance between real P_i and generated Q_i distributions for several dimensions of the data, namely graph properties (node out-degree distribution, node in-degree distribution, edge weight distribution), inter-event time distribution, and feature distribution. RMSE and K-divergence are standard metrics used to measure the distance between two distributions: they belong to the L_2 and Entropy families, respectively (Cha, 2007; Lin, 1991), and distance values close to zero indicate that the two distributions are similar.

Baselines For the inductive link prediction task, we use several state-of-the-art models for dynamic graph representation learning as baselines. Namely, methods proposed by Rossi et al. (2020); Veličković et al. (2017); Hamilton et al. (2017b); Kumar et al. (2019a); Xu et al. (2020); Trivedi et al. (2019) are compared to our implementation. Given that to the best of our knowledge, there is no existing method for inductive generation of CTDGs, it is not possible to provide baselines for the graph generation task. However, we study the effect of several changes to our implementation in detailed ablation studies in Section 4.2.

4.1 Results

Inductive Link Prediction Table 2 reports the AP and AUC of TG-Gen verus other state-of-the-art methods for dynamic graph representation learning. Our model outperforms previous methods on the Wikipedia and Reddit datasets and matches TGN-attn (Rossi et al., 2020) on the Bikeshare dataset. We report Average Percision (AP) and Area Under the ROC Curve (AUC) on tested models (TGN-attn and TG-Gen). Results for other models are taken from Rossi et al. (2020) and thus do not include AUC or results for Bikeshare. Future work should rigorously evaluate these other

methods for AUC and for the Bikeshare dataset. We also note that results for TGN-attn vary from that reported by the original authors. This is due to the fact that a slightly different evaluation setup is used. Please see Appendix A.4 for more information.

Graph Generation A trained TG-Gen model can be used to generate a synthetic temporal graph with similar statistical properties of the real temporal graph seen during training. Note that, since nodes are only identified by their embeddings and not by their node ids, there is no hard-coded one-to-one correspondence between real and synthetic nodes, hence TG-Gen allows to generate graphs with a different number of nodes than the original graph. In our experiments, we generate a synthetic graph with 100,000 interactions for each of the three datasets, starting with empty memories and updating memories and embeddings every 10 generated interactions. Several statistical distributions of the generated data look qualitatively similar to those of the real data (see Figure 2). We use Root Mean Square Error and K-Divergence to quantitatively measure the distance between real and synthetic distributions are report the results in Table 1. These indicate that we are able to produce robust and representative synthetic graphs. However, we note that our model is sometimes not able to generate nodes for distrubtions at the tail end of long-tail distributions. This is an open challenge in many areas of machine learning that our model also suffers from. Please refer to Appendix A.3 for more details.

4.2 Ablations

In this section, we investigate the effects of varying or removing key components of our final model on synthetic graph generation.

Memory We compare a version of our model which uses memory in the encoder with an otherwise identical model with no memory. We find that removing the memory module decreases performance across all tested datasets and metrics. This is especially significant in the quality of generated *features*, wherein the model with memory outperforms significantly (e.g. for Wikipedia, 0.02 vs. 0.04 and 0.08 vs. 0.16 for feature RMSE and K-div, respectively). While removing the memory module does increase training and inference speed significantly (about 2.2x), the results of this ablation show that the long-term global information learned by the memory module is essential to performance on the graph generation task.

Choice of Merge Layer We also experiment with a different implementation of the Merge module. Namely, we use an concatination, rather than addition scheme. That is, we modify the formulation discussed in 3 to be $\text{Merge}(\mathbf{z}_{src}, \mathbf{z}_{dst}) = \mathbf{W}_f \cdot ReLU(ReLU(\mathbf{W}_{src} \cdot \mathbf{z}_{dst}) + ReLU(\mathbf{W}_{dst} \parallel \mathbf{z}_{dst}))$. For all datasets and all metrics, we find negligible performance differences. However, the added weights in this concatination scheme lead to comparatively slower training and inference times. Thus, we conclude that addition in the merge module is sufficient to learn a prudent representation of $p(t, \mathbf{e}|src, dst)$ and opt to use that scheme.

Number of Layers We also experiment with adding additional linear layers in each of the decoder modules. We find that this leads to no noticable improvement in any dataset or metric. As discussed by Rossi et al. (2020), this contrasts results for other (discriminative) methods such as Xu et al. (2020) and is likely due to the presence of the memory module which captures complex node relationships without need for additional layers. Thus, we find the analysis presented in Rossi et al. (2020) to hold in the generative regime.

