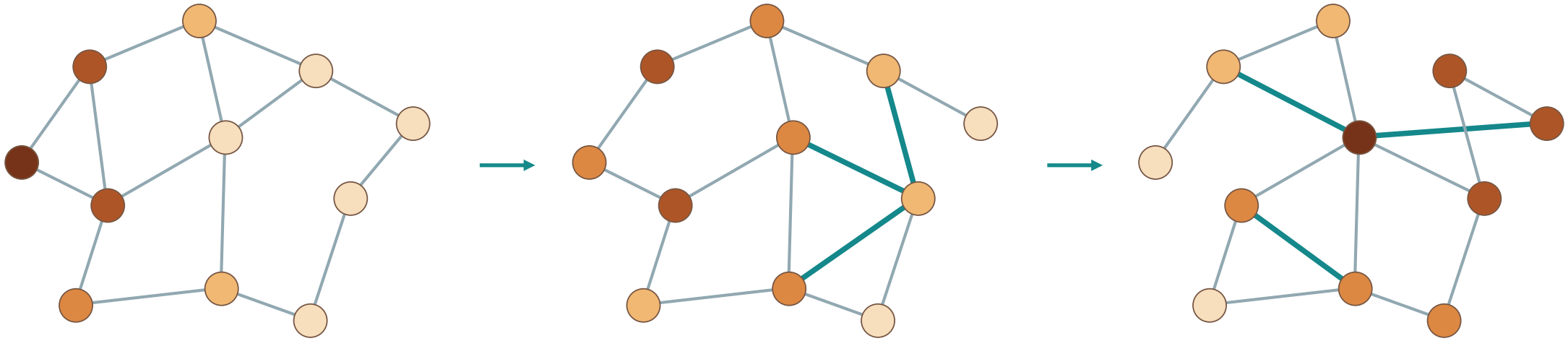
Lecture 21 Temporal graph learning



Forecasting signals and interactions

Two prediction settings, followed by evaluation and current research

1 Data, targets, and available history

2 Forecasting signals on graphs

3 Learning from interaction streams

4 Evaluation and recent research

Data and prediction targets

SECTION 1 · OVERVIEW

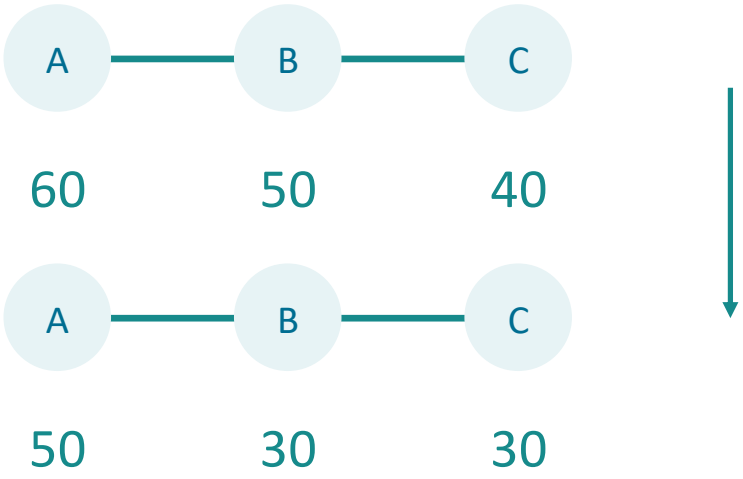
Data	Snapshots describe node values; Events describe timestamped interactions.
Prediction	Specify the target, query time, and available history.
Shared principle	Every model uses information available at its prediction cutoff.

Start with the data, prediction target, and available history.

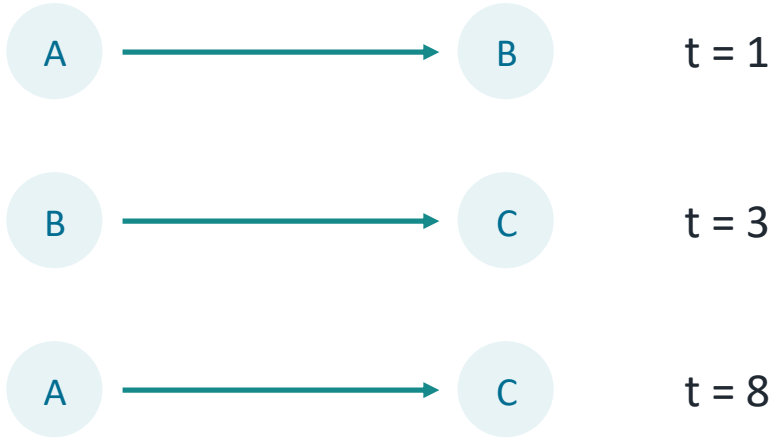
Changing signals and changing interactions

Time can change **node values, graph structure, or both**

Changing node values

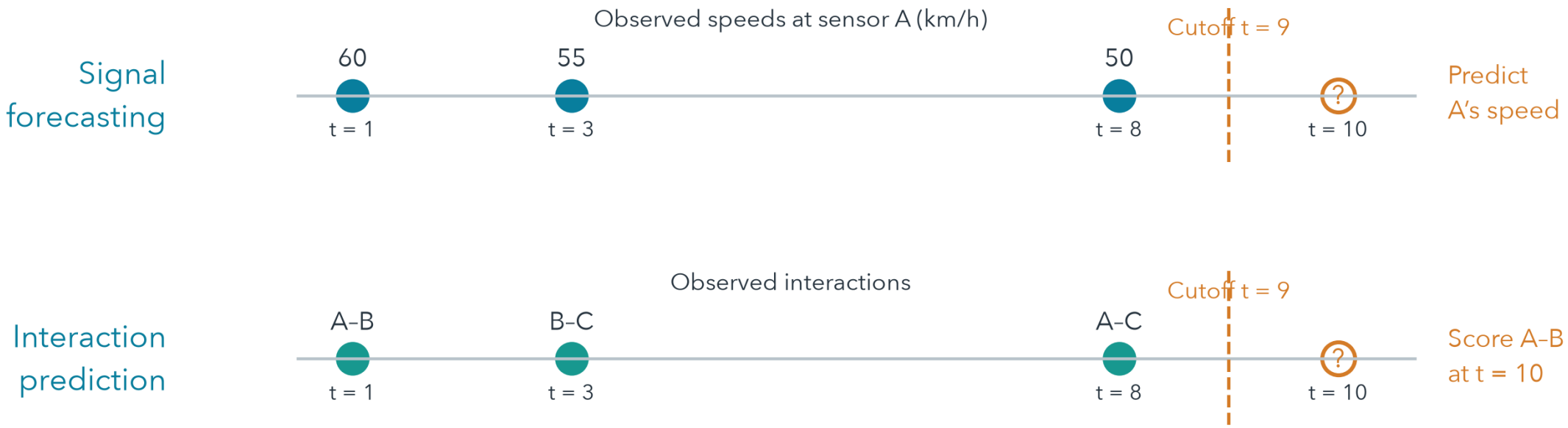


Timestamped interactions



Define the **target** and **cutoff**: use the history available when the prediction is made.

