Can We Go Beyond Visual Features? Neural Tissue Relation Modeling for Relational Graph Analysis in Non-Melanoma Skin Histology

Supplementary Material

A1. Extended Methodology

This supplementary material provides a detailed mathematical formulation of the Neural Tissue Relation Modeling (NTRM) approach presented in the main paper. We extend the methodology by providing in-depth explanations of the tissue relation module, graph construction, message passing, and spatial projection mechanisms.

A1.1. Architectural Details

The NTRM architecture extends a standard encoder-decoder framework with a novel Tissue Relation Module (TRM) that operates on intermediate features and initial segmentation predictions. While the main paper describes the overall structure, here we provide a more detailed mathematical description of each component.

The encoder extracts hierarchical features $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_5\}$ from an input image $x \in \mathbb{R}^{3\times H\times W}$, where $\mathcal{E}_i \in \mathbb{R}^{C_i\times H_i\times W_i}$ represent features at different resolutions. The decoder produces intermediate features $\mathcal{D}_1 \in \mathbb{R}^{256\times H_1\times W_1}$ and $\mathcal{D}_2 \in \mathbb{R}^{128\times H_2\times W_2}$, which serve as inputs to the TRM.

The initial segmentation head is defined as a function $\phi : \mathbb{R}^{512\times H_5\times W_5} \to \mathbb{R}^{K\times H_0\times W_0}$ that maps the bottleneck features $\mathcal{E}_5$ to a K-channel output, where K is the number of tissue classes. The output is upsampled via bilinear interpolation $\mathcal{B} : \mathbb{R}^{K\times H_0\times W_0} \to \mathbb{R}^{K\times H_2\times W_2}$ to match the resolution of $\mathcal{D}_2$ and passed through a softmax function σ to produce class probabilities:

$$p = \sigma(\mathcal{B}(\phi(\mathcal{E}_5))) \in \mathbb{R}^{K\times H_2\times W_2} \tag{1}$$

These probabilities, along with features $\mathcal{D}_2$, serve as inputs to the TRM, which we describe in detail in the following sections.

A1.2. Tissue Relation Module (TRM): Mathematical Formulation

The Tissue Relation Module consists of four main components: region proposal, node feature extraction, edge formation, and graph neural network processing. Each component plays a critical role in modeling tissue relationships.

A1.2.1. Region Proposal

The region proposal network takes initial segmentation probabilities $p \in \mathbb{R}^{K\times H_2\times W_2}$ and extracts binary masks for each tissue class. For each class $c \in \{1, 2, \ldots, K\}$, we define a binary mask $M_c \in \mathbb{R}^{H_2\times W_2}$ by thresholding the probability map:

$$M_c = \mathbb{I}[p_c > \tau] \tag{2}$$

where $\mathbb{I}[\cdot]$ is the indicator function and τ is a threshold parameter (set to 0.5 in our implementation). To enhance mask connectivity and address small gaps, we apply morphological operations approximated by max-pooling followed by thresholding:

$$M_c = \mathbb{I}[\text{maxpool}(p_c, k = 3, s = 1, p = 1) > \tau] \tag{3}$$

where k, s, and p represent the kernel size, stride, and padding of the max-pooling operation, respectively.

A1.2.2. Node Feature Extraction

Given the binary masks $\{M_1, M_2, \ldots, M_K\}$ and intermediate features $\mathcal{D}_2$, we extract node features for each tissue class. The features $\mathcal{D}_2$ are first processed through a convolutional layer $\psi : \mathbb{R}^{128\times H_2\times W_2} \to \mathbb{R}^{d\times H_2\times W_2}$ to obtain refined features $F = \psi(\mathcal{D}_2)$, where d is the node embedding dimension.

For each class c, we compute a masked representation $F_c = F \odot M_c$, where $\odot$ denotes the Hadamard product broadcast along the channel dimension. The node embedding $h_c \in \mathbb{R}^d$ is obtained via masked global average pooling:

$$h_c = \frac{\sum_{i,j} F_c(i, j)}{\sum_{i,j} M_c(i, j) + \varepsilon} \tag{4}$$

where (i, j) indexes spatial locations and ε is a small constant (10^{-6} in our implementation) to prevent division by zero when a class is absent. The complete node feature matrix is $H = [h_1, h_2, \ldots, h_K]^T \in \mathbb{R}^{K\times d}$.

A1.2.3. Edge Formation

The edge formation process constructs a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where vertices $\mathcal{V} = \{1, 2, \ldots, K\}$ represent tissue classes and edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represent spatial relationships.

