Enhance Stability Dynamic Graphs Using Bias-Driven Graph Transformers and Memory Mechanism

Ying Wang, Miao Wang, Noah Dawang

Columbia University

Goal & Setting

We study how **biased high-dimensional graphs** affect t-SNE embeddings across rolling time windows. We compare three modules:

1. **Dynamic threshold (GAE/dyn-thr):** learn a dense affinity with a graph autoencoder (GAE), then *auto-select* a percentile q and enlarge high-dimensional distances on "weak" edges before running t-SNE. (Auto-q is self-adaptive across windows).
2. **Fixed-q:** same stretching idea but using a fixed percentile q across windows.
3. **Vanilla kNN+t-SNE:** standard pipeline without memory/GAE; kNN graph used only for component coloring and diagnostics.

We optionally align consecutive windows with an **orthogonal Procrustes** transform for visual stability, and track ρ correlations and Jaccard-like overlaps of edges over time.

Biased High-Dimensional Graph for t-SNE (what we optimize)

Given features $\mathbf{X} \in \mathbb{R}^{N\times D}$ and low-d embeddings $\mathbf{Y} \in \mathbb{R}^{N\times 2}$, t-SNE uses conditional and joint probabilities $\{p_{j|i}\}, \{P_{ij}\}$ in high-D and $\{q_{ij}\}$ in low-D:

$$d_{ij}^2 = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \qquad s_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right),$$

$$p_{j|i} = \frac{s_{ij}}{\sum_{k\neq i} s_{ik}}, \quad P_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \quad q_{ij} = \frac{\left(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k\neq \ell}\left(1+\|\mathbf{y}_k - \mathbf{y}_\ell\|^2\right)^{-1}}, \; q_{ii} = 0.$$

The KL objective $\mathcal{L} = \sum_{i\neq j} P_{ij} \log \frac{P_{ij}}{q_{ij}}$ has gradient

$$\frac{\partial \mathcal{L}}{\partial \mathbf{y}_i} = 4 \sum_{j\neq i} \left(P_{ij} - q_{ij}\right) \frac{\mathbf{y}_i - \mathbf{y}_j}{1+\|\mathbf{y}_i - \mathbf{y}_j\|^2}.$$

(Perplexity sets the entropy of each row H_i such that $2^{H_i} \approx$ Perp.) [van der Maaten & Hinton, JMLR 2008; scikit-learn TSNE API supports `metric='precomputed'` for our stretched distances]

Method 1: Memory-based + GAE + auto-q (dynamic threshold)

Memory encoder (per node). For each entity i we carry a hidden state $\mathbf{m}_i \in \mathbb{R}^{d_m}$ updated by a GRU

$$\mathbf{m}_i^{(t)} = \mathrm{GRU}\left(\mathbf{x}_i^{(t)}, \mathbf{m}_i^{(t-1)}\right),$$

and form a window feature by concatenating normalized features and memories.

Graph autoencoder (reconstruction). With encoder $f_\phi : \mathbb{R}^{\cdot} \to \mathbb{R}^{d_z}$ and decoder heads,

$$\mathbf{Z} = f_\phi(\mathbf{X}), \quad \hat{\mathbf{A}} = \sigma\left(\frac{\mathbf{Z}\mathbf{Z}^\top}{\sqrt{d_z}\,\tau}\right), \quad \hat{\mathbf{X}} = W\mathbf{Z},$$

$$\mathcal{L}_{\text{GAE}} = \mathrm{BCE}(\hat{\mathbf{A}}, \mathbf{A}_{\text{target}}) + \lambda_x \,\mathrm{MSE}(\hat{\mathbf{X}}, \mathbf{X}).$$

Auto-q selection. Let W be a dense affinity (e.g., Gaussian on features+memory). Compute a cluster compactness statistic s (e.g., std. of within-cluster weights). Set

$$q \;=\; \tfrac{1}{2} + 0.6\,\tanh\left(\frac{s-\mu}{\tau}\right) \;\in [q_{\min}, q_{\max}],$$

then the quantile threshold $\theta_q = \mathrm{Quantile}_q(W_{ij} : i \neq j, W_{ij} > 0)$. Define a *stretched* matrix of precomputed distances for t-SNE:

$$D'_{ij} = \begin{cases} D_{ij}, & W_{ij} \geq \theta_q, \\ \alpha \cdot D_{ij}, & W_{ij} < \theta_q, \end{cases} \qquad \alpha > 1.$$

We pass D' to `sklearn.manifold.TSNE` with `metric='precomputed'`.