Prediction targets and the information cutoff



The cutoff selects the available history. The query specifies the target and prediction time.

Training compares predictions with the subsequently observed targets.

Forecasting graph signals

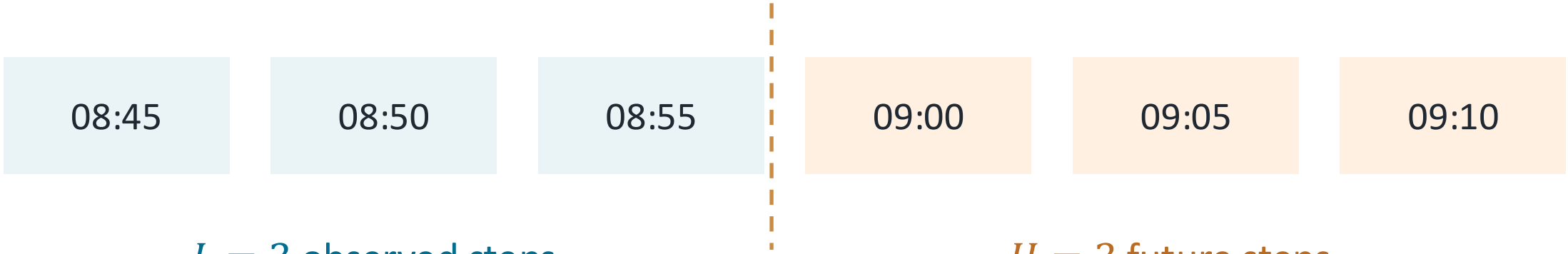
SECTION 2 · OVERVIEW

Common computation	Combine each node's history with information from related nodes.
Architecture choices	DCRNN: recurrence; STGCN: convolutions; Graph WaveNet: adaptive graph.
Evidence	Compare baselines and report errors by forecast horizon.

Focus: how **spatial** information and **temporal** history support forecasting.

Forecast windows and horizons

L observed steps produce H future steps



$$X_t \in \mathbb{R}^{N \times F}$$

$$\hat{X}_{t+1:t+H} = f_\theta(X_{t-L+1:t}, A)$$

Chronological training, validation, and testing

The deployment timeline determines the data split

Train: fit parameters

Validation: choose settings

Test: evaluate later targets



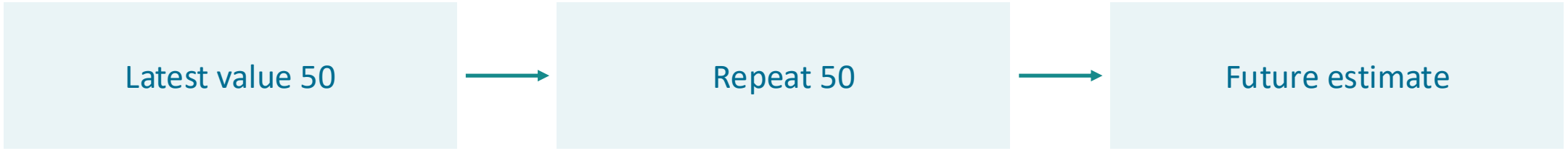
Fit preprocessing and any learned graph using the permitted history

Evaluate MAE by horizon: average |prediction − observed target|.

Simple baselines reveal the value of the graph

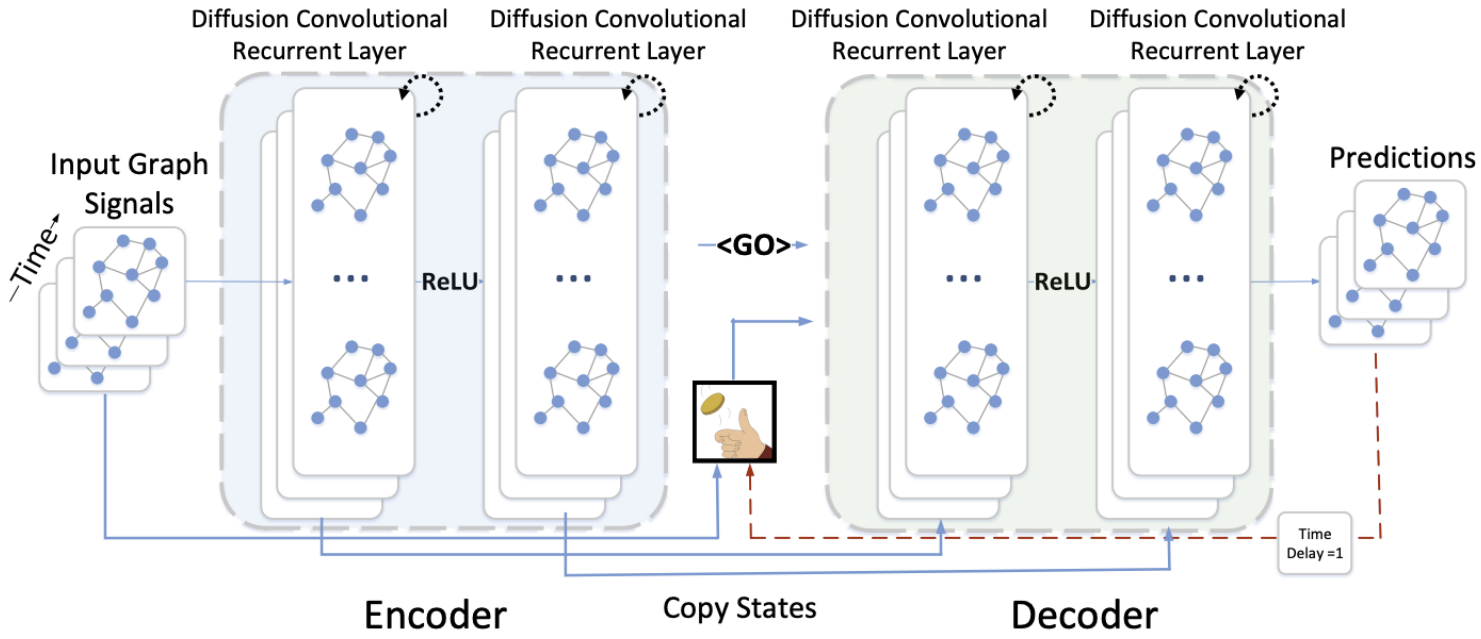
Compare temporal information and spatial information separately

