

DyGnROLE: Asymmetric Pretraining for Edge Classification on Dynamic Graphs

Tyler Bonnet and Marek Rei
Imperial College London
 {t.bonnet24, marek.rei}@imperial.ac.uk

Abstract—Edge classification on directed dynamic graphs requires modeling interactions between source and destination nodes exhibiting asymmetrical behavioral patterns and temporal dynamics. However, existing dynamic graph architectures largely rely on shared parameters for processing source and destination nodes, with limited or no systematic role-aware modeling. We propose DyGnROLE (Dynamic Graph Node-Role-Oriented Latent Encoding), a Transformer-based architecture that disentangles source and destination representations. By using separate embedding tables and role-semantic positional encodings, the model captures the distinct structural and temporal contexts unique to each role. Critical in limited-label settings, which are common in edge classification, is a self-supervised pretraining objective we introduce: Directional Role Alignment (DRA). DRA learns distinct but aligned source and destination embedding spaces by training source representations to retrieve their corresponding destination representations while a historical positive masking strategy excludes previously observed interactions from future negative comparisons. The masks introduce a temporally directional training signal in which node pairs progress monotonically from unseen to observed, after which the relationship is eligible only for further alignment. A comprehensive evaluation on four edge classification tasks across eight datasets demonstrates that DyGnROLE consistently outperforms a wide range of state-of-the-art baselines, highlighting the importance of role-aware representation learning and asymmetric pretraining for modeling complex directed interactions when labeled data is limited.

Index Terms—Dynamic Graphs, Graph Transformers, Self-Supervised Learning, Directed Graphs, Edge Classification

I. INTRODUCTION

Dynamic graphs are widely used to model systems where interactions between entities evolve over time, appearing frequently in domains such as e-commerce, knowledge graphs, and social networks. The primary objective is to learn representations that can predict future states of such systems while respecting underlying causal dependencies. To better capture these temporal dynamics, research has largely shifted from discrete snapshot-based approaches, such as DySAT [1] and EvolveGCN [2], to Continuous-Time Dynamic Graph Networks (CTDGNs) that process interactions as a continuous chronological sequence. Seminal frameworks such as JODIE [3], DyRep [4], and TGN [5] established the use of recurrent memory modules to maintain evolving node states. Most recently, the field has adopted the Transformer [6] architecture to overcome the memory bottlenecks of these recurrent models, with approaches such as TCL [7] and DyGFormer [8] using self-attention mechanisms to better encode long-range dependencies within interaction sequences. Parallel to

these architectural advancements, the field has also developed self-supervised learning frameworks in an effort to improve generalization [9]–[12].

While these architectures have achieved significant progress in modeling complex temporal dependencies, they often neglect the distinct functional roles of the participating nodes. Crucially, the interactions within these sequences are often inherently *directed*, with one node acting as the source and the other as the destination. The resulting asymmetry is fundamental: the source node drives the interaction, while the destination node serves as the target. Despite this distinction, existing architectures for dynamic graphs generally ignore role semantics in favor of unified processing, typically relying on shared parameterization, projecting source and destination nodes into a common latent space using identical weight matrices and attention mechanisms. Such a role-blind design forces the model to implicitly infer role-specific logic from the interaction history alone, effectively requiring the shared parameters to capture two distinct behavioral distributions (source vs. destination). The conflation of semantics creates a representational bottleneck, requiring the model to expend significant capacity disentangling these meanings rather than codifying them as architectural priors. These limitations become particularly acute in limited-label regimes, where the model lacks sufficient supervision to rediscover these fundamental structural distinctions from scratch.

To bridge this gap, we propose **DyGnROLE** (Dynamic Graph Node-Role-Oriented Latent Encoding)¹, a Transformer-based architecture designed to enforce role awareness. The core intuition behind DyGnROLE is that source and destination nodes should be treated as distinct entity types that require specialized processing streams. Our architecture implements this via four key innovations: (1) role-semantic positional encodings that distinguish query and neighbor roles across both source and destination contexts; (2) within- and cross-sequence frequency embeddings to capture role-specific recurring interactions and local connectivity patterns; (3) dual-CLS representation pooling to extract separate global representations for source and destination nodes; and (4) Directional Role Alignment (DRA), a self-supervised pretraining objective designed to align these distinct source and destination representation spaces without large labeled datasets.

¹Code will be made available upon publication.

We conduct a comprehensive evaluation of DyGnROLE on four edge classification tasks across eight diverse datasets from the DTGB benchmark [13], spanning e-commerce, social media, knowledge graphs, and communication networks. In the limited-label finetuning regime, DyGnROLE consistently outperforms a wide range of state-of-the-art supervised and self-supervised baselines. An ablation study confirms the necessity of each proposed innovation, revealing that the removal of any of the four key components results in a decline in performance.

II. RELATED WORK

A. Asymmetry in Continuous-Time Dynamic Graph Networks

Modeling the functional asymmetry between source and destination nodes is critical for directed interaction prediction, yet standard CTDGN architectures often neglect this distinction. JODIE [3] serves as a notable exception: by maintaining distinct parameter spaces, including separate mutually-recursive RNNs for predefined user and item sets, JODIE explicitly decouples role processing. However, this bipartite assumption fails in real-world homogeneous networks where any given node can act as either a source or a destination. Standard framework adaptations, such as those in DyGLIB [8] and DTGB [13], resolve this by unifying the parameter space of JODIE into a single shared RNN, which forces identical processing for both roles and removes the asymmetric capacity of the model. To overcome this critical limitation, DyGnROLE integrates role-semantic positional encodings and role-specific embedding tables directly into the architecture, preserving role awareness for every interaction even when individual nodes frequently switch between source and destination roles.

Other architectures addressed broader expressivity limitations but implicitly exchanged structural role-awareness for parameter efficiency. Unified encoding schemes are standard across memory-based models (DyRep [4]), attention-based architectures (TGAT [14]), MLP-based networks (GraphMixer [15]), and walk-based approaches (CAWN [16]). Whether updating a memory state, aggregating neighbors via attention, or encoding random walks, these models apply identical weights and transformation logic regardless of whether the node is acting as a source or a destination.