We define two tissue classes i and j as spatially adjacent if their dilated masks have a non-zero intersection:

$$(i, j) \in \mathcal{E} \iff \sum_{x,y} (\text{dilate}(M_i) \odot \text{dilate}(M_j))(x, y) > 0 \tag{5}$$

where dilate is a morphological dilation operation with a 3×3 kernel. In practice, this is approximated using a max-pooling operation:

$$\text{dilate}(M) \approx \text{maxpool}(M, k = 3, s = 1, p = 1) \tag{6}$$

For each edge $(i, j) \in \mathcal{E}$, we compute an edge feature $e_{ij} \in \mathbb{R}^d$ using a two-layer MLP that operates on the concatenated node features:

$$e_{ij} = \text{MLP}([h_i \parallel h_j]) \tag{7}$$

where $\parallel$ denotes concatenation. The MLP consists of two linear layers with a ReLU activation in between:

$$\text{MLP}(x) = W_2(\text{ReLU}(W_1 x + b_1)) + b_2 \tag{8}$$

where $W_1 \in \mathbb{R}^{d\times 2d}$, $W_2 \in \mathbb{R}^{d\times d}$, $b_1 \in \mathbb{R}^d$, and $b_2 \in \mathbb{R}^d$ are learnable parameters.

A1.2.4. Graph Neural Network

The graph neural network (GNN) processes the tissue graph to refine node features. We employ an L-layer GNN with residual connections and layer normalization. The update rule for node i at layer ℓ is:

$$\tilde{h}_i^{(\ell)} = \text{LN}\left(h_i^{(\ell-1)} + \sum_{j\in\mathcal{N}(i)} \alpha_{ij} \cdot (W^{(\ell)} h_j^{(\ell-1)} \odot e_{ji})\right) \tag{9}$$

$$h_i^{(\ell)} = \text{LN}\left(\tilde{h}_i^{(\ell)} + \text{FFN}(\tilde{h}_i^{(\ell)})\right) \tag{10}$$

where $\mathcal{N}(i)$ denotes the neighborhood of node i in the graph, α_{ij} is an attention coefficient, $W^{(\ell)} \in \mathbb{R}^{d\times d}$ is a learnable weight matrix, LN denotes layer normalization, and FFN is a feed-forward network consisting of two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = W_{\text{out}}(\text{ReLU}(W_{\text{in}} x + b_{\text{in}})) + b_{\text{out}} \tag{11}$$

The attention coefficient α_{ij} is computed as:

$$\alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{k\in\mathcal{N}(i)} \exp(a_{ik})} \tag{12}$$

where $a_{ij} = \text{LeakyReLU}(q^T[W h_i^{(\ell-1)} \parallel W h_j^{(\ell-1)}])$ with $q \in \mathbb{R}^{2d}$ being a learnable attention vector. After L GNN layers, we obtain refined node embeddings $H^{(L)} = [h_1^{(L)}, h_2^{(L)}, \ldots, h_K^{(L)}]^T \in \mathbb{R}^{K\times d}$.

A1.3. Global Tissue Embeddings

In histopathology image analysis, not all tissue types are present in every image. To handle cases where certain tissue classes are absent, we introduce global tissue embeddings. For each class c that is absent in an image (i.e., $\sum_{i,j} M_c(i, j) = 0$), we replace its node embedding h_c with a learned global embedding $g_c \in \mathbb{R}^d$:

$$h_c = \begin{cases} h_c & \text{if } \sum_{i,j} M_c(i, j) > 0 \\ g_c & \text{otherwise} \end{cases} \tag{13}$$

The global embeddings $\{g_1, g_2, \ldots, g_K\}$ are learnable parameters that capture prior knowledge about tissue types and their relationships. They are initialized from a normal distribution $\mathcal{N}(0, 0.02)$ and updated during training.

The presence of global embeddings allows the model to reason about potential tissue interactions even when certain tissues are not present in the current image. This is particularly important for rare tissue types or when analyzing small image patches where not all tissues can be observed simultaneously. Mathematically, the global embeddings modify the node feature matrix H by replacing absent tissue embeddings:

$$H' = H \odot P + G \odot (1 - P) \tag{14}$$

where $P \in \{0, 1\}^{K\times d}$ is a binary presence matrix with $P_c = \mathbf{1}$ if tissue c is present and $P_c = \mathbf{0}$ otherwise, and $G = [g_1, g_2, \ldots, g_K]^T \in \mathbb{R}^{K\times d}$ is the matrix of global embeddings.