A’s history: 60, 55, 50



Comparison: persistence → **own-history** model → own-history model **with graph**

DCRNN (Diffusion Convolutional Recurrent Neural Network): diffusion-GRU encoder and recurrent decoder



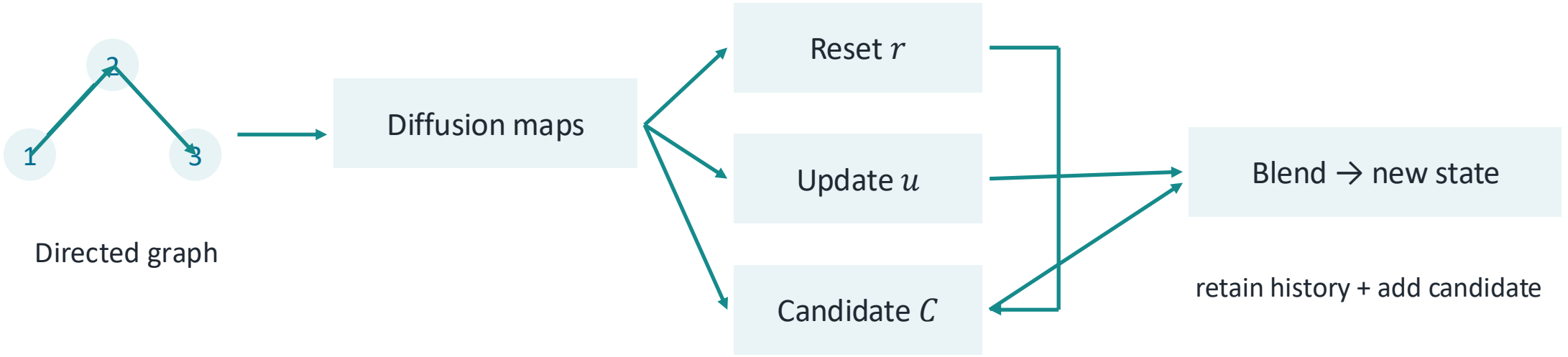
Inside each cell: forward and reverse graph diffusion replace the dense maps in GRU (Gated Recurrent Units) gates.

$$\hat{X}_{t+1:t+H} = \text{Decoder}(\text{Encoder}(X_{t-L+1:t}, \text{G}), \text{G})$$

Decoding is sequential: each predicted step becomes an input for the next step.

Li et al. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR, 2018.

Inside DCRNN: graph diffusion controls the GRU gates



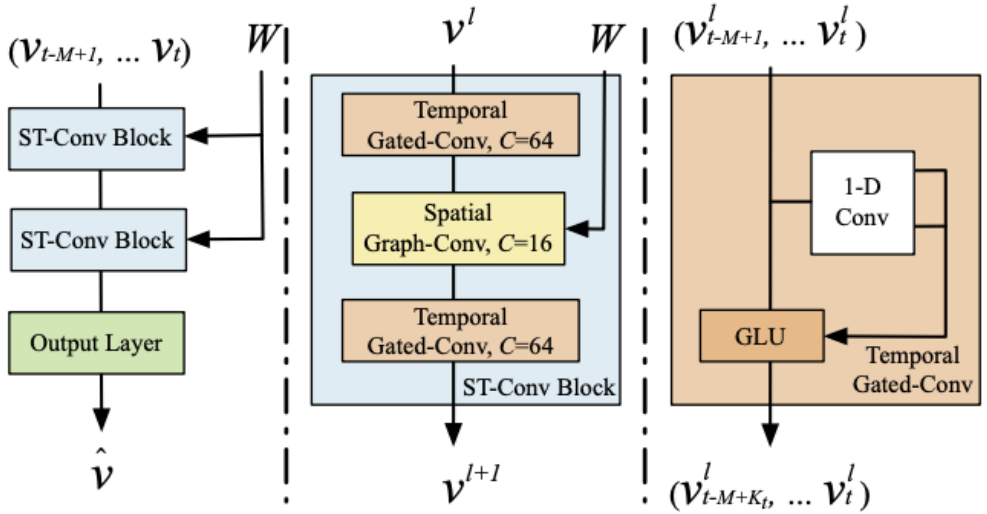
Diffusion convolutional map Θ_G: Graph convolution on directed graphs

$$r_t = \sigma(\Theta_{r,G}([X_t, H_{t-1}])); \qquad u_t = \sigma(\Theta_{\mathrm{u},G}([X_t, H_{t-1}]))$$

$$C_t = \tanh(\Theta_{\mathrm{C},G}([X_t, r_t \odot H_{t-1}])); \qquad H_t = u_t \odot H_{t-1} + (1 - u_t) \odot C_t$$

Li et al. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR, 2018.

STGCN: stacked temporal–graph–temporal blocks



Stack two blocks, then map the features to node forecasts

One block: temporal gate → graph filter → temporal gate

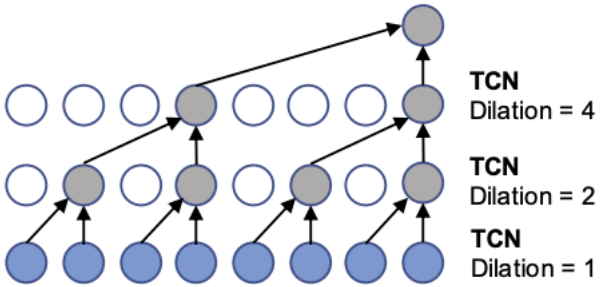
Temporal layers use Gated Linear Units (GLU);
the middle layer mixes nodes using graph filters.

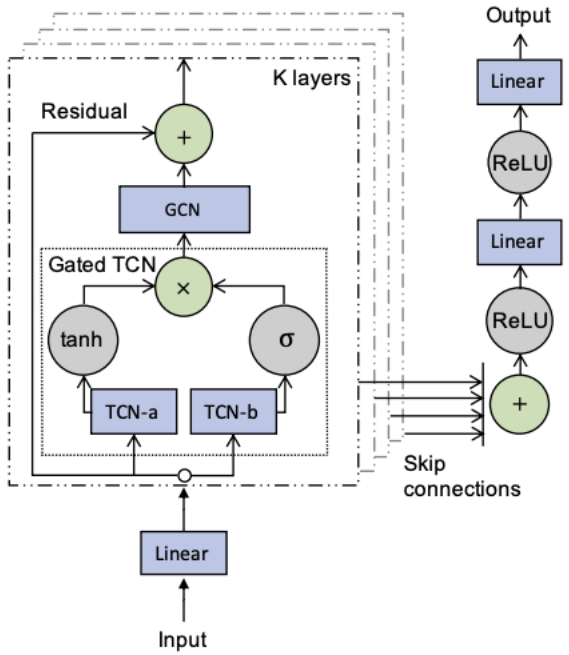
$$\text{TemporalGLU}(S) = P \odot \sigma(Q), \qquad [P, Q] = \text{Conv}_t(S)$$

$S \in \mathbb{R}^{N \times M \times C} \rightarrow P, Q \in \mathbb{R}^{N \times (M - K_t + 1) \times C'}$ N nodes · M steps · C channels · K_t kernel width

Yu, Yin & Zhu. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI, 2018.