This limitation persists in state-of-the-art Transformer-based models. DyGFormer [8] and TCL [7] lack architectural mechanisms to encode a node’s functional role during an interaction. Because these Transformer architectures are permutation-invariant and omit explicit role-semantic positional encodings, their encoding streams apply the same parameterized transformations to both interacting nodes. Even though TCL uses a two-stream encoder to partition source and destination subgraphs, it applies identical co-attention weights to both streams. DyGFormer encodes neighbor frequency information using a shared co-occurrence MLP for both source and destination histories, further homogenizing the representation. Consequently, the distinct behavioral logic of source and destination is lost during the representation learning stage. This symmetric processing is fundamentally limited: directionality

is either imposed superficially via a fixed concatenation order at the final classification layer (DyGFormer), or lost through a mathematically commutative scoring function (TCL). This prevents the network from modeling asymmetric interactions during the deep encoding phase, forcing the final prediction module to operate on heavily homogenized representations. In contrast, DyGnROLE integrates role modeling directly into the Transformer architecture. By using independent embedding tables and role-specific encodings, it ensures source and destination nodes are processed as distinct functional entities.

B. Self-Supervised Learning for Dynamic Graphs

To address the challenge of generalization, recent research has increasingly turned to self-supervised learning to extract training signals from unlabeled data. Contrastive methods such as CLDG [9] and IDOL [10] construct multiple views based on chronological evolution. CLDG samples distinct timespan windows to enforce temporal translation invariance across a node’s history, while IDOL uses topology-monitorable sampling to ensure representation coherence between current and historical embeddings. Generative methods such as DyGMAE [12] and DVGMAE [11] use masked autoencoding to capture the temporal evolution of graphs. These models mask structural and temporal information within historical interactions and learn representations by reconstructing the missing data, with DVGMAE further incorporating variational inference to model interaction uncertainty.

However, while optimizing for invariance or reconstruction allows these models to achieve identity stability and structural recovery, they do not enforce the learning of asymmetric interaction patterns. To capture these directed dynamics, DyGnROLE uses a Directed Role Alignment (DRA) objective. Rather than invariance or reconstruction, this objective focuses on the plausibility of interaction. It drives the model to establish an informative structural inductive bias regarding which source-destination pairs are valid given their history. By optimizing this objective, the model geometrically aligns the specialized source and destination embeddings. This process ensures that the distinct role embedding tables serve not only to isolate roles but also to capture the underlying causal and temporal dynamics of the network, providing a strong initialization for complex downstream tasks.

III. METHOD

A. Problem Formulation

A dynamic graph is defined as

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{D}, \mathcal{R})$$

where $\mathcal{V}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{T}$ contains timestamped edges, $\mathcal{T}$ is the set of timestamps at which edges occur, $\mathcal{D}$ is the set of node attributes, and $\mathcal{R}$ is the set of edge attributes. An edge $(u, v, t) \in \mathcal{E}$, with its associated node and edge attributes, connects a source node u to a destination node v at time t , and has a label y_{uv}^t representing the edge class.

At timestamp t , with a set of edges

$$\mathcal{E}_{\leq t} = \{(u, v, t', y_{uv}^{t'}) \in \mathcal{E} \mid t' \leq t\}$$

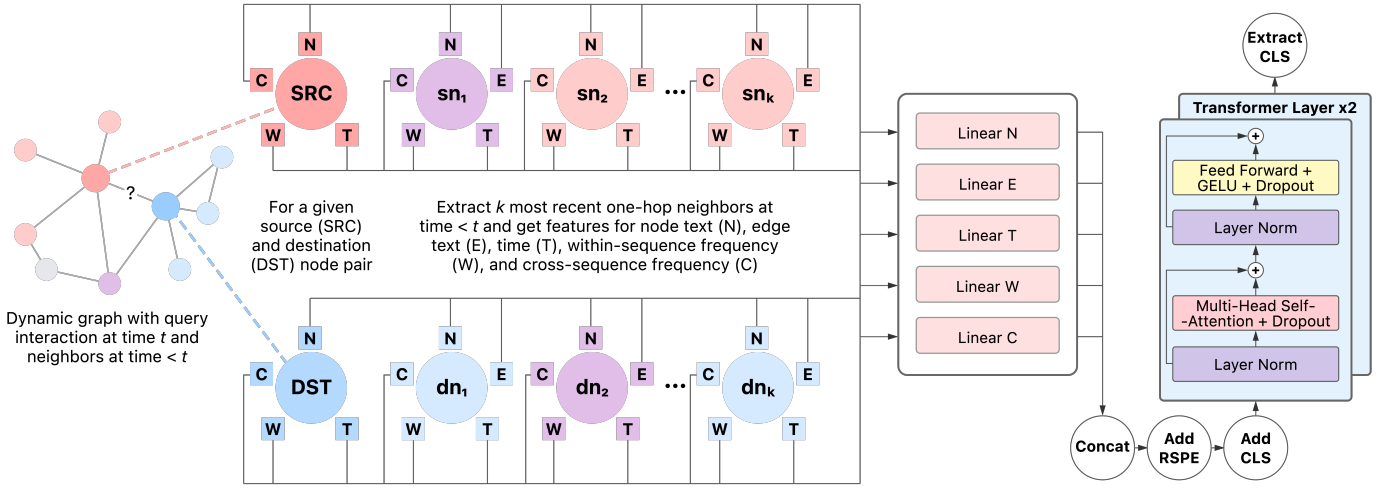


Fig. 1. **Overview of the DyGnROLE architecture.** The model begins processing a query interaction by retrieving the k most recent historical one-hop neighbors for both the source (SRC) and destination (DST) nodes. The query nodes are prepended to their respective neighbor sequences, and the complete sequences are transformed into multi-faceted feature sets including node text (N), timestamps (T), within-sequence frequency (W), and cross-sequence frequency (C). Edge text (E) is also extracted for historical neighbors but is masked for the prepended query nodes to prevent label leakage. These features are linearly projected and concatenated, and role-semantic positional encodings (RSPE) are added to the latent representations. Finally, role-specific CLS tokens are prepended to the sequences before a shared Transformer encoder processes them (independently during pretraining, concatenated during finetuning). The CLS tokens are then extracted to serve as the final node representations $\mathbf{z}_u$ and $\mathbf{z}_v$.

the evaluation follows the edge classification protocol of the DTGB benchmark [13], where the task is defined as predicting the label of an edge at timestamp $t + 1$ given historical information, with the objective of learning a function

$$f : (\mathcal{V}, \mathcal{E}_{\leq t}, \mathcal{D}, \mathcal{R}) \rightarrow y_{uv}^{t+1}$$

which predicts edge classes between node pairs at future timestamps.

B. Model Architecture

DyGnROLE processes the interaction history of source and destination nodes using a Transformer-based architecture designed to capture both temporal dynamics and structural role semantics. To construct the input history, we apply the standard temporal neighbor sampling strategy used in prior works [5, 14], retrieving the k most recent historical one-hop neighbors for both the source and destination nodes. The model takes these sequences as input and transforms them through a multi-stage pipeline consisting of feature construction, role-aware Transformer encoding, and dual-CLS representation pooling.