Across-window alignment. For common nodes between two windows, solve the orthogonal Procrustes and rotate current embeddings.

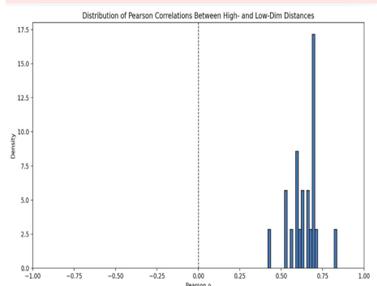
Method 2: Fixed-q hard separation

Same stretching rule as above, but θ_q uses a *fixed* percentile across windows. This isolates how much dynamics come from the adaptive gate vs. from the graph model.

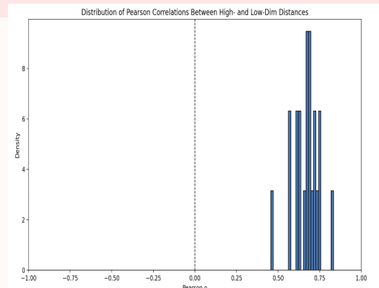
Method 3: Vanilla kNN + t-SNE (control)

Here we *do not* learn a GAE and *do not* have memory-based. We only have high-dimensional KNN plus t-SNE.

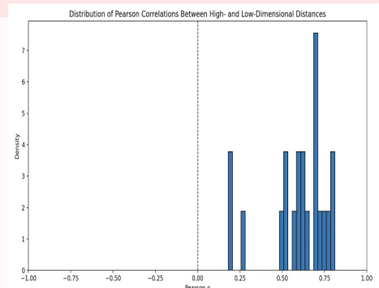
comparison



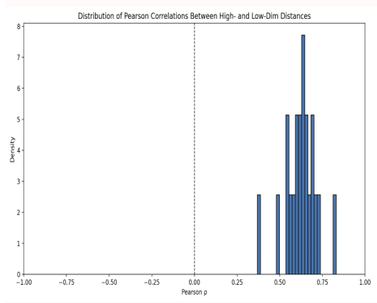
(without alignment: GAE / dynamic threshold)



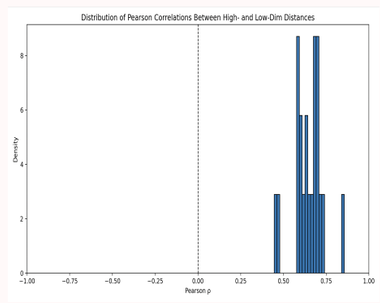
(without alignment: Fixed-q)



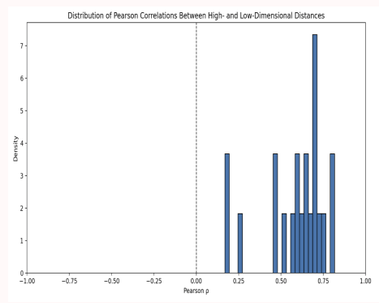
(without alignment: Vanilla kNN+t-SNE)



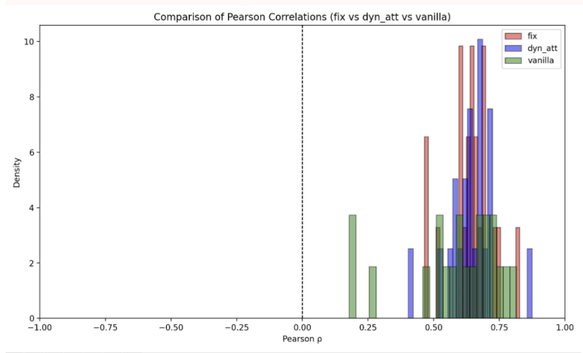
(with alignment: GAE / dynamic threshold)