Graph WaveNet: temporal blocks plus an adaptive graph





$$[\mathbf{X}^{(t-S):t}, G] \xrightarrow{f} \mathbf{X}^{(t+1):(t+T)}$$

Temporal: gated dilated convolution

$$h = tanh(\Theta_1 \star X) \odot \sigma(\Theta_2 \star X), \qquad (x \star f)(t) = \Sigma_s f(s) x(t - d \cdot s)$$

Dilation d skips d steps; 8 layers, d = 1, 2, 1, 2, … tanh = features, σ = gate

Spatial: given graph + learned graph

$$A_{\text{adp}} = \text{softmax}_{\text{row}}(\text{ReLU}(E_1 {E_2}^T))$$

Added to forward/backward diffusion (K = 2). $E_1, E_2 \in \mathbb{R}^{N \times 10}$, learned end to end

Output: residual + skip, one pass

Skip paths feed every block to the output; all 12 horizons predicted in one pass

Wu et al. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. IJCAI, 2019.

Interaction histories

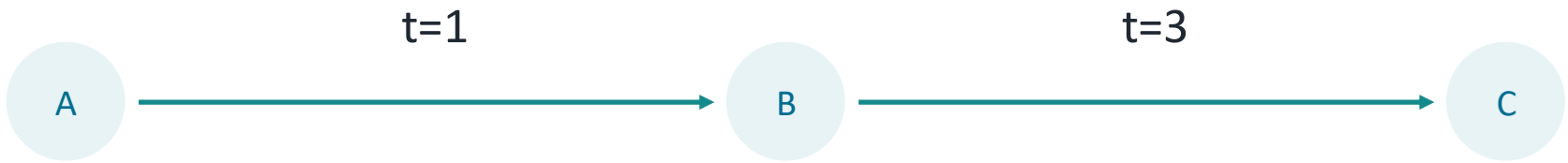
SECTION 3 · OVERVIEW

History	Use earlier events, their order, and their elapsed times.
Representations	TGAT: attention; TGN: memory; GraphMixer: mixing; DyGFormer: patches.
Prediction protocol	Score a query using its permitted history, then incorporate observed events.

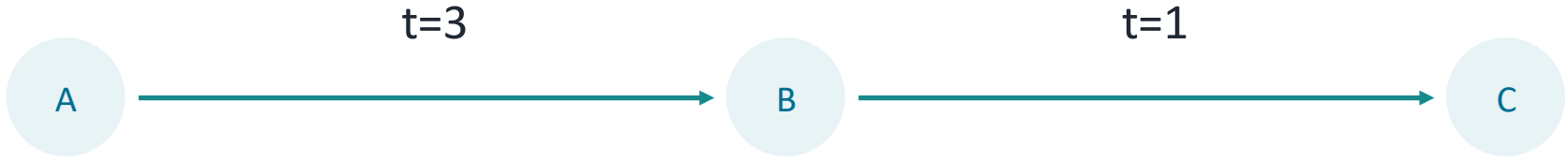
Focus: how earlier interactions become representations for prediction.

Time-respecting paths depend on event order

The same two edges can support different temporal reachability



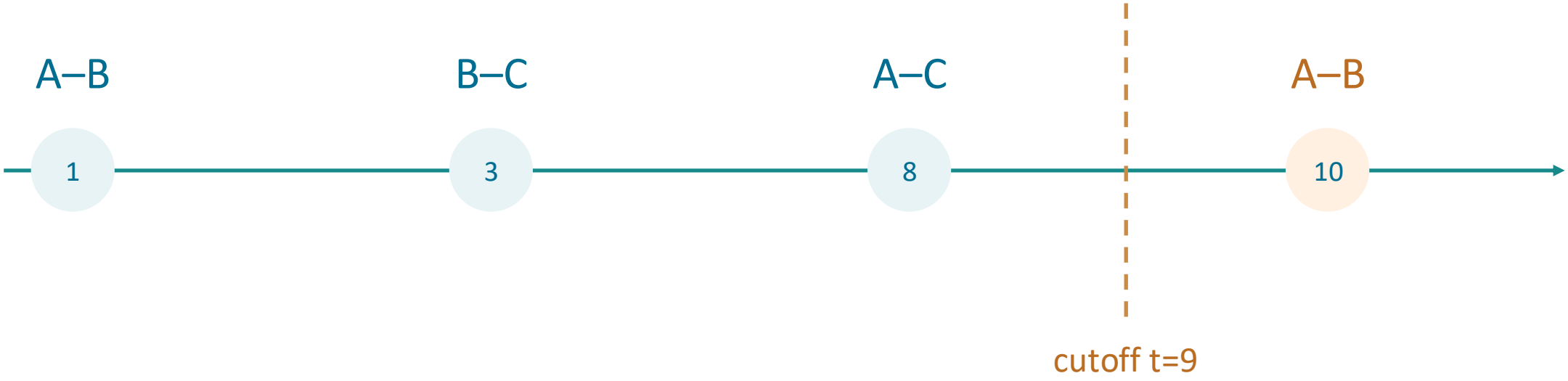
A’s information can reach C in this order



A later B–C event would be needed

Temporal neighborhoods use the query cutoff

A query at time 9 can access interactions at 1, 3, and 8



Each observation includes endpoints, time, and optional event attributes

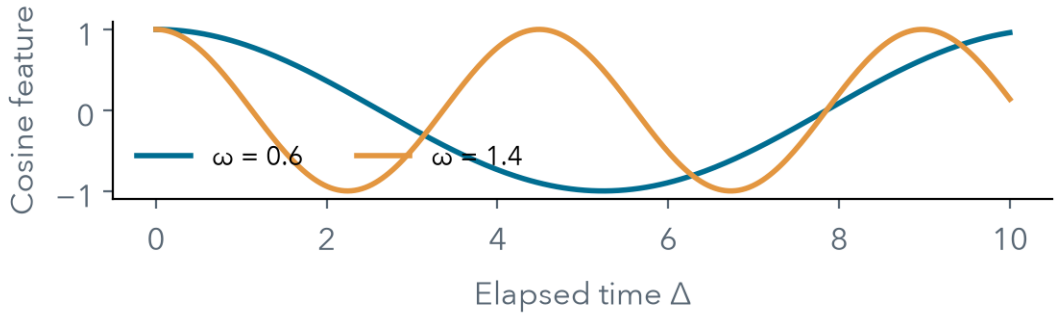
TGAT: a temporal kernel becomes a learnable time encoding