Stabilizing Noise As described in Section 3, we introduce random noise during training in order to lead to more stable convergence. We compare this training scheme with one where no noise is added. We find that the quality of generated data suffers significantly without this noise. For example, on the Wikipedia dataset RMSE at least doubles across all graph properties. Most significant is the feature RMSE which is 0.09 vs. 0.02 on the model with noise. Thus, we empirically show that the analysis of stabilizing noise discussed in Section 3 significantly improves the quality of the generated graph data.

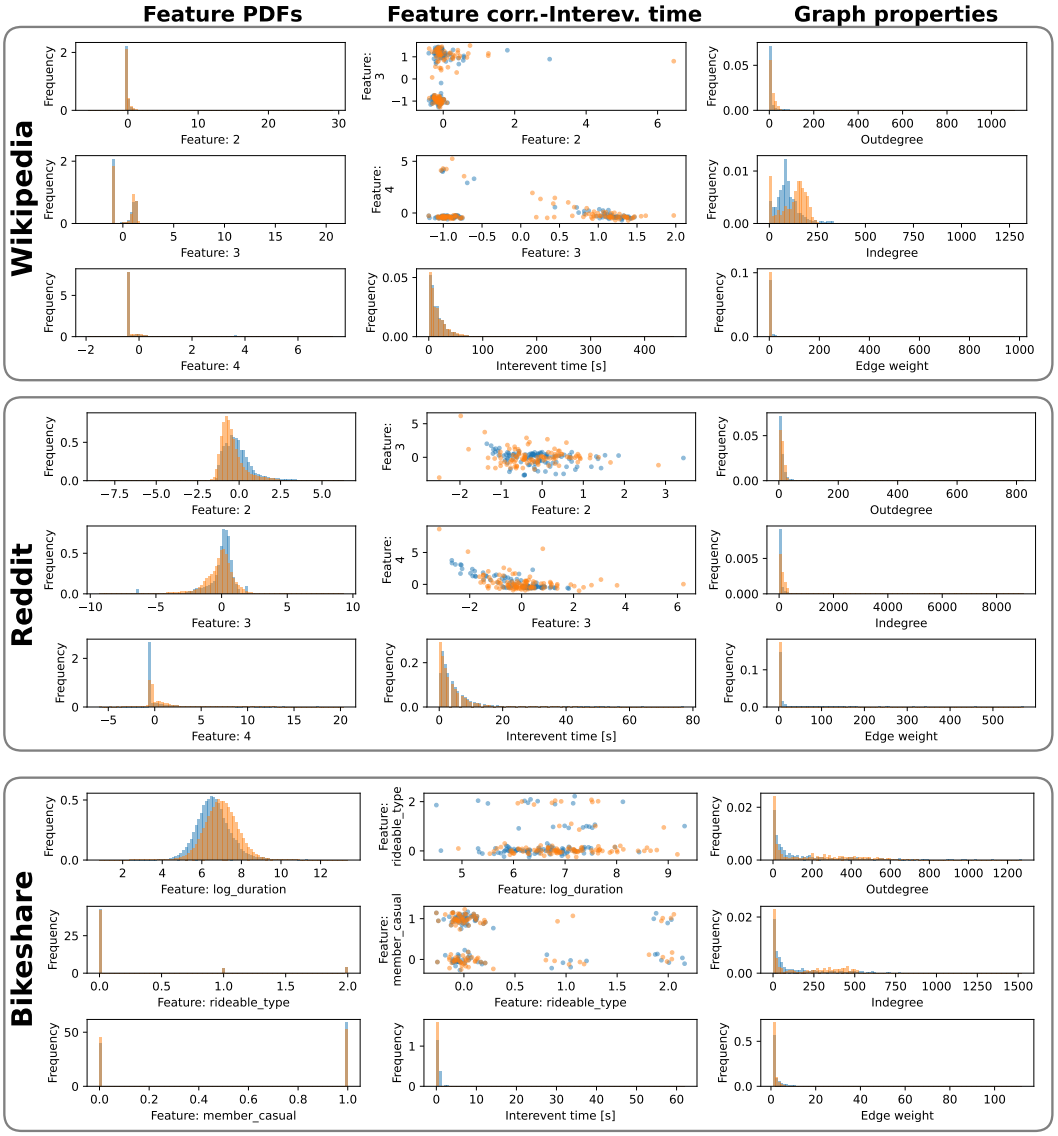

Figure 2: Comparison between statistical distributions of the real data (blue) and of the synthetic data generated by TG-Gen (orange). Plots are grouped by row according to the datasets: Wikipedia (top nine plots), Reddit (middle nine plots) and Bikeshare (bottom nine plots). Plots are also grouped by column according to the kind of statistics considered: histograms of the distribution of three edge features per dataset (left column), correlation plots between features pairs and histogram of the inter-event times between interactions (center column) and histograms of graph properties (right column), namely the outdegree, indegree and edge weight distributions.