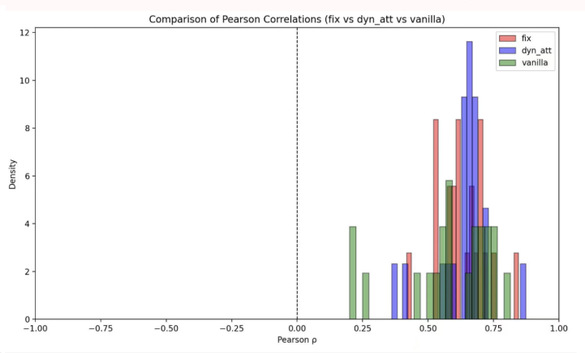
(with alignment: Fixed-q)



(with alignment: Vanilla kNN+t-SNE)



(without orthogonal alignment: t-SNE with dynamic threshold (purple), fixed threshold (red), and vanilla (green))



(with orthogonal alignment: t-SNE with dynamic threshold (purple), fixed threshold (red), and vanilla (green))

Training/rolling pipeline (what the code does)

1. GRU updates per-node memory; concatenate with current features.
2. Train/finetune GAE to get denoised dense affinity $\hat{\mathbf{A}}$.
3. Auto-q builds stretched distances D' (or fixed-q/vanilla); run t-SNE on `metric=precomputed` with warm start; optional orthogonal Procrustes.
4. Log Pearson/Jaccard vs. previous/first windows; save figures.

Metrics across windows (whole vs. subgraph)

Let $A^{(t)}$ be dense affinities (from GAE or Gaussian on features). For common nodes between $t-1$ and t:

Whole-graph Pearson: $\rho_{\text{whole}} = \mathrm{corr}\left(\{A_{ij}^{(t-1)}\}_{i<j}, \{A_{ij}^{(t)}\}_{i<j}\right)$.

Weighted Jaccard (upper triangle): $\mathrm{J_W} = \dfrac{\sum_{i<j}\min(A_{ij}^{(t-1)}, A_{ij}^{(t)})}{\sum_{i<j}\max(A_{ij}^{(t-1)}, A_{ij}^{(t)})}$ (extended Jaccard/Tanimoto for real-valued, nonnegative weights).

Subgraph versions: compute compute Pearson/Jaccard on sungraph S.

HD↔LD consistency: per window, Pearson ρ between pairwise high-D distances and 2D distances checks if t-SNE preserves neighborhood scales (rotation-invariant).

Table A (rolling Pearson/Jaccard without orthogonal alignment)

Tag	pear_prev_whole left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE	pear_prev_sub left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE	pear_first_whole left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE	pear_first_sub left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE
2005-12-31_2006-11-30	0.63220.61900.4858	0.44200.38880.3303	0.33630.34270.3278	0.34310.20190.1075
2006-12-31_2007-11-30	0.58450.58460.3761	0.42430.34280.3011	0.28010.31730.2579	0.36620.12740.2051
2007-12-31_2008-11-30	0.58350.60260.0591	0.36090.28720.1395	0.18650.26260.0388	0.25680.19490.0584
		...		

Table B (rolling Pearson/Jaccard with orthogonal alignment)

Tag	pear_prev_whole left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE	pear_prev_sub left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE	pear_first_whole left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE	pear_first_sub left = GAE/dyn-thr middle = fixed-q right = pure kNN+TSNE
2005-12-31_2006-11-30	0.57030.60180.4708	0.50750.39610.3153	0.33240.33340.3128	0.36570.19770.0925
2006-12-31_2007-11-30	0.57560.52490.3611	0.44120.35340.2861	0.29330.29320.2429	0.32080.10210.1901
		...		