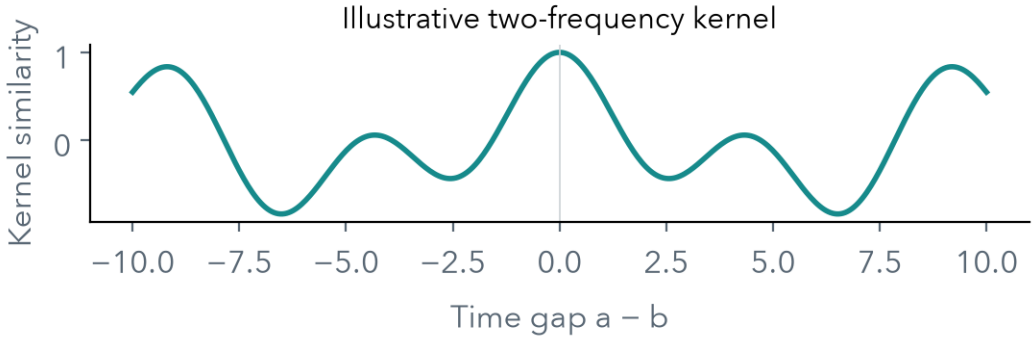
Temporal similarity depends on the time gap. Bochner's theorem gives a frequency representation.

$$K(t_1, t_2) := \langle \Phi(t_1), \Phi(t_2) \rangle \qquad K(t_1, t_2) = \psi(t_1 - t_2) = E_{\omega \sim p}[\cos(\omega(t_1 - t_2))]$$

$$\Phi(\Delta) = \frac{1}{\sqrt{m}} [\cos(\omega_1 \Delta), \sin(\omega_1 \Delta), \ldots, \cos(\omega_m \Delta), \sin(\omega_m \Delta)]$$

m frequencies give $2m$ features; TGAT learns the frequencies jointly with the prediction model.





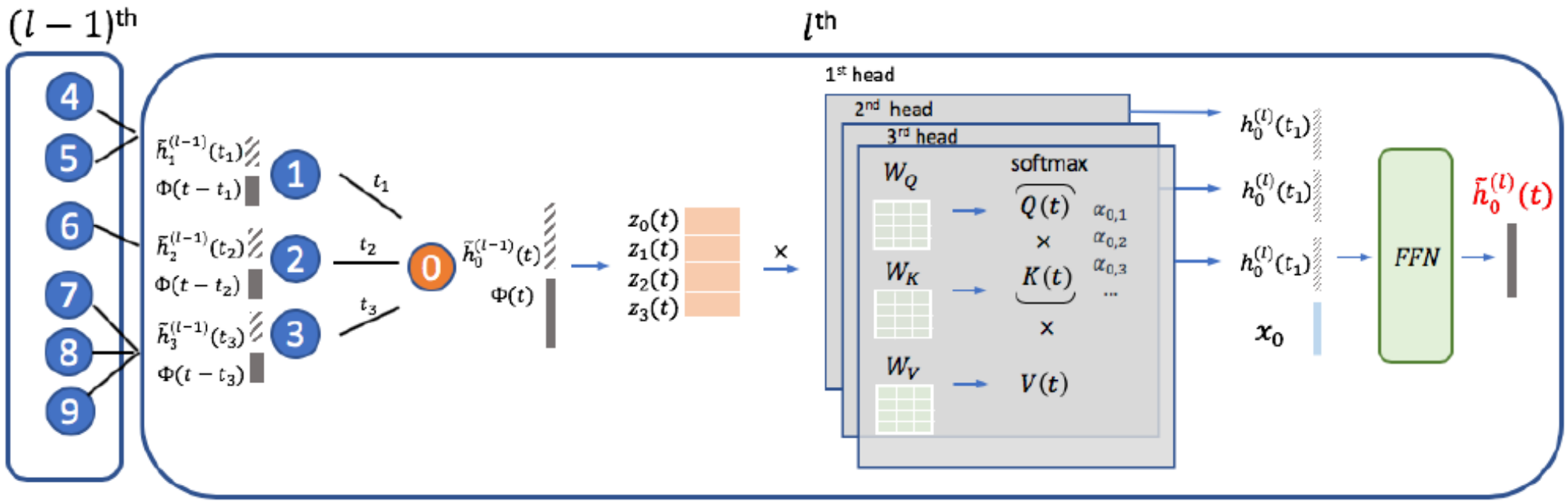
$$\Phi(a)\Phi(b)^T = \frac{1}{m} \sum_{r=1}^{m} \cos\big(\omega_r (a - b)\big) \approx K(a, b)$$

Use in TGAT: concatenate $\Phi(t - t_i)$ with neighbor features, then learn query/key/value projections.

Xu et al. Inductive Representation Learning on Temporal Graphs. ICLR, 2020.

TGAT: combine time and node features through attention

At query time t, aggregate interactions (j, t_j) with $t_j < t$.