A1.4. Feature Projection and Fusion

After obtaining refined node embeddings $H^{(L)}$, we project them back to the spatial domain to produce enhanced features. For each tissue class c, we broadcast its embedding $h_c^{(L)}$ to the corresponding mask region:

$$S_c = h_c^{(L)} \otimes M_c \in \mathbb{R}^{d\times H_2\times W_2} \tag{15}$$

where $\otimes$ denotes the outer product that broadcasts the embedding to all spatial locations where $M_c = 1$. The enhanced spatial tensor S is obtained by summing over all classes:

$$S = \sum_{c=1}^{K} S_c \tag{16}$$

A 1×1 convolution with batch normalization is applied to project S to match the channel dimension of $\mathcal{D}_2$:

$$S' = \text{BN}(\text{Conv}_{1\times 1}(S)) \tag{17}$$

The enhanced features are fused with the original features via residual addition:

$$\mathcal{D}_2' = \mathcal{D}_2 + S' \tag{18}$$

The fused features $\mathcal{D}_2'$ are then passed through the remaining decoder stages to produce the final segmentation.

A1.5. Loss Function Analysis

We train the model using a composite loss function that combines predictions from both the initial and final segmentation stages:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ce}}(\hat{y}_{\text{final}}, y) + \lambda \mathcal{L}_{\text{ce}}(\hat{y}_{\text{init}}, y) \tag{19}$$

where λ (set to 0.4 in our implementation) balances the auxiliary loss from the initial segmentation, and y denotes the ground truth segmentation map. The cross-entropy loss $\mathcal{L}_{\text{ce}}$ is weighted to account for class imbalance:

$$\mathcal{L}_{\text{ce}}(\hat{y}, y) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{K} w_c \cdot y_{n,c} \log(\hat{y}_{n,c}) \tag{20}$$

where N is the number of pixels, w_c is the weight for class c, and $y_{n,c}$ and $\hat{y}_{n,c}$ are the ground truth and predicted probabilities for pixel n and class c, respectively. The class weights w_c are computed dynamically for each batch based on the frequency of each class:

$$w_c = \frac{N}{\sum_{n=1}^{N} \mathbb{I}[y_n = c] \cdot K} \tag{21}$$

This formulation assigns higher weights to less frequent classes, helping the model learn from imbalanced data. The dual-stage loss serves multiple purposes:

1. It provides direct supervision to the initial segmentation head, ensuring meaningful features for the TRM.
2. It creates an auxiliary gradient path that facilitates training of the deeper layers.
3. It regularizes the model by encouraging consistent predictions at different stages.

A2. Algorithmic Analysis

A2.1. Edge Weight Computation

The edge weights in the tissue graph represent the strength of the relationship between different tissue types. We compute these weights based on both spatial adjacency and feature similarity.

For each pair of adjacent tissue classes $(i, j) \in \mathcal{E}$, we define the edge weight w_{ij} as:

$$w_{ij} = \sigma(e_{ij}^T W e_{ij}) \tag{22}$$

where e_{ij} is the edge feature, $W \in \mathbb{R}^{d \times d}$ is a learnable weight matrix, and σ is the sigmoid function that maps the weight to the range $(0, 1)$. To capture the asymmetric nature of tissue relationships (e.g., tumor cells might influence surrounding tissues differently than vice versa), we allow $w_{ij} \neq w_{ji}$ by computing them separately.

A2.2. Handling Boundary Cases

Histopathology images often contain tissue boundaries where multiple tissue types meet. These regions require special attention in the graph construction process. We employ two strategies to handle boundary cases:

1. *Soft Mask Assignment*: Instead of using hard binary masks, we can use soft masks based on the probability values p_c. This allows a pixel to contribute to multiple tissue nodes proportionally to its class probabilities.

2. *Boundary-Aware Edge Features*: For edges that connect tissues with a significant boundary, we compute additional boundary-specific features:

$$b_{ij} = \frac{\sum_{x,y}(M_i \odot \text{dilate}(M_j))(x, y)}{\sum_{x,y} M_i(x, y)} \tag{23}$$

This boundary ratio b_{ij} represents the fraction of tissue i that is adjacent to tissue j. It is incorporated into the edge feature computation:

$$e_{ij} = \text{MLP}([h_i \parallel h_j \parallel b_{ij} \parallel b_{ji}]) \tag{24}$$

By explicitly modeling boundary information, the graph can better capture tissue interactions at interface regions, which are often clinically significant (e.g., tumor invasions).

A2.3. Forward Pass Algorithm

Algorithm 1 outlines the complete forward pass of the NTRM model.