1) Feature Construction:

a) *Static Textual Features:* Node and edge textual attributes are initialized using TinyBERT [17] and remain static throughout both pretraining and finetuning phases, serving as fixed input representations.

b) *Within- and Cross-Sequence Frequency Embeddings:* To capture role-specific recurring interactions and local connectivity patterns, we introduce a module that counts the frequency of nodes within and across the source and destination sequences. For each node in the sequence, which consists of the query node and its historical neighbors, the module calculates its individual within-sequence and cross-sequence frequencies. Specifically, for a given node in the

source sequence, we record (1) the number of times that node appears in the source sequence and (2) the number of times it appears in the destination sequence. Likewise, for a given node in the destination sequence, we record (3) the number of times that node appears in the destination sequence and (4) the number of times it appears in the source sequence. The discrete counts are mapped to indices and embedded via separate tables. Only count values occurring at least $N_{\min} = 10,000$ times in the pretraining set are assigned distinct indices. Rarer counts are mapped to a single UNK token. This balances granularity with statistical reliability and prevents overfitting to rare values. Formally, the index set for the embedding table is

$$\mathcal{V}_{\text{embed}} = \{c \mid N_c \geq N_{\min}\}$$

where N_c denotes the occurrence of count c . The embedding for a count c is

$$\mathbf{e}_c = \begin{cases} \text{Embedding}(c), & \text{if } c \in \mathcal{V}_{\text{embed}} \\ \text{Embedding}(\text{UNK}), & \text{otherwise} \end{cases}$$

yielding $\mathbf{e}_c \in \mathbb{R}^d$.

c) *Time Encoding:* To capture temporal dynamics, we adopt the standard Fourier-based time encoding scheme used in prior continuous-time graph learning architectures [14]. This module encodes relative timestamps into high-dimensional vectors using a learnable linear layer followed by a cosine activation:

$$\text{TimeEncoding}(t) = \cos(Wt + b), \quad W \in \mathbb{R}^{d_t \times 1}$$

where d_t is the temporal feature dimension. The weight matrix W is initialized using inverse log scales, resembling sinusoidal encoding in Transformers [6].

d) *Feature Projection*: Following the feature fusion design of DyGFormer [8], the node (d_f), edge (d_f), temporal (d_t), within-sequence (d_c), and cross-sequence (d_c) features are each projected into a shared latent space of dimension d_c using independent linear layers. These projections are concatenated along the channel dimension, yielding an embedding of shape $[B, L, 5 \cdot d_c]$.

2) *Role-Semantic Positional Encoding*: For each source and destination node in a query interaction, the features of the node itself are prepended to the sequence of its historical neighbors. Crucially, because the edge connecting the query nodes is the target of prediction, its features must not be leaked to the encoder. Therefore, at the prepended position (index 0), the edge is masked (set to a zero vector). This prevents the encoder from using attributes of the target edge during prediction.

Transformers are inherently permutation-invariant, meaning their output does not depend on the order or position of the input. If the source and destination node sequences of raw concatenated input features are passed to the Transformer, whether independently during pretraining or concatenated during finetuning, the model cannot distinguish between source and destination features. They are treated identically. Similarly, the model cannot differentiate between the query node features at position 0 and the neighbor features at positions 1 through k . To distinguish source from destination and query node from neighbor node within the Transformer, we inject learnable role identifiers directly into the latent space. We define four distinct learnable embeddings: $\mathbf{E}_{\text{src_node}}$, $\mathbf{E}_{\text{dst_node}}$, $\mathbf{E}_{\text{src_neighbor}}$, and $\mathbf{E}_{\text{dst_neighbor}}$. Each embedding has dimension $D = 5 \cdot d_c$, matching the concatenated feature size.

These embeddings are added to the projected features based on the role of the node in the current batch and its position in the sequence:

- $\mathbf{E}_{\text{src_node}}$ is added to the prepended query node feature at the start of the source sequence.
- $\mathbf{E}_{\text{dst_node}}$ is added to the prepended query node feature at the start of the destination sequence.
- $\mathbf{E}_{\text{src_neighbor}}$ is added to all subsequent historical neighbor features in the source sequence.
- $\mathbf{E}_{\text{dst_neighbor}}$ is added to all subsequent historical neighbor features in the destination sequence.

3) *Transformer Encoder*: The backbone of the architecture is a stack of L Transformer encoder layers [6]. Each layer consists of multi-head self-attention followed by a position-wise feed-forward network, both with residual connections and pre-layer normalization. Formally, for input H ,

$$H' = H + \text{Dropout}(\text{MHA}(\text{LayerNorm}(H)))$$

$$H'' = H' + \text{Dropout}(\text{Linear}_2(\text{Dropout}(\text{GELU}(\text{Linear}_1(\text{LayerNorm}(H'))))))$$

where MHA is multi-head self-attention, Linear_1 and Linear_2 are the FFN linear layers, GELU is the activation function [18], and Dropout is applied after both the attention and FFN transformations.

4) *Dual-CLS Representation Pooling*: To generate distinct global representations, we use a dual-CLS strategy. We prepend role-specific CLS tokens with dimension $D = 5 \cdot d_c$ to the neighbor sequences, yielding augmented inputs $\mathbf{X}_u = [\text{CLS}]_S, \mathcal{H}_u$ and $\mathbf{X}_v = [\text{CLS}]_D, \mathcal{H}_v$.

The Transformer input $\mathbf{H}$ depends on the training phase. During pretraining, sequences are processed independently to enforce role separation, while during finetuning, they are concatenated ($\mathbf{H} = [\mathbf{X}_u || \mathbf{X}_v]$) to enable cross-role attention. The encoder output is computed as:

$$\mathbf{Z} = \text{Transformer}(\mathbf{H})$$

We extract the final hidden states of the special tokens as the role-specific node representations:

$$\mathbf{z}_u = \mathbf{Z}[\text{CLS}]_S, \quad \mathbf{z}_v = \mathbf{Z}[\text{CLS}]_D$$

These final embeddings $\mathbf{z}_u$ and $\mathbf{z}_v$ are used for both the pretraining objective and the downstream task.

