Mathematical Formulations for Hybrid GNN-RNN Model

Cardiomyocyte Differentiation Prediction

Tumo Kgabeng Lulu Wang Harry Ngwangwa
Thanyani Pandelani

September 25, 2025

Contents

1 Introduction

This document presents the complete mathematical formulation of the Hybrid Graph Neural Network and Recurrent Neural Network (GNN-RNN) model for predicting cardiomyocyte differentiation trajectories. The model integrates spatial transcriptomics data through GNN processing and temporal gene expression patterns via RNN processing, achieving superior classification performance through multimodal fusion strategies.

2 Graph Neural Network (GNN) Component

2.1 Graph Attention Network (GAT) Node Update

The node representation update in the Graph Attention Network follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{1}$$

where:

- $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ is the feature vector of node i at layer l
- $\mathcal{N}(i)$ denotes the spatial neighborhood of node i
- $\alpha_{ij}^{(l)}$ is the attention coefficient between nodes i and j at layer l
- $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ is the learnable weight matrix
- $\sigma(\cdot)$ is the activation function (LeakyReLU)

2.2 Attention Mechanism

The attention coefficients are computed using the attention mechanism:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]\right)\right)} \tag{2}$$

where:

- $\mathbf{a} \in \mathbb{R}^{2d'}$ is the learnable attention parameter vector
- $\|$ denotes the concatenation operation
- $\text{LeakyReLU}(x) = \max(0.01x, x)$ is the LeakyReLU activation function

2.3 Spatial Graph Construction

The spatial adjacency matrix is constructed based on physical proximity:

$$A_{ij} = \begin{cases} 1 & \text{if } d(\mathbf{s}_i, \mathbf{s}_j) \leq \tau \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where:

- $d(\mathbf{s}_i, \mathbf{s}_j) = \|\mathbf{s}_i - \mathbf{s}_j\|_2$ is the Euclidean distance
- $\mathbf{s}_i, \mathbf{s}_j \in \mathbb{R}^2$ are spatial coordinates
- $\tau = 55\mu$m is the distance threshold for 10X Visium technology

3 Recurrent Neural Network (RNN) Component

3.1 Bidirectional LSTM Architecture

The BiLSTM processes temporal sequences in both forward and backward directions. The LSTM cell operations are defined by:

3.1.1 Gate Computations

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad \text{(Forget Gate)} \tag{4}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad \text{(Input Gate)} \tag{5}$$
$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \quad \text{(Candidate Values)} \tag{6}$$
$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t \quad \text{(Cell State)} \tag{7}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \quad \text{(Output Gate)} \tag{8}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \quad \text{(Hidden State)} \tag{9}$$

where:

- $\mathbf{W}_*, \mathbf{b}_*$ are learnable parameters for each gate
- $\sigma(\cdot)$ is the sigmoid function
- $\odot$ denotes element-wise multiplication
- $[\cdot, \cdot]$ represents concatenation

3.1.2 Bidirectional Processing

$$\overrightarrow{\mathbf{h}}_t = \text{LSTM}_{\text{forward}}(\mathbf{x}_t, \overrightarrow{\mathbf{h}}_{t-1}) \tag{10}$$
$$\overleftarrow{\mathbf{h}}_t = \text{LSTM}_{\text{backward}}(\mathbf{x}_t, \overleftarrow{\mathbf{h}}_{t+1}) \tag{11}$$
$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] \tag{12}$$

4 Hybrid Fusion Strategies

4.1 Early Fusion (Concatenation)

The concatenation fusion strategy combines embeddings at the feature level:

$$\mathbf{h}_{\text{fused}} = [\mathbf{h}_{\text{GNN}}; \mathbf{h}_{\text{RNN}}] \tag{13}$$

$$\mathbf{y} = \text{MLP}(\mathbf{h}_{\text{fused}}) \tag{14}$$

where $\text{MLP}(\cdot)$ represents a multi-layer perceptron classifier.

4.2 Attention Fusion (Dynamic Weighting)

The attention-based fusion learns dynamic weights for each modality:

$$\boldsymbol{\alpha} = \text{softmax}(\text{MLP}([\mathbf{h}_{\text{GNN}}; \mathbf{h}_{\text{RNN}}])) \tag{15}$$

$$\mathbf{h}_{\text{fused}} = \alpha_{\text{GNN}} \cdot \mathbf{W}_{\text{GNN}}\mathbf{h}_{\text{GNN}} + \alpha_{\text{RNN}} \cdot \mathbf{W}_{\text{RNN}}\mathbf{h}_{\text{RNN}} \tag{16}$$

where:

- $\boldsymbol{\alpha} = [\alpha_{\text{GNN}}, \alpha_{\text{RNN}}]^T$ with $\alpha_{\text{GNN}} + \alpha_{\text{RNN}} = 1$
- $\mathbf{W}_{\text{GNN}}, \mathbf{W}_{\text{RNN}}$ are learnable projection matrices

4.3 Late Fusion (Ensemble)

The ensemble fusion combines predictions from separate modality-specific classifiers:

$$\mathbf{y}_{\text{GNN}} = \text{MLP}_{\text{GNN}}(\mathbf{h}_{\text{GNN}}) \tag{17}$$

$$\mathbf{y}_{\text{RNN}} = \text{MLP}_{\text{RNN}}(\mathbf{h}_{\text{RNN}}) \tag{18}$$

$$\mathbf{y}_{\text{final}} = \lambda \cdot \mathbf{y}_{\text{GNN}} + (1 - \lambda) \cdot \mathbf{y}_{\text{RNN}} \tag{19}$$

where $\lambda = \sigma(w)$ is a learnable ensemble weight parameter.