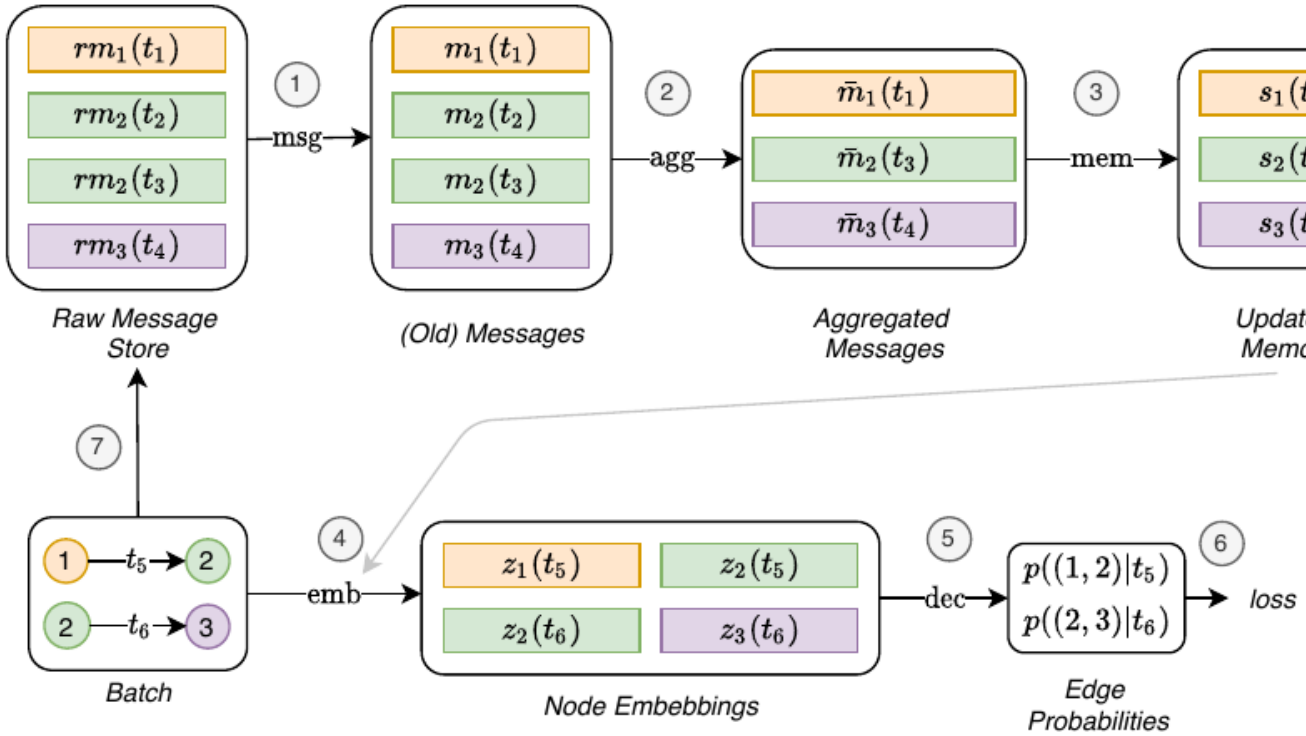
$$z_0 = [h_0{}^{(l-1)}(t) \| \Phi(0)], \qquad z_j = [h_j{}^{(l-1)}(t_j) \| \Phi(t - t_j)]$$

$$q = z_0 W_Q, \qquad k_j = z_j W_K, \qquad v_j = z_j W_V \qquad a = \text{softmax}(\frac{qK^T}{\sqrt{d_k}})V$$

Xu et al. Inductive Representation Learning on Temporal Graphs. ICLR, 2020.

TGN: persistent node memory supports temporal predictions

Memory s_i stores event history; the embedding $z_i(t)$ combines memory with query-time context.



EVENT → MESSAGE

$$m_i(\mathrm{t}) = \mathrm{msg}(s_i(\mathrm{t}^-), s_j(\mathrm{t}^-), \Delta t, e_{ij}(\mathrm{t}))$$

AGGREGATE → UPDATE MEMORY

$$s_i(t) = \mathrm{GRU}(\mathrm{agg}(\mathrm{m}_i(t_1, \ldots, t_b \leq t)), s_i(\mathrm{t}^-))$$

READ MEMORY → EMBEDDING

$$z_i(t) = \mathrm{emb}(i, t; S, H_{<t})$$

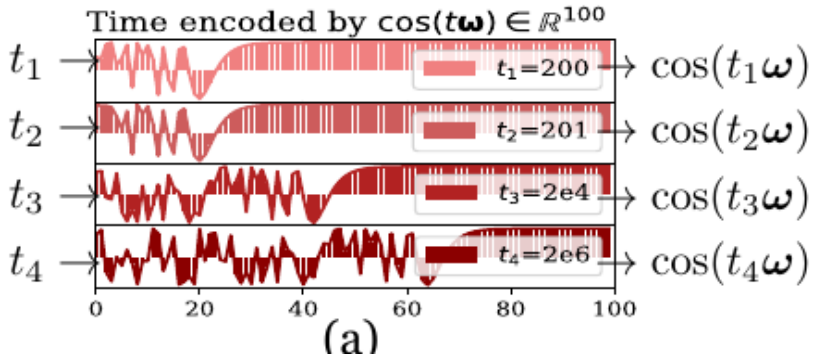
M_i: messages for node i
S: node memories
$H < t$: history before the query

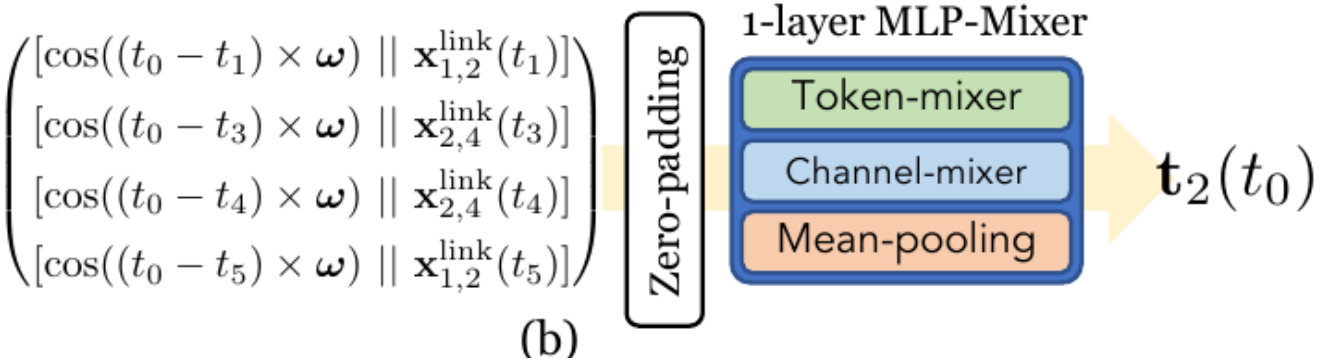
Training flow: update from earlier batches → predict the current batch → store its raw messages.

Rossi et al. Temporal Graph Networks for Deep Learning on Dynamic Graphs. ICML GRL+ Workshop, 2020.

GraphMixer: mix interaction histories and pool node features

Link encoder: fixed time features + recent link features → MLP-Mixer → a history summary.





WHAT THE MIXER DOES

Token mixing: combine information across events.
Channel mixing: combine features within each event.
Residual MLPs + layer normalization; then mean pooling.

NODE ENCODER: TARGET + RECENT NEIGHBORS

$$s_i = x_i^{\text{node}} + \text{Mean}_{j \in N_i}(x_j^{\text{node}})$$

N_i uses the time window $[t_0 - T, t_0)$.