C. DRA Pretraining

Self-supervised learning is an effective means of improving generalization and preventing overfitting in limited-label regimes. The DRA pretraining objective is designed to achieve these benefits while structurally aligning the distinct source and destination representation spaces constructed by the role-aware components of the DyGnROLE architecture. A critical challenge in applying a noise-contrastive estimation loss [19] to dynamic graphs is the sampling of negatives and handling of historical interactions. Unlike prior works that rely on random negative sampling that may present historical positives as negatives [5], or explicitly treat historical positives as hard negatives [20], we use in-batch negative sampling, where all other destination nodes within the batch that are not the true destination node serve as candidate negatives for a given source node, paired with a historical positive masking strategy. Historically connected nodes represent established structural relationships, yet the learned associations between these nodes are often degraded by existing negative sampling techniques. To preserve these structural alignments and prevent false training signals, we first query the neighbor sampler for the historical neighbors of each node up to the batch's minimum timestamp $t_{\min}$. A binary mask $M \in \{0, 1\}^{N \times N}$ is constructed where $M_{ij} = 0$ if destination node j is a historical neighbor of source node i . The masked pairs are excluded by setting their similarity scores to $-\infty$ before applying the softmax in the loss computation, resulting in no contribution to the training. Importantly, recurring interactions continue to produce a signal, further aligning the representations. The pretraining loss is formalized as

$$\mathcal{L}_{\text{DRA}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\left(\frac{\hat{\mathbf{S}}_i \cdot \hat{\mathbf{D}}_i}{\tau}\right)}{\sum_{j=1}^N M_{ij} \exp\left(\frac{\hat{\mathbf{S}}_i \cdot \hat{\mathbf{D}}_j}{\tau}\right)}$$

where N is the batch size, $\hat{\mathbf{S}}$ is the ℓ_2 normalized source embedding $\mathbf{z}_u$, $\hat{\mathbf{D}}$ is the ℓ_2 normalized destination embedding

$\mathbf{z}_v$, and τ is the temperature parameter set to 0.07. The mask $M_{ij} \in \{0, 1\}$ determines valid comparisons: $M_{ij} = 1$ if $i = j$, which is the true interaction, or if candidate destination j has never interacted with source i prior to the current batch timestamp. Otherwise, $M_{ij} = 0$.

While the node frequency embeddings, role-semantic positional encodings, and dual-CLS pooling provide the architectural capacity for asymmetric role modeling, the DRA objective supplies the training signal required to organize these role-specific representations. During pretraining, source and destination histories are processed independently, producing embeddings from two distinct latent spaces. Formally, the loss is computed row-wise over a source-to-destination similarity matrix, treating source embeddings as queries and destination embeddings as retrieval targets. Because supervision is applied only in this direction, exchanging the roles of source and destination would produce a different optimization problem and generally a different embedding geometry. This distinction is particularly important for directed dynamic graphs, where source and destination nodes often fulfill different functional roles and exhibit different interaction patterns. By aligning the optimization objective with the source-to-destination prediction task, the learned representation space is structured around the directional interaction process itself rather than a symmetric notion of node similarity. Consequently, the model learns a role-dependent compatibility function that captures which destination behaviors are plausible given the historical interaction pattern of a source.

The historical positive masking strategy further introduces a temporal asymmetry into the optimization process. Before two nodes have interacted, they may contribute to the denominator of the objective as negative candidate pairs. Following their first observed interaction, however, the pair is permanently removed from future negative comparisons and may thereafter contribute only as a positive pair if subsequent interactions occur. Thus, a node pair progresses monotonically from being unseen to observed, after which the relationship is eligible only for further alignment. This one-way transition mirrors the temporal evolution of relationships in real interaction networks, where the history established by past interactions is not corrupted through an arbitrary negative pairing scheme.

D. Training and Finetuning

During pretraining, the model is trained using the DRA objective on the training split of the DTGB datasets [13]. During edge classification, the models compute representations $\mathbf{z}_u$ and $\mathbf{z}_v$ for the source and destination nodes, respectively, and these are concatenated and passed through a multi-layer perceptron (MLP):

$$\hat{\mathbf{y}}_{(u,v)} = \text{MLP}([\mathbf{z}_u \parallel \mathbf{z}_v])$$

The MLP consists of two hidden layers with ReLU activations and dropout regularization. Both the pretrained Transformer backbone and the randomly initialized MLP are finetuned end-to-end from the first epoch. The downstream classification

objective is optimized using standard cross-entropy loss between the predicted logits and the ground-truth edge labels. For all models, including baselines, we enforce a general grace period of five training epochs during which the early stopping patience counter is suppressed to allow for initial model stabilization. Following this period, early stopping is evaluated based on the macro F1 score with a patience of 5 epochs.

IV. EXPERIMENTS

A. Datasets and Tasks

To thoroughly evaluate our framework, we conduct our experiments on 4 tasks across 8 diverse datasets from the Dynamic Text-Attributed Graph Benchmark (DTGB) [13]. One of the main reasons we selected DTGB is because its datasets have edge labels, unlike many standard dynamic graph datasets (e.g., MOOC, LastFM, or Wikipedia). The datasets also span a diverse set of domains including email, political events, technical forums, and user reviews, and offer a variety of textual content and label distributions. The datasets are available through the official DTGB repository² as chronologically ordered edge lists with source and destination node IDs, timestamps, and edge labels.

Enron: This dataset includes email communications among employees of the Enron corporation from 1999 to 2002. Nodes represent individual employees, and edges represent emails sent between them. Node texts contain email addresses, while edge texts consist of the raw email contents. The task performed on this dataset is **email topic classification**, where the model classifies the topic of a future email into 10 categories, e.g., “calendar”, “deal”.

GDELT and ICEWS: These datasets are temporal knowledge graphs that model complex geopolitical events. GDELT is built from the Global Database of Events, Language, and Tone³, while ICEWS was transitioned from the Integrated Crisis Early Warning System of Lockheed Martin⁴. In both datasets, nodes represent international political entities, and edges represent a recorded interaction between them. The node and edge text attributes consist of short descriptions of the entities and their interactions. On these datasets, the task of **political relation classification** is performed, where the model classifies a future interaction into one of over 200 categories (236 for GDELT, 266 for ICEWS), e.g., “provide economic aid”, “engage in negotiation”.

Stack E and Stack U: Sourced from Stack Exchange⁵, these forums focus on electrical engineering and the Ubuntu operating system, respectively. Nodes represent users or questions. Edges indicate the submission of an answer to a question or a comment. Node texts include user profiles (name, location, introduction) or question titles and bodies. Edge texts contain the raw written answer or comment. The task performed on

²<https://github.com/zjs123/DTGB>

³<https://www.gdeltproject.org/>

⁴<https://dataverse.harvard.edu/dataverse/icews>

⁵<https://archive.org/details/stackexchange>

TABLE I
RESULTS WITH 10K TRAINING LABELS: AVERAGE MACRO (MF1) AND WEIGHTED (WF1) F1 SCORES (%). BOLD INDICATES BEST ON DATASET.