A3. Optimization Details

A3.1. Learning Rate Scheduling

We employ a learning rate scheduling strategy to improve convergence, which reduces the learning rate when the validation loss plateaus:

$$\text{lr}_{\text{new}} = \begin{cases} \text{lr}_{\text{old}} \cdot \gamma & \text{if no improvement for patience epochs} \\ \text{lr}_{\text{old}} & \text{otherwise} \end{cases} \tag{25}$$

where $\gamma = 0.5$ is the reduction factor and patience = 5 epochs. This scheduling strategy allows the model to make large updates initially and then fine-tune as training progresses.

A3.2. Weight Initialization

Proper weight initialization is crucial for training deep networks. We employ the following initialization schemes:

- *Convolutional Layers*: Weights are initialized using Kaiming initialization with a normal distribution:

$$W \sim \mathcal{N}(0, \sqrt{\frac{2}{(1 + a^2) \cdot \text{fan\_in}}}) \tag{26}$$

Algorithm 1 NTRM Forward Pass

Require: Input image $x \in \mathbb{R}^{3\times H\times W}$
Ensure: Final segmentation $\hat{y}_{\text{final}} \in \mathbb{R}^{K\times H\times W}$
$\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_5\} \leftarrow \text{Encoder}(x)$ ▷ Extract encoder features
$\mathcal{D}_1 \leftarrow \text{Decoder1}(\mathcal{E}_5, \mathcal{E}_4)$ ▷ First decoder block
$\mathcal{D}_2 \leftarrow \text{Decoder2}(\mathcal{D}_1, \mathcal{E}_3)$ ▷ Second decoder block
$\hat{y}_{\text{init}} \leftarrow \phi(\mathcal{E}_5)$ ▷ Initial segmentation
$p \leftarrow \sigma(\mathcal{B}(\hat{y}_{\text{init}}))$ ▷ Upsample and apply softmax
$\mathcal{G}, H, E \leftarrow \text{ConstructGraph}(p, \mathcal{D}_2)$ ▷ Construct tissue graph
for $\ell = 1$ to L **do**
  for $i = 1$ to K **do**
    $\tilde{h}_i^{(\ell)} \leftarrow h_i^{(\ell-1)} + \sum_{j\in\mathcal{N}(i)} \alpha_{ij} \cdot (W^{(\ell)} h_j^{(\ell-1)} \odot e_{ji})$
    $\tilde{h}_i^{(\ell)} \leftarrow \text{LayerNorm}(\tilde{h}_i^{(\ell)})$
    $h_i^{(\ell)} \leftarrow \tilde{h}_i^{(\ell)} + \text{FFN}(\tilde{h}_i^{(\ell)})$
    $h_i^{(\ell)} \leftarrow \text{LayerNorm}(h_i^{(\ell)})$
$S \leftarrow \sum_{c=1}^{K} h_c^{(L)} \otimes M_c$ ▷ Project to spatial domain
$S' \leftarrow \text{BN}(\text{Conv}_{1\times1}(S))$ ▷ Project to match channels
$\mathcal{D}_2' \leftarrow \mathcal{D}_2 + S'$ ▷ Fuse features
$\mathcal{D}_3 \leftarrow \text{Decoder3}(\mathcal{D}_2', \mathcal{E}_2)$ ▷ Third decoder block
$\mathcal{D}_4 \leftarrow \text{Decoder4}(\mathcal{D}_3, \mathcal{E}_1)$ ▷ Fourth decoder block
$\mathcal{D}_5 \leftarrow \text{Decoder5}(\mathcal{D}_4)$ ▷ Fifth decoder block
$\hat{y}_{\text{final}} \leftarrow \text{FinalConv}(\mathcal{D}_5)$ ▷ Final segmentation
return $\hat{y}_{\text{final}}, \hat{y}_{\text{init}}$

where a is the negative slope of the leaky ReLU (or $a = 0$ for standard ReLU) and fan_in is the number of input units.

- *Graph Neural Network*: Edge weights and attention parameters are initialized from a uniform distribution:

$$W \sim \mathcal{U}(-\sqrt{\frac{6}{d_{\text{in}} + d_{\text{out}}}}, \sqrt{\frac{6}{d_{\text{in}} + d_{\text{out}}}}) \tag{27}$$

where d_{in} and d_{out} are the input and output dimensions.

- *Global Tissue Embeddings*: Initialized from a normal distribution with $\mu = 0$ and $\sigma = 0.02$:

$$g_c \sim \mathcal{N}(0, 0.02) \tag{28}$$

A3.3. Normalization Configurations

Batch normalization is applied after convolutional layers to stabilize training. For the TRM, we use layer normalization instead of batch normalization for the graph neural network, as it is more suitable for graph-structured data where the batch size may be variable. The batch normalization layers use the following configuration: momentum of 0.1, epsilon of 1×10^{-5}, and affine parameters enabled. During inference, we use the running statistics accumulated during training for batch normalization.