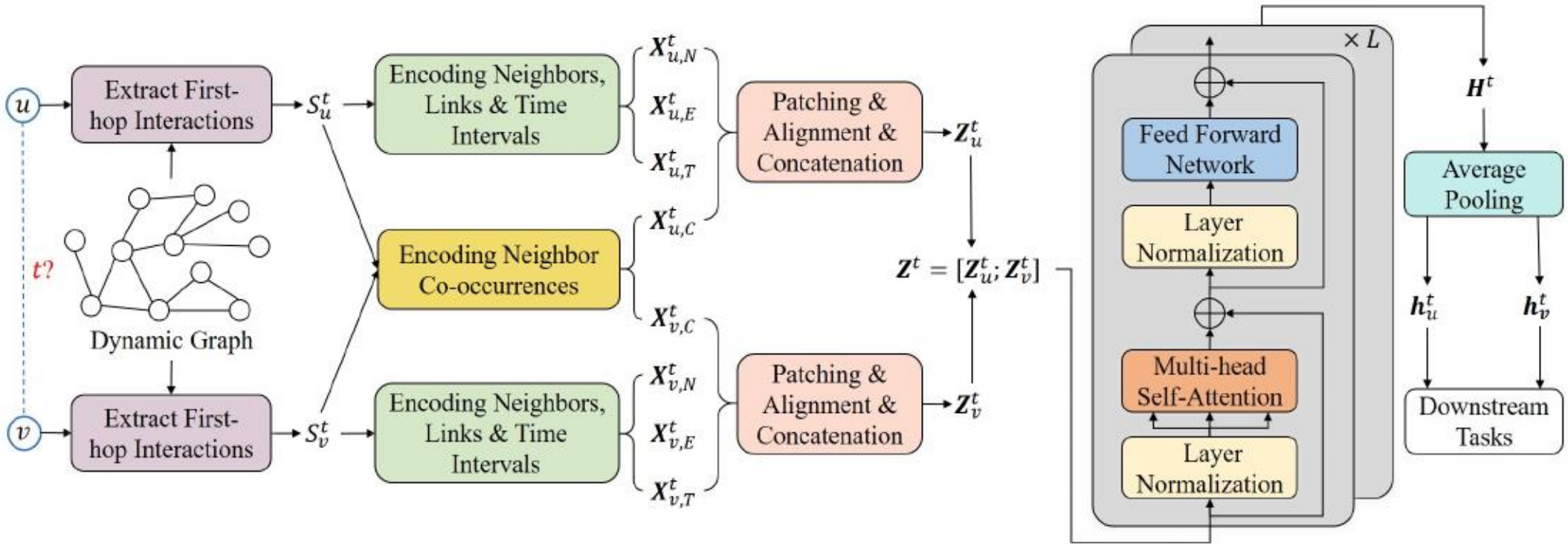
$$h_i = [s_i \| t_i], \qquad p_{ij} = \text{MLP}([h_i \| h_j])$$

Encode both candidate endpoints using history before t_0; the classifier predicts their interaction.

Cong et al. Do We Really Need Complicated Model Architectures for Temporal Networks? ICLR, 2023.

DyGFormer: connect two histories through co-occurrence and patches

Retrieve both endpoints' first-hop histories before t; encode four feature streams and patch them.



$$X_{*,C} = f(C_*[:,0]) + f(C_*[:,1])$$

$$H = \text{Transformer}([Z_u; Z_v])$$

C: counts in both histories • Z: projected patches • Pool each endpoint's output, then predict the pair.

Yu et al. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. NeurIPS, 2023.

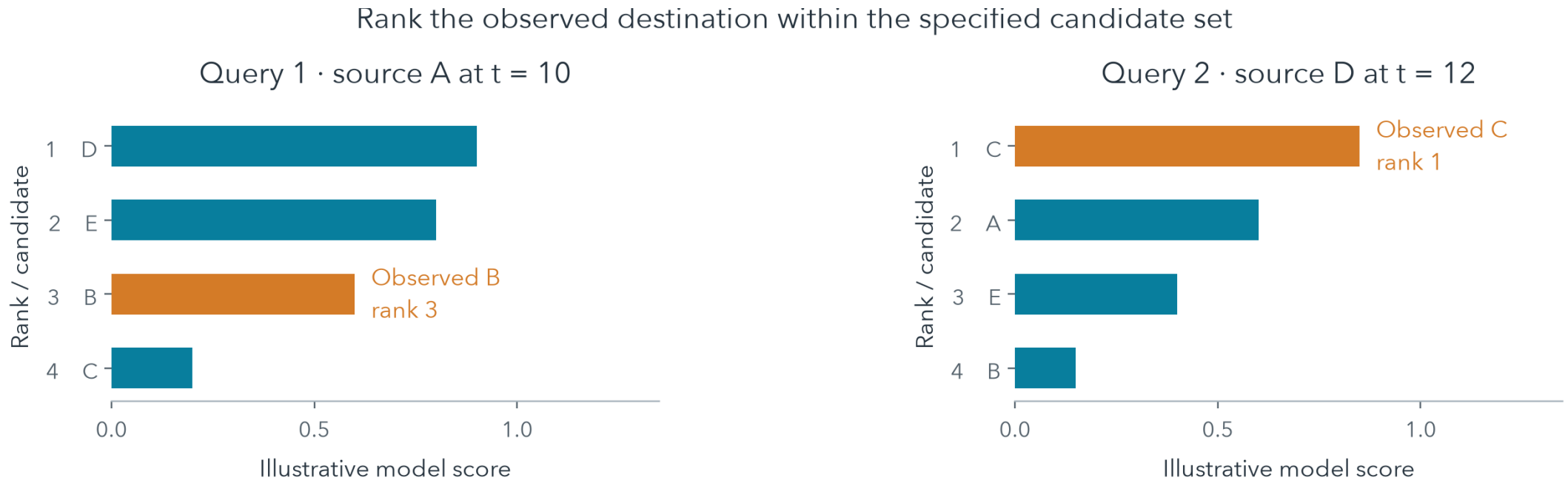
Evaluation and recent research

SECTION 4 · OVERVIEW

Comparable evaluation	State the chronological split, candidates, metrics, and update rules.
Research directions	Study temporal reasoning, graph integration, and efficient history processing.
Practical reporting	Report predictive quality alongside runtime, memory, and history budget.

Focus: what an evaluation measures and what current research explores.