Method	Amazon		Enron		GDELT		Google		ICEWS		Stack E		Stack U		Yelp		Avg	
	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1
Random	15.2	24.7	7.0	13.0	0.2	0.7	15.0	25.0	0.1	0.6	46.7	53.3	44.7	55.1	16.9	23.1	18.2	24.4
Majority	15.8	51.2	4.1	8.2	0.1	1.2	15.9	52.8	0.1	4.6	42.8	64.1	44.6	72.0	14.0	37.6	17.2	36.5
DyGMAE	15.8	51.2	5.2	15.9	0.1	1.7	15.9	52.8	0.1	4.1	42.8	64.1	44.6	72.0	14.2	37.7	17.3	37.4
DyRep	15.0	41.6	8.7	27.4	0.1	1.5	14.5	43.2	0.1	4.8	43.5	63.5	45.1	70.9	14.8	33.5	17.7	35.8
TCL	15.8	51.2	6.8	16.9	0.1	1.9	15.9	52.8	1.9	21.3	42.8	64.1	44.6	72.0	14.0	37.6	17.7	39.7
IDOL	15.8	51.2	8.4	26.1	0.2	2.8	15.9	52.8	0.1	4.8	42.8	64.1	44.6	72.0	14.1	36.8	17.7	38.8
DVGMAE	16.6	50.6	8.0	25.1	0.1	2.1	15.9	52.8	0.2	5.7	42.8	64.1	45.8	71.9	14.0	37.7	17.9	38.7
DyGFormer	15.8	51.2	7.2	19.0	0.4	4.7	15.9	52.8	1.6	19.9	42.8	64.1	44.6	72.0	20.4	43.0	18.6	40.8
JODIE	17.7	50.3	7.2	21.7	0.1	1.4	17.7	51.3	0.1	4.9	44.2	61.4	45.9	69.0	17.4	39.5	18.8	37.4
CLDG	15.8	51.2	7.0	26.6	0.2	2.7	15.9	52.8	0.3	6.5	42.8	64.1	44.6	72.0	24.1	48.1	18.8	40.5
GraphMixer	17.1	51.9	9.2	24.5	0.2	3.4	15.9	52.8	1.2	15.9	52.4	65.6	44.6	72.0	16.1	38.1	19.6	40.5
CAWN	18.0	52.4	8.6	21.8	0.4	5.3	17.8	52.3	2.1	21.7	42.8	64.1	44.6	72.0	25.7	47.9	20.0	42.2
TGAT	21.5	53.0	8.2	18.5	0.2	3.1	17.4	52.0	2.1	22.1	42.8	64.1	44.6	72.0	23.9	46.1	20.1	41.4
DyGnROLE	23.0	54.5	10.3	27.2	0.6	6.7	22.4	52.8	2.1	21.4	55.6	67.8	53.9	71.1	33.0	54.9	25.1	44.6

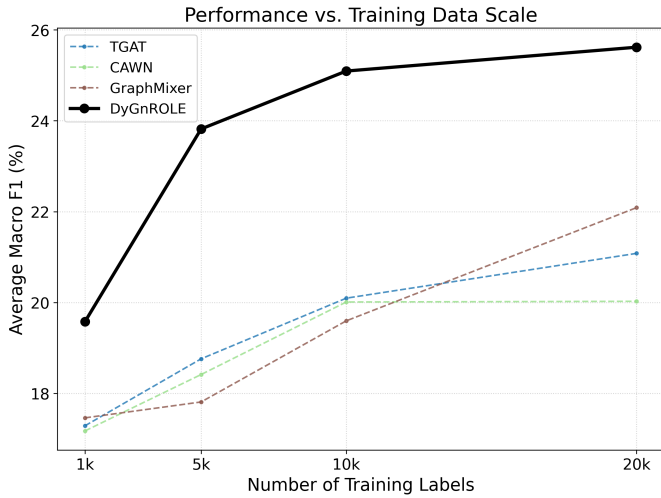


Fig. 2. Average Macro F1 scores of DyGnROLE and the top 3 baselines across varying levels of label availability from 1k to 20k training labels.

these datasets is **post usefulness prediction**, where the model predicts whether a future reply of a user will be “useful” or “useless” determined by the community score (upvotes minus downvotes, where a score greater than 1 is useful).

Google, Amazon, and Yelp: These e-commerce datasets model consumer behavior and user satisfaction. The Google dataset uses Google Local review data covering Connecticut businesses⁶. The Amazon dataset is based on Amazon review data for Movies and TV products⁷. The Yelp dataset uses the Yelp Open Dataset to model reviews of businesses such as restaurants and hotels⁸. Nodes represent users and businesses or products. Edges represent a user reviewing an item. The node attributes include user profiles, product names,

categories, and descriptions. The edge texts capture the raw written user reviews. On these datasets, the task of **rating prediction** is performed, where the model predicts a future rating of 1 to 5 stars assigned by the user for a product or service.

B. Baselines

To ensure a rigorous evaluation, we compare DyGnROLE against two distinct classes of state-of-the-art dynamic graph baselines: standard supervised models and models capable of self-supervised learning. The standard supervised class includes memory-based models (JODIE [3], DyRep [4]), a walk-based model (CAWN [16]), and attention-based models (TGAT [14], TCL [7], GraphMixer [15], DyGFormer [8]). The self-supervised class includes contrastive models (CLDG [9], IDOL [10]) and generative masked autoencoders (DyGMAE [12], DVGMAE [11]). We discuss the representational limitations and shared parameterization of these architectures in Section V-C.

C. Evaluation Protocol

We report Macro and Weighted F1 scores for model evaluation. While we include Weighted F1 for consistency with the DTGB benchmark, we prioritize Macro F1 for our analysis. In real-world interaction networks, class distributions are often heavily skewed, yet rare interaction types (e.g., military action or high-quality technical answers) frequently carry the most semantic importance. Macro F1 ensures that models are evaluated on their ability to capture these distinct behavioral patterns rather than merely memorizing the dominant statistical trend.

The experimental setup handles unlabeled data based on the native capabilities of each baseline class. Standard supervised baselines lack self-supervised objectives and require labeled data to learn representations, rendering them incapable of processing the unlabeled history. Conversely, the self-supervised

⁶https://mcauleylab.ucsd.edu/public_datasets/gdrive/googlelocal/

⁷https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

⁸<https://www.yelp.com/dataset>

baselines and DyGnROLE use unlabeled data during a pre-training phase. The models are pretrained on the full unlabeled training split of a given dataset.

