C^2MIL: Synchronizing Semantic and Topological Causalities in Multiple Instance Learning for Robust and Interpretable Survival Analysis

Supplementary Material

6. Method Supplementary

6.1. Graph Transformer Architecture Description

Graph Transformer [41] consists of L stacked identical layers, each containing multi-head graph attention mechanisms, positional encoding fusion, and position-enhanced feed-forward networks. The architecture is formally defined as follows:

Input Representation. Let graph $G = (V, E)$ contain n nodes, where each node i has feature vector $h_i \in \mathbb{R}^d$, with adjacency matrix $A \in \{0, 1\}^{n \times n}$. The input feature matrix is $H^{(0)} = [h_1, \cdots, h_n]^T \in \mathbb{R}^{n \times d}$.

Relative Posit Encoding. The encoder structural relationship uses random walk probabilities:

$$\mathbf{R}_{ij} = \text{Softmax}\left(\frac{\log(P_{ij})}{\sqrt{d}}\right), \tag{18}$$

where $P \in \mathbb{R}^{n \times n}$ is the random walk transition probability matrix computed using k-step truncated values.

Multi-head Graph Attention Mechanism. For the h-th attention head in layer l:

$$\mathbf{Q}^{(h)} = \mathbf{H}^{(l)}\mathbf{W}_Q^{(h)}, \mathbf{K}^{(h)} = \mathbf{H}^{(l)}\mathbf{W}_K^{(h)}, \mathbf{V}^{(h)} = \mathbf{H}^{(l)}\mathbf{W}_V^{(h)},$$

$$\alpha_{ij}^{(h)} = \frac{\exp\left(\sigma\left(\frac{\mathbf{Q}_i^{(h)}(\mathbf{K}_j^{(h)})^\top}{\sqrt{d/H}} + \phi(A_{ij})\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\sigma\left(\frac{\mathbf{Q}_i^{(h)}(\mathbf{K}_k^{(h)})^\top}{\sqrt{d/H}} + \phi(A_{ik})\right)\right)}, \tag{19}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is an edge information mapping function, σ denotes LeakyReLU activation, and H is the number of attention heads.

Structure-Aware Attention Aggrgation.

$$\mathbf{Z}^{(h)} = \text{Softmax}(\boldsymbol{\alpha}^{(h)})\mathbf{V}^{(h)} + \mathbf{R} \circ (\boldsymbol{\alpha}^{(h)}\mathbf{V}^{(h)}), \tag{20}$$

where o denotes the Hadmard product. The multi-head output is concatenated:

$$\hat{\mathbf{H}}^{(l)} = \|_{h=1}^{H} \mathbf{Z}^{(h)}\mathbf{W}_O^{(h)}. \tag{21}$$

Residual Connection & Layer Normalization.

$$\bar{\mathbf{H}}^{(l)} = \text{LayerNorm}\left(\mathbf{H}^{(l)} + \hat{\mathbf{H}}^{(l)}\right). \tag{22}$$

Postition-Enhanced Feed-Forward Network.

$$\mathbf{H}^{(l+1)} = \text{LayerNorm}\left(\bar{\mathbf{H}}^{(l)} + \mathbf{W}_2 \cdot \text{GELU}(\mathbf{W}_1\bar{\mathbf{H}}^{(l)} + \mathbf{b}_1) + \mathbf{b}_2\right). \tag{23}$$

where $W_1 \in \mathbb{R}^{4d \times d}$ and $W_2 \in \mathbb{R}^{d \times 4d}$ are learnable parameters.

Output Layer Final node representations are obtained via K-hop neighborhood pooling:

$$\mathbf{y}_i = \sum_{k=0}^{K} \gamma_k \cdot \text{MEAN}\left(\{\mathbf{H}_j^{(L)} | j \in \mathcal{N}_k(i)\}\right), \tag{24}$$

where η_k are learnable decay coefficients.

6.2. Subgraph Sampling Pseudocodes

Algorithm 1 Subgraph Sampling

```
Input: Adjusted graph G(Ṽ, E, A); Linear MLP(·); Graph Transformer Model GT(·); subgraph(·,·) function of graph containing the mask nodes; Activation function sigmoid σ(·).
Output: Causal graph C and non-causal graph S.
G'.V = MLP(G.V)
G'.E = G.E
G'.A = [G'.V_i; G'.V_j]⟨i, j⟩ ∈ G.E
P = σ(GT(G'))
if training stage then
    sample = Bernoulli(P)
    mask = sample.detach() + P − P.detach()
    C = subgraph(G', mask)
    S = subgraph(G', 1 − mask)
else
    C = subgraph(G', P)
    S = subgraph(G', 1 − P)
```

7. Experiments Supplementary

7.1. Implement Details

A pretrained UNI is used to extract features from both thumbnails and patches. The thumbnails are derived from WSIs at $40\times$ magnification with a $30\times$ downsampling. The patches are obtained by segmenting WSIs at $40\times$ magnification into images of size 1024×1024 pixels. Before being fed into the feature extractor, both thumbnails and patches are resized to 224×224. Patches in a WSI is constructed as a graph by K nearest neighborhood (KNN) through the coordinates of patches. The proposed framework is implemented with PyTorch [29] and PyTorch Geometric [10] and all the experiments are conducted on one

NVIDIA A100 GPU with 40GB memory with batch size 16 and 100 epochs. The warm-up epoch is 2 on internal experiments and 10 on external experiments.

7.2. Results of Adaptive Cluster Number (K) Analysis

Specific value in Section 4.3

	TCGA-KIRC	TCGA-ESCA	TCGA-BLCA
$K = 2$	0.6775	0.6591	0.5905
$K = 3$	0.6920	0.6684	0.5775
$K = 4$	0.6844	0.6418	0.5762
$K = 5$	0.6795	0.6557	0.5934
$K = 6$	0.6893	0.6445	0.5812
Adaptive clusters (Ours)	**0.7078**	**0.6904**	**0.6081**
Oracle clusters	0.7131	0.6949	0.6098

Table 4. Predictive performance analysis of the adaptive optimal clustering number method compared with fixed number K of clusters.