Candidate selection and mean reciprocal rank



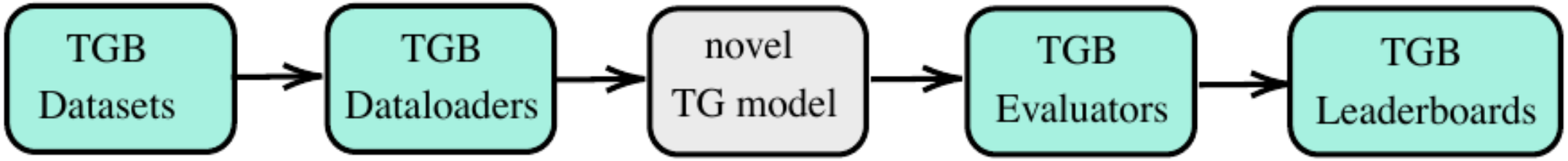
Observed destinations rank third and first. Their reciprocal ranks are 1/3 and 1.

$$\mathrm{MRR} = \frac{1}{2}\left(\frac{1}{3} + 1\right) = \frac{2}{3}$$

Illustrative observations and scores

TGB: a shared evaluation pipeline for temporal prediction

The benchmark fixes data, evaluation rules, and metrics so model comparisons are reproducible.



LINK PREDICTION

Rank an observed destination against the benchmark's candidate negatives at time t.
Metric: mean reciprocal rank (MRR).

NODE AFFINITY PREDICTION

Predict a node's changing preferences over target categories or items.
Metric: NDCG for the ranked preferences.

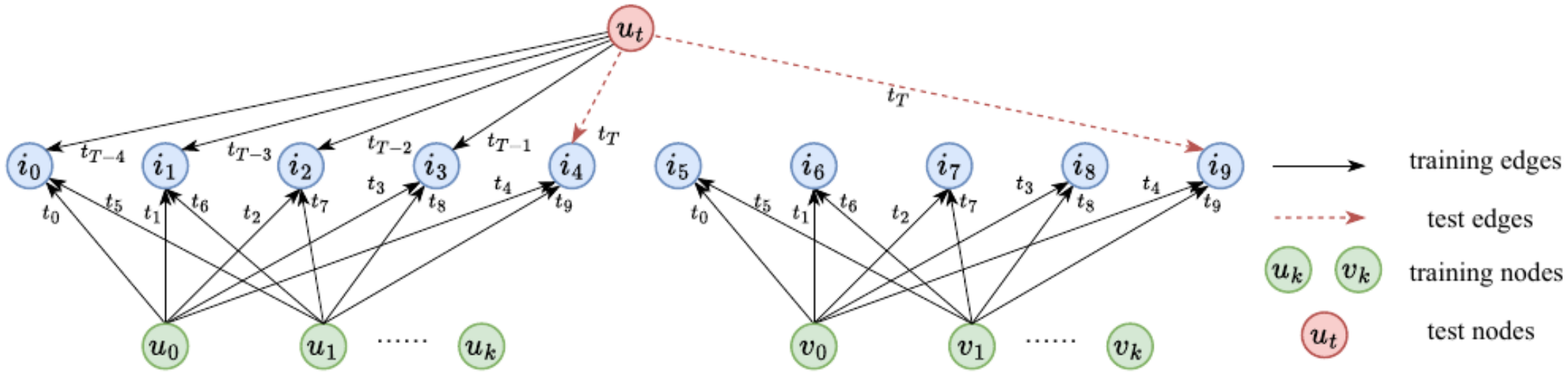
$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\mathrm{rank}_q}$$

TGB 2.0 adds temporal knowledge graphs and heterogeneous graphs with relation and node types.

Huang et al. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. NeurIPS Datasets and Benchmarks, 2023.
Gastinger et al. TGB 2.0: A Benchmark for Learning on Temporal Knowledge Graphs and Heterogeneous Graphs. NeurIPS, 2024.

TGB-Seq: transfer a learned interaction sequence to a new user

Paper's diagnostic example: two user groups follow different item sequences.



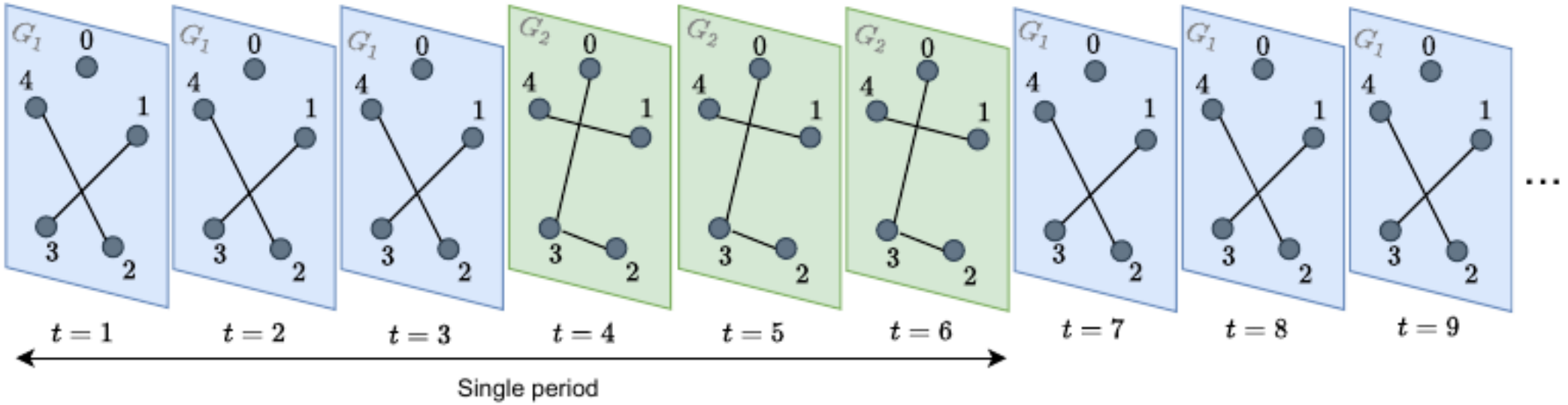
$$(i_0, i_1, i_2, i_3) \rightarrow i_4, \qquad (i_5, i_6, i_7, i_8) \rightarrow i_9$$

Test user has seen $i_0, \ldots, i_3$. Rank i_4 above i_9 using the learned sequence; both candidate pairs are new.

Yi et al. TGB-Seq Benchmark: Challenging Temporal GNNs with Complex Sequential Dynamics. ICLR, 2025.

T-GRAB: controlled tasks separate counting, memory, and delay

Periodicity example from the paper: $k = 2$ graph patterns, each repeated $n = 3$ times.



$$G_1, G_1, G_1, G_2, G_2, G_2, G_1, \ldots$$

Increase n to test counting; increase k to test pattern memory. Other tasks vary delay and graph distance.

Dizaji et al. T-GRAB: A Synthetic Diagnostic Benchmark for Learning on Temporal Graphs. KDD, 2026.

Temporal graph learning

Graph structure, history representation, and evaluation work together

1. Signals: forecast future node values
2. Events: score interactions from earlier history
3. Models: choose how spatial and temporal information combine
4. Research: test temporal skills and computational efficiency