We enforce limited-label regimes to evaluate data efficiency. We evaluate model performance across four settings of label availability: 1,000 training and 150 validation edges; 5,000 training and 750 validation edges; 10,000 training and 1,500 validation edges; and 20,000 training and 3,000 validation edges. In each setting, models are trained using only the specified number of the most recent labeled edges immediately preceding the validation split. The models are validated using the specified number of the most recent edges immediately preceding the test split. Performance is evaluated on the full test split. All reported metrics are averaged over 5 independent runs using different random seeds.

D. Implementation

We optimize DyGnROLE using AdamW with a learning rate of $1e-4$, weight decay of 0.01, and a batch size of 256. The architecture is configured with 2 Transformer layers, 2 attention heads, a channel embedding dimension (d_c) of 50, a temporal feature dimension (d_t) of 100, and a dropout rate of 0.1. The maximum input sequence length is capped at 10, and the DRA temperature is set to 0.07. Early stopping is based on mean reciprocal rank (MRR) with a patience of 5 epochs.

V. RESULTS AND ANALYSIS

A. Main Results

The edge classification results are presented in Table I. DyGnROLE consistently outperforms the baselines, achieving the highest Macro F1 scores on 7 out of 8 datasets and tying for the highest on the 8th. On Yelp, our method achieves a Macro F1 of 33.0%, surpassing the strongest baseline, CAWN (25.7%), by a margin of 7.3 points. Similarly, on Google, DyGnROLE outperforms CAWN by 4.6 points (22.4% vs 17.8%). While performance on the political knowledge graph ICEWS remains competitive with state-of-the-art baselines (tying CAWN and TGAT at 2.1% Macro F1), DyGnROLE demonstrates superior generalization on the GDELT dataset, where it achieves a 50% relative improvement over the next best methods (0.6% vs 0.4%).

Notably, the baselines struggle to generalize in this limited-label regime, often collapsing to a trivial majority-class predictor. This is most evident on the Stack U dataset, where eight out of 11 baselines achieve an identical Macro F1 of 44.6% and Weighted F1 of 72.0%. This score corresponds exactly to predicting the majority class (“useless”) for every interaction. This reveals a fundamental architectural weakness: existing dynamic graph architectures struggle to transcend static frequency biases without role-aware modeling and pretraining. In contrast, DyGnROLE’s ability to maintain distinct behavioral profiles for source and destination nodes allows it to escape this local optimum, achieving a Macro F1 of 53.9% on Stack U and 55.6% on Stack E, an 8.0-point and 3.2-point improvement over the next best models, respectively. On Enron and Stack U, while certain baselines achieve marginally higher Weighted

F1 scores as a direct result of predicting the majority class more frequently at the expense of minority class accuracy, DyGnROLE maintains a higher Macro F1.

The underperformance of the self-supervised baselines, despite their access to extensive unlabeled pretraining data, suggests a mismatch between the objectives used during pretraining and the requirements of the downstream edge classification task. Contrastive methods such as CLDG and IDOL primarily optimize for temporal invariance or node-level identity stability across evolving graph snapshots. CLDG, for example, constructs multiple temporal views of a node’s history and encourages their representations to remain consistent over time, promoting invariance to chronological variation rather than sensitivity to the directional semantics of individual interactions [9]. Similarly, IDOL is designed to preserve identity coherence under graph evolution by contrasting current and historical representations generated from topology-aware sampling strategies [10]. While these objectives can produce stable node embeddings, they do not explicitly encourage the model to organize representations around the interaction patterns that distinguish edge classes.

Generative approaches such as DVGMAE and DyGMAE exhibit a similar disconnect. DVGMAE reconstructs masked graph structure through a variational masked autoencoding objective, learning representations that are effective for recovering masked structural and attribute information [11]. DyGMAE likewise reconstructs masked structural and temporal context from interaction histories, using multi-scale masking strategies to improve representation quality for downstream link prediction and node classification tasks [12]. While these objectives undoubtedly capture meaningful interaction patterns, the supervision they extract from unlabeled data differs substantially from the requirements of edge classification. Success under these objectives is measured by how accurately missing graph elements can be recovered from the latent representation, whereas edge classification requires representations that distinguish among different classes of interactions between nodes. Consequently, the latent spaces induced by these objectives are organized around reconstruction of graph structure and context, whereas DRA organizes the representation space around source-to-destination interaction compatibility. This closer match between the pretraining objective and the directional interaction process underlying edge classification provides a more informative initialization for downstream learning when labeled data is scarce.

B. Ablation Study

To validate the contributions of the DRA pretraining objective and the role-aware architectural features, we conduct an ablation study. The same pretraining (where applicable) and finetuning procedure used for the main experiment is used here in the 10k limited-label regime. Table II compares the complete DyGnROLE architecture against variants partitioned into two groups: *Baseline Architecture* variants, which measure the contribution of DRA pretraining on the most basic version of our architecture where RSPE, NFE, and CLS

TABLE II
ABLATION RESULTS SHOWING AVERAGE MACRO (MF1) AND WEIGHTED (WF1) F1 SCORES (%). BOLD INDICATES BEST ON DATASET.

Variants	Amazon		Enron		GDELT		Google		ICEWS		Stack E		Stack U		Yelp		Avg	
	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1	MF1	WF1
<i>Baseline Architecture</i>																		
w/o DRA	16.5	51.9	6.3	14.2	0.3	4.0	15.9	52.8	1.6	19.5	42.8	64.1	44.6	72.0	14.0	37.6	17.8	39.5
w/ DRA	20.7	51.9	9.2	21.1	0.6	6.6	16.9	53.2	1.6	18.5	56.0	67.6	55.0	70.5	28.6	52.0	23.6	42.7
<i>DyGnROLE Architecture</i>																		
w/o RSPE	22.3	54.3	10.0	26.7	0.6	6.5	21.5	52.9	2.0	21.2	54.3	68.2	51.9	72.8	27.9	50.4	23.8	44.1
w/o CLS	21.1	53.8	9.1	26.4	0.6	6.7	22.0	49.1	2.0	21.8	55.8	68.3	53.7	70.1	27.8	49.6	24.0	43.2
w/o NFE	22.7	54.6	9.4	25.6	0.6	7.1	17.3	53.5	2.0	21.7	55.9	68.3	53.3	71.1	32.8	55.0	24.3	44.6
Complete	23.0	54.5	10.3	27.2	0.6	6.7	22.4	52.8	2.1	21.4	55.6	67.8	53.9	71.1	33.0	54.9	25.1	44.6

have been removed, and *DyGnROLE Architecture* variants, which measure the contributions of RSPE, NFE, and CLS individually with a remove-one ablation protocol while DRA pretraining is run as normal. The Baseline group includes: (1) **w/o DRA**, which is the base architecture trained solely on the limited labels without DRA pretraining; and (2) **w/ DRA**, which applies our DRA pretraining with the base architecture before finetuning. The DyGnROLE group includes: (3) **w/o RSPE**, which removes role-semantic positional encodings; (4) **w/o CLS**, which replaces role-specific dual-CLS pooling with mean pooling; and (5) **w/o NFE**, which replaces asymmetric node frequency embeddings with symmetric co-occurrence features.