5 Conclusion

We introduce TG-Gen, a generic framework for generating a general range dynamic graphs. We show that our method is able to generate representations of dynamic graphs which outperform existing state-of-the-art approaches on established datasets on discriminative tasks such as link prediction. Additionally, we show that TG-Gen is able to generate robust and accurate synthetic data for temporal graphs. We are excited for future applications of our work, as a method for generating continuous temporal graphs in the inductive regime will be an effective tool in a wide variety of domains. We hope that TG-Gen can be used to create synethetic data for a wide variety of tasks,

such as for data augmentation, training data from sensitive information, and *de-novo* generation in areas such as automated drug discovery.

Ethics Statement We hope that the development of a generative framework for dynamic graphs will lead to new innovation in areas such as automated drug discovery, recommender systems, and particle physics simulations. We note that generative models, by definition, can only be as perfect as their underlying training data. For example, our testing of social network graphs is limited to a single social media site, Reddit. Thus, care must be taken when curating datasets to avoid biases, especially in areas wherein non-representative training data (e.g. medical data from a single country) can lead to significant inequity of downstream outcomes.

Reproducibility Statement Here, we summarize the steps taken to ensure reproducability of our results. We clearly state the full assumptions made in our model in the body of the work. Additionally, for downstream task analysis, we run the model 5 times across different random seeds, and report the standard error in 2. Appendix A.2 clearly details the hyperparameter search space used to tune our models and Appendix A.5 includes the computing resources used. Finally, the source code and instructions needed to reproduce our experiment can be found at [*online code repository hidden for anonymous review; please see code accompanying submission*].

REFERENCES

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani. A survey on embedding dynamic graphs. *ACM Computing Surveys (CSUR)*, 55(1):1–37, 2021.

Nikolaos Bastas, Theodoros Semertzidis, Apostolos Axenopoulos, and Petros Daras. evolve2vec: Learning network representations using temporal unfolding. In *International Conference on Multimedia Modeling*, pp. 447–458. Springer, 2019.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252, 2021.

Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical models and Methods in Applied Sciences*, 1(4):300–307, 2007.

Bo Chen, Le Sun, and Xianpei Han. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. *arXiv preprint arXiv:1809.00773*, 2018.

Dawei Cheng, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. Graph neural network for fraud detection via spatial-temporal attention. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 315–324, 2020.

Javier Duarte and Jean-Roch Vlimant. Graph neural networks for particle tracking and reconstruction. In *Artificial intelligence for high energy physics*, pp. 387–436. World Scientific, 2022.

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.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Anuththari Gamage, Eli Chien, Jianhao Peng, and Olgica Milenkovic. Multi-motifgan (mmgan): Motif-targeted graph generation and prediction. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4182–4186. IEEE, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017a.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017b.

Petter Holme. Epidemiologically optimal static networks from temporal network data. *PLoS computational biology*, 9(7):e1003142, 2013.

Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23, 2021.

Xiangyang Ju, Steven Farrell, Paolo Calafiura, Daniel Murnane, Lindsey Gray, Thomas Klijnsma, Kevin Pedro, Giuseppe Cerati, Jim Kowalkowski, Gabriel Perdue, et al. Graph neural networks for particle reconstruction in high energy physics detectors. *arXiv preprint arXiv:2003.11603*, 2020.

Micha Karoński and Andrzej Ruciński. The origins of the theory of random graphs. In *The mathematics of Paul Erdös I*, pp. 311–336. Springer, 1997.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21 (70):1–73, 2020.

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

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*, pp. 1269–1278, 2019a.

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*, pp. 1269–1278, 2019b.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 719–728, 2020.

Giang H Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Dynamic network embeddings: From random walks to temporal random walks. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1085–1092. IEEE, 2018a.

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*, pp. 969–976, 2018b.

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*, pp. 701–710, 2014.

Nicola Perra, Bruno Gonçalves, Romualdo Pastor-Satorras, and Alessandro Vespignani. Activity driven modeling of time varying networks. *Scientific reports*, 2(1):1–7, 2012.

Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*, 2019.

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.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pp. 412–422. Springer, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, pp. 3462–3471. PMLR, 2017.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Christian L Vestergaard, Mathieu Génois, and Alain Barrat. How memory generates heterogeneous dynamics in temporal networks. *Physical Review E*, 90(4):042805, 2014.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. A neural transition-based approach for semantic dependency graph parsing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020b.

Bingbing Xu, Huawei Shen, Bingjie Sun, Rong An, Qi Cao, and Xueqi Cheng. Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4537–4545, 2021.