5 Loss Functions and Optimization

5.1 Weighted Cross-Entropy Loss

To handle class imbalance, we employ weighted cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \log(\text{softmax}(\mathbf{f}(\mathbf{x}_i))_{y_i}) \tag{20}$$

where:

- w_{y_i} is the class weight for the true class y_i
- $\mathbf{f}(\mathbf{x}_i)$ is the model output for sample i
- N is the total number of samples

5.2 Focal Loss

For severe class imbalance, we implement focal loss:

$$\mathcal{L}_{\text{focal}} = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{21}$$

where:

- p_t is the predicted probability for the true class
- α_t is the class weighting factor
- $\gamma = 2.0$ is the focusing parameter

5.3 Monte Carlo Dropout for Uncertainty Quantification

For uncertainty estimation, we employ Monte Carlo dropout:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} \text{softmax}(\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}_t)) \tag{22}$$

$$\text{Uncertainty} = -\sum_{c=1}^{C} p(y = c|\mathbf{x}) \log p(y = c|\mathbf{x}) \tag{23}$$

where:

- T is the number of Monte Carlo samples
- $\boldsymbol{\theta}_t$ represents model parameters with dropout enabled
- C is the number of classes

6 Evaluation Metrics

6.1 Classification Accuracy

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[y_i = \hat{y}_i] \tag{24}$$

where $\mathbb{1}[\cdot]$ is the indicator function.

6.2 F1-Score

The weighted F1-score is computed as:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \tag{25}$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \tag{26}$$

$$\text{F1}_c = \frac{2 \times \text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{27}$$

$$\text{F1}_{\text{weighted}} = \sum_{c=1}^{C} \frac{n_c}{N} \text{F1}_c \tag{28}$$

where n_c is the number of samples in class c.

6.3 Area Under the ROC Curve (AUC-ROC)

For multi-class classification, we use macro-averaged AUC:

$$\text{AUC}_{\text{macro}} = \frac{1}{C} \sum_{c=1}^{C} \text{AUC}_c \tag{29}$$

where AUC_c is the AUC for class c in a one-vs-rest setting.

7 Activation Functions

7.1 Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{30}$$

7.2 Softmax Function

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{31}$$

7.3 LeakyReLU

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases} \tag{32}$$

7.4 Hyperbolic Tangent

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{33}$$

8 Data Preprocessing

8.1 Min-Max Normalization

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{34}$$

8.2 Standard Scaling (Z-score)

$$x_{\text{scaled}} = \frac{x - \mu}{\sigma} \tag{35}$$

where μ is the mean and σ is the standard deviation.

8.3 Log Transformation

$$x_{\log} = \log(x + 1) \tag{36}$$

9 Performance Results

9.1 Component Performance

- GNN Spatial Component: 36.2% accuracy
- RNN Temporal Component: 93.75% accuracy

9.2 Hybrid Model Performance

- Best Fusion Strategy: Attention-based fusion
- Final Accuracy: **96.67%**
- F1-Score: 0.9654
- Improvement over RNN: +2.92%

9.3 Uncertainty Quantification

- Average Confidence: 0.9234
- Predictive Entropy: 0.1876
- Monte Carlo Samples: 100

10 Summary

This document presents the complete mathematical framework underlying the Hybrid GNN-RNN model for cardiomyocyte differentiation prediction. The integration of spatial graph neural networks and temporal recurrent neural networks through attention-based fusion achieves state-of-the-art performance (96.67% accuracy) while providing uncertainty quantification and biological interpretability.

The mathematical formulations demonstrate how spatial relationships in tissue architecture (captured by GAT) and temporal gene expression dynamics (modeled by BiLSTM) can be effectively combined through learnable attention mechanisms to predict complex biological processes with high accuracy and reliability.

References

[1] Kuppe, C., Ramirez Flores, R. O., Li, Z., Hayat, S., Levinson, R. T., Liao, X., Hannani, M. T., Tanevski, J., Wünnemann, F., Nagai, J. S., Halder, M., Schumacher, D., Menzel, S., Schäfer, G., Hoeft, K., Cheng, M., Ziegler, S., Zhang, X., Peisker, F., & Kramann, R. (2022). Spatial multi-omic map of human myocardial infarction. *Nature*, 608, 766–777. https://doi.org/10.1038/s41586-022-05060-x

[2] Elorbany, R., Popp, J. M., Rhodes, K., Strober, B. J., Barr, K., Qi, G., Gilad, Y., & Battle, A. (2022). Single-cell sequencing reveals lineage-specific dynamic genetic regulation of gene expression during human cardiomyocyte differentiation. *PLoS Genetics*, 18(1). https://doi.org/10.1371/journal.pgen.1009666

[3] Kgabeng, T., Wang, L., Ngwangwa, H., & Pandelani, T. (2025). Hybrid GNN-RNN Model for Cardiomyocyte Differentiation Prediction. GitHub repository: https://github.com/Tumo505/HybridGnnRnn