The results clearly isolate the impact of the self-supervised asymmetric DRA pretraining phase. Comparing w/o DRA to w/ DRA reveals a substantial performance gain. Across the evaluated datasets, w/ DRA outperforms w/o DRA by an average of 5.8% in Macro F1, rising from 17.8% to 23.6%. This uniform improvement confirms that the DRA objective, with its directional retrieval task and historical positive masking, successfully establishes a robust structural inductive bias. By organizing the latent space around the plausibility of source-to-destination interactions before fine-tuning, DRA prevents the model from overfitting to the limited labeled data.

Furthermore, the results show that while the DRA objective provides a strong structural foundation, the specialized role-aware architectural components are required to capture fine-grained directed dynamics and reach peak performance. The Complete DyGnROLE model improves upon w/ DRA, raising the average Macro F1 score from 23.6% to 25.1%. The specific architectural contributions become evident in the failure modes of the individual features. The specific architectural contributions become evident in the failure modes of the individual features. On the Yelp dataset, removing role-semantic positional encodings (w/o RSPE) or dual-CLS pooling (w/o CLS) reduces the Macro F1 score by over 5 percentage points, indicating that role disentanglement is vital for modeling user-item interactions. On the Google dataset, which relies heavily on structural interaction counts, replacing asymmetric frequency embeddings with symmetric features

(w/o NFE) results in the largest drop, falling from 22.4% to 17.3%. Ultimately, these findings suggest that the pretraining objective helps align the interaction space, while the role-aware features provide the required architectural capacity to fully use this asymmetric training signal.

C. Role-Agnostic Processing in Existing Architectures

A central claim of this work is that many existing dynamic graph architectures rely on role-agnostic parameterizations, using shared transformation functions for nodes acting as sources and destinations. As a result, role-specific behavior must be inferred implicitly from interaction histories rather than encoded directly through architectural priors. To support this, we provide a detailed technical discussion of the symmetrical processing of existing dynamic graph architectures.

In **CAWN** [16], network motifs are extracted via temporal random walks and relies on a set-based anonymization strategy to track node appearances relative to both the queried source and destination nodes, formalized as I_{CAW} in Eq. 3 of their work. While this theoretically captures dual-role appearances, the framework discards directional context during the neural encoding phase. According to Eq. 6 of the CAWN formulation, the node encodings from the source and destination sets are aggregated via sum-pooling. As stated in the methodology, this design choice was made because “the order of u, v is not relevant”. The operation merges source-relative and destination-relative appearance statistics into a shared representation, mathematically merging “closeness to source” and “closeness to destination” into a single symmetric feature vector before it is processed by the recurrent encoding layers.

Similarly, **TGAT** [14] processes local neighborhoods using shared aggregation functions. In TGAT, the temporal graph attention layer constructs an entity-temporal feature matrix $Z(t)$ by concatenating node features with functional time encodings, as defined in Eq. 6 of their formulation. This matrix is then projected into query $q(t)$, key $K(t)$, and value $V(t)$ spaces using weight matrices W_Q , W_K , and W_V . Crucially, these projection matrices and the subsequent dot-product attention operations are globally shared across all nodes. When encoding an interaction (u, v) , the architecture processes the historical neighborhood $\mathcal{N}(u; t)$ to compute

the embedding for u , and processes $\mathcal{N}(v; t)$ to compute the embedding for v , applying the exact same transformation logic to both. Because the embedding generation treats both the source and the destination merely as identical topological centers, the mathematical formulation lacks any parameterization conditioned on the node’s functional role, stripping away the directional asymmetry of the current interaction.

GraphMixer [15] shares this limitation by applying entirely symmetric transformations through an MLP-based architecture. To generate context-dependent representations, the model applies a link-encoder based on an MLP-Mixer to summarize temporal link information and a node-encoder using neighbor mean-pooling to aggregate 1-hop spatial features. However, the exact same projection layers, token-mixing MLPs, and channel-mixing MLPs are applied to both the source node history and the destination node history. By passing both the source and destination queries through identical encoding pipelines before concatenating them for the final link classifier, the model maps source and destination histories through the same sequence of transformations, requiring role-specific behavior to emerge implicitly from the input histories.

This semantic conflation persists in state-of-the-art Transformer-based architectures. **DyGFormer** [8] attempts to capture structural overlap using a neighbor co-occurrence encoding scheme. As defined in Eq. 1 of their formulation, after counting the appearances of a neighbor in both the source and destination histories, the model maps these two discrete counts through a shared transformation function and sums the results. This summation mathematically merges the source-side and destination-side occurrences, eliminating the distinction between the two roles. Furthermore, the architecture aligns and concatenates the encoded sequences before processing them through identical Multi-head Self-Attention blocks. Because the attention mechanism applies the same projection matrices regardless of the node’s initial role in the queried interaction, the resulting representations are produced through a shared encoding function that does not explicitly condition on source or destination role. **TCL** [7] shares a similar bottleneck. While the framework uses a two-stream encoder to separately process the source and destination subgraphs, the intermediate embeddings of both streams are projected into query, key, and value spaces using the exact same weight matrices (W_Q , W_K , W_V), as defined in Eq. 15 and 16 of their work. Furthermore, the co-attentional Transformer defined in Eq. 17 and 18 computes the final representations by exchanging key-value pairs between the two streams. Because these cross-attention operations apply identical linear projections and transformation logic regardless of the queried role, the final representations remain projected into a common latent space.

In memory-based architectures such as **JODIE** [3] and **DyRep** [4], node states are maintained in a unified continuous memory matrix. As previously noted, while the original JODIE formulation assumes a strictly bipartite graph with independent user and item RNNs (detailed in Section 3.1), adapting the model for general homogeneous networks requires collapsing

these distinct update functions into a single shared RNN cell. When an interaction (u, v) occurs, the network constructs messages for both the source node u and the destination node v . Crucially, these messages are processed by the exact same recurrent weight matrices to produce the updated memory states, and the subsequent temporal projection operation (detailed in Section 3.2) applies a shared linear transformation to both roles. **DyRep** exhibits an identical bottleneck. According to Eq. 4 of the **DyRep** formulation, the recurrent update for an interacting node combines temporally attended structural neighborhood information, previous memory states, and exogenous time-shift features. However, the weight matrices governing this update (W^{struct} , W^{rec} , and W^t) are globally shared across all entities. During an interaction, the architecture applies the exact same parametric function to update both the source and the destination memory states. Because the update and projection functions are shared across roles, neither architecture explicitly conditions its parameterization on whether a node is acting as the source or destination of the current interaction.