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

Ruiping Yin, Kan Li, Guangquan Zhang, and Jie Lu. A deeper graph neural network for recommender systems. *Knowledge-Based Systems*, 185:105020, 2019.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Jiaxuan You, Tianyu Du, and Jure Leskovec. ROLAND: graph learning framework for dynamic graphs. In Aidong Zhang and Huzefa Rangwala (eds.), *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 2358–2366. ACM, 2022. doi: 10.1145/3534678.3539300. URL https://doi.org/10.1145/3534678.3539300.

Bing Yu, Mengzhang Li, Jiyong Zhang, and Zhanxing Zhu. 3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting. *arXiv preprint arXiv:1903.00919*, 2019.

Weiping Yu, Taojiannan Yang, and Chen Chen. Towards resolving the challenge of long-tail distribution in uav images for object detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3258–3267, 2021.

Giselle Zeno, Timothy La Fond, and Jennifer Neville. Dymond: Dynamic motif-nodes network generative model. In *Proceedings of the Web Conference 2021*, pp. 718–729, 2021.

Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021.

Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 401–411, 2020.

A Appendix

A.1 Datasets

The Reddit and Wikipedia datasets are bipartite temporal interaction graphs released by Kumar et al. (2019a). The Reddit dataset contains two node types: users and subreddits (communities within the Reddit social network). When a user interacts with a subreddit (e.g. writes a post) an interaction (edge) is formed. The Wikipedia dataset contains the following two node types: users and pages. When a user edits a page, an interaction is formed. For both datasets, all interactions are timestamped and text information is used as features.

The Bikeshare dataset is an open source dataset containing information about users renting city bikes that we convert to a temporal graph. Each node is a station where a user can rent a bike and an interaction (edge) is a user trip between two stations. Edge features include information about the rider ("member" versus "casual user") and bike type (one of "classic_bike" "electric_bike" "docked_bike"). Like the Reddit and Wikipedia datasets, all interactions are timestamped. Unlike the other datasets, this graph is *not* a bipartite graph.

A.2 Experimental Details

As explained in the main work, we followed a 70%-15%-15% train-eval-train temporal data split. However, we also further split the evaluation partition in half (defining an eval-one and eval-two dataset). This allows us to test our top performing models after performing hyperparameter sweeps

to ensure that we are not overfitting to the evaluation set. Test sets were only run once on the chosen model and final numbers reported in the paper.

We used Bayesian search for our hyperameter sweeps, using the approach outlined by Snoek et al. (2012). We sweep across the following parameters: learning rate $[0.00001, 0.001]$, neighborhood distance $\{1, 2, 5, 10, 20, 30, 40, 50, 100, 200\}$, sampled source nodes $values$: $\{250, 300, 400, 500, 600\}$, memory hidden dimension $\{26, 50, 76, 100, 200, 300, 400, 500, 600, 800\}$, time hidden dimension $\{26, 50, 76, 100, 200, 300, 400, 500, 600, 800\}$, embedding dimension $\{26, 50, 76, 100, 200, 300, 400, 500, 600, 800\}$, feature hidden dimension $\{8, 16, 32, 64, 128, 256\}$, and noise added during training $[0.1, 10.0]$. All candidate models were trained for 1,000 epochs and the weights used were at the epoch with the best validation loss.

Please see our accompanying code for hyperparameters of final models for all datasets.

A.3 Tail End Distributions

We note that though it may not be clear from the reported statistics in Table 2 or visualizations in Figure 2, our model sometimes fails at generating nodes at the extreme tail ends of the node distributions. This long-tail distribution issue is a challenging topic in many contemporary areas of machine learning research Zhang et al. (2021); Yu et al. (2021). However, the consequences for graph generated data can be significant. For example, a graph may have very few nodes with a high number of edges but nonetheless have significant impact on the properties of the graph (e.g. airport hub in graph of worldwide airports). We leave the task of improving our generated data and extreme tail ends to future work.

A.4 Evaluation Setup for Link Prediction

Our evaluation setup for link prediction differs slightly from that used by Rossi et al. (2020). Namely, predictions that are within the same batch are made in parallel so any interactions that occur later in the batch will not have access to previous interactions within the same batch. This is different than the approach taken by Rossi et al. (2020) who process interactions sequentially and thus have access to these previous interactions. Thus, our testing on the TGN architecture leads to different results. However, we use this former scheme as the authors of TGN have pubically stated that this approach is realistic and there serves as a better future benchmark.

A.5 Computing Resources

We use a single NVIDIA A100-SXM4-40GB GPU for training and evaluation of a single model and a set of 8 of such GPUs for hyperparameter optimization.