Self-supervised learning architectures similarly suffer from a lack of role-modeling capacity. Contrastive models such as **CLDG** [9] and **IDOL** [10] process historical interactions through entirely symmetric spatial aggregations. In **CLDG**, dynamic graphs are divided into timespan views and processed via a shared Graph Convolutional Network (GCN). As defined in Eq. 6 of their formulation, the GCN applies identical transformation matrices to all nodes. Subsequently, the network computes local neighbor representations using a uniform readout aggregation (Eq. 7) and maps them into the contrastive space using a shared projection head (Eq. 8). **IDOL** shares this limitation by relying on a decoupled architecture that calculates node embeddings via a K -hop Push algorithm. As formalized in Eq. 3 of their work, this topological diffusion propagates attributes uniformly along the graph structure without distinguishing functional intent. Furthermore, the architecture processes nodes by passing their diffused embeddings individually through a shared MLP encoder. Because the same K -hop propagation logic and shared MLP weights (w_θ) are applied to all nodes regardless of their functional role, the resulting embeddings are generated through shared transformation functions that do not explicitly distinguish source and destination.

Generative models such as **DVGMAE** [11] and **DyGMAE** [12] similarly suffer from this structural homogenization. In the **DVGMAE** implementation, the architecture processes entire graph snapshots uniformly rather than isolating role-specific interaction streams. As defined in Equations 8 and 9 of their formulation, the model extracts the mean and variance for the variational distribution by applying shared GCNs across the global adjacency matrix. Furthermore, the globally enhanced decoder reconstructs the adjacency matrices using a symmetric dot product between the decoded embeddings, formalized in Equations 13 and 15 as an inner product between identical GNN decoders ($f_1 = f_2$). This design provides no architectural mechanism for maintaining

role-specific representations. **DyGMAE** operates under an identical symmetric bottleneck. Despite introducing a Multi-Scale Masking Strategy to generate diverse structural views, the architecture processes graph snapshots uniformly. The spatial and temporal contexts of all nodes are aggregated using globally shared GCN layers operating on the global adjacency matrix (Eq. 4) and a shared GRU sequence model. Finally, the native decoder of the network predicts edge formations using a symmetric dot-product between these unspecialized node representations, as defined in Section 3.4 (Eq. 17). This design provides no explicit architectural mechanism for encoding source and destination roles differently during the representation learning phase.

VI. CONCLUSION

In this work, we identified a fundamental limitation in existing dynamic graph neural networks: the symmetric modeling of asymmetric interactions. We proposed DyGnROLE, an architecture that explicitly disentangles source and destination roles through specialized role-semantic positional encodings, node frequency embeddings, and a dual-CLS pooling strategy. To address the challenge of limited-label regimes common in real-world edge classification tasks, we proposed Directional Role Alignment (DRA), a self-supervised pretraining objective. DRA aligns the distinct role representation spaces while using a temporally directional training signal to preserve learned associations with a historical positive masking strategy.

Our comprehensive evaluation on four tasks across eight diverse datasets from the DTGB benchmark demonstrates that DyGnROLE consistently outperforms state-of-the-art supervised and self-supervised baselines. An ablation study confirms the necessity of DRA and all three role-aware architectural components for optimal performance. These results suggest that elevating directionality to a central architectural component, rather than treating it as an implicit feature, is a promising direction for the next generation of dynamic graph learning models.

REFERENCES

- [1] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM)*. ACM, 2020, pp. 519–527.
- [2] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34. AAAI Press, 2020, pp. 5363–5370.
- [3] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, AK, USA: ACM, Jul. 2019, pp. 1269–1278.
- [4] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International Conference on Learning Representations (ICLR)*, 2019.
- [5] E. Rossi, B. P. Chambers, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [7] L. Wang, X. Chang, S. Li, Y. Chu, H. Li, W. Zhang, X. He, L. Song, J. Zhou, and H. Yang, "Tcl: Transformer-based dynamic graph modelling via contrastive learning," *arXiv preprint arXiv:2105.07944*, 2021.
- [8] L. Yu, L. Sun, B. Du, and W. Lv, "Towards better dynamic graph learning: New architecture and unified library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 67 686–67 700.
- [9] Y. Xu, B. Shi, T. Ma, B. Dong, H. Zhou, and Q. Zheng, "Cldg: Contrastive learning on dynamic graphs," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, Apr. 2023, pp. 696–707.
- [10] Z. Zhu, K. Wang, H. Liu, J. Li, and S. Luo, "Topology-monitorable contrastive learning on dynamic graphs," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2024, pp. 4700–4711.
- [11] M. Gao, X. Zhang, P. Jiao, T. Li, and Z. Zhao, "Dygmoe: Self-supervised dynamic variational graph masked autoencoder," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 10, pp. 17 901–17 913, 2025.
- [12] W. Liu, J. Cheng, Z. Pan, C. He, and Q. Guan, "Dygmoe: A novel dynamic graph masked autoencoder for link prediction," in *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, S. Chiappa and S. Magliacane, Eds., vol. 286. PMLR, 21–25 Jul 2025, pp. 2685–2700.
- [13] J. Zhang, J. Chen, M. Yang, A. Feng, S. Liang, J. Shao, and R. Ying, "Dtgb: A comprehensive benchmark for dynamic text-attributed graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37. Curran Associates, Inc., 2024, pp. 91 405–91 429.
- [14] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *International Conference on Learning Representations (ICLR)*, 2020.
- [15] W. Cong, S. Zhang, J. Kang, B. Yuan, H. Wu, X. Zhou, H. Tong, and M. Mahdavi, "Do we really need complicated model architectures for temporal networks?" in *International Conference on Learning Representations (ICLR)*, 2023.
- [16] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," in *International Conference on Learning Representations (ICLR)*, 2021.
- [17] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Nov. 2020, pp. 4163–4174.
- [18] D. Hendrycks, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.
- [19] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
- [20] F. Poursafaei, S. Huang, K. Pelrine, and R. Rabbany, "Towards better evaluation for dynamic link prediction," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 32 928–32 941.