# Supplementary Notes for "CoxFormer enables spatial omics inference with multimodal generative modeling"

Yiyang Yang[1,3†], Xu Liao[1†], Haoyu Zhang[1†], Yida Wu[1], Xiaobo Sun[2], Yao Wang[3*], Tianshu Yu[1*], and Jin Liu[1*]

[1]School of Data Science, The Chinese University of Hong Kong-Shenzhen, Shenzhen, China,

[2]Department of Human Genetics, Emory University School of Medicine, Atlanta, USA.

[3]School of Management, Xi'an Jiaotong University, Xi'an, China,

[†]Equal contributions.

[*]Corresponding author. Email: yao.s.wang@gmail.com, yutianshu@cuhk.edu.cn, liujinlab@cuhk.edu.cn

## 1 Implementation details for co-expression completion

As illustrated in Supplementary Fig. S21a, CoxFormer is trained to predict gene co-expression values on a global gene correlation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Correlation information is incorporated implicitly through correlation-guided graph construction: for each gene node $i$, we retain only the top-$K$ most correlated partners according to $(\mathbf{C}_{\text{cor}})_{i,j}$ and create directed edges $\mathcal{E} = \{(i,j) \mid j \in \mathcal{N}_K(i)\}$. This top-$K$ filtering yields a candidate neighborhood that is enriched for high-correlation relations; therefore, subsequent neighbor sampling can be viewed as an *indirect* form of correlation-guided sampling (the sampling itself is uniform over the remaining adjacency). For efficiency, optimization is performed on sampled computation subgraphs. Concretely, for each mini-batch of target edges $\mathcal{B}_E = \{(i_n, j_n)\}_{n=1}^{|\mathcal{B}_E|}$, we define the seed node set as the union of all edge endpoints $\mathcal{S} = \bigcup_{(i,j)\in\mathcal{B}_E}\{i,j\}$, and apply a two-layer neighborhood sampler (PyG `NeighborLoader`) to obtain a 2-hop computation subgraph with fanouts $[m_1, m_2] = [15, 10]$. The sampled subgraph thus contains both first-order context (neighbors of the endpoints) and second-order context (neighbors-of-neighbors), while keeping memory and computation bounded. We additionally store the correlation score $(\mathbf{C}_{\text{cor}})_{i,j}$ for analysis and reproducibility. With each subgraph, CoxFormer updates node representations using a $L$-layer GNN, producing node embeddings $\mathbf{h}_i^{(L)}$ for nodes in the sampled subgraph. Edge representations are then constructed from pairs of endpoint embeddings and passed to a separate one-layer regression head to predict co-expression values.

In our implementation, we empirically set $L = 3$ and $K = 50$. The model is trained using the Adam optimizer with a fixed learning rate of $10^{-3}$, cosine annealing schedule ($T_{\max} = 200$ and $\eta_{\min} = 10^{-6}$), a batch size of 512 and a total of 200 epochs. We set the graph layer depth $L$ to 3 because increasing the number of message-passing layers can lead to over-smoothing, where node representations become increasingly similar and less discriminative [1, 2]. Therefore, we follow the commonly adopted shallow setting and use an additional regression layer for co-expression prediction. The choice of the top-$K$ reflects a trade-off between performance and computational efficiency. We evaluate the impact of $K$ in a representative gene-level binary classification task that identifies the dosage sensitivity of transcription factors (TFs). Using logistic regression (LR) and support vector machine (SVM) classifiers, we measure performance in terms of the area under the ROC curve (AUC) and find that the performance is relatively insensitive to $K$ (Table S1). Meanwhile, increasing $K$ substantially enlarges the edge set, which in turn increases training time. With our limited computational budget, we finally set $K = 50$ in all experiments.

Table S1: Effect of top-$K$ on performance and efficiency. The experiment was conduct on a PC with Nvidia Tesla A40.

| $K$ | LR | SVM | Training Time (hour) |
|---|---|---|---|
| 50 | 0.944 | 0.955 | $\approx 7$ |
| 100 | 0.940 | 0.956 | $\approx 22$ |
| 200 | 0.938 | 0.952 | $\approx 64$ |
| 300 | 0.934 | 0.949 | $\approx 155$ |
| 400 | 0.935 | 0.954 | $\approx 200$ |
| 500 | 0.939 | 0.944 | $\approx 232$ |

# 2 Implementation details for CoxFormer embedding training

To obtain the CoxFormer gene embeddings, we train the Transformer-based autoencoder described in the Methods section to compress each high-dimensional co-expression vector $(\mathbf{C}_{cox})_{i,:} \in \mathbb{R}^{32,016}$ into a 512-dimensional latent representation $\mathbf{e}_i$, while preserving the global co-expression structure via reconstruction. Specifically, the encoder $f_{\text{enc}}$ maps the input co-expression vector to the latent space, and the decoder $f_{\text{dec}}$ reconstructs the original vector by minimizing the mean squared reconstruction loss $\mathcal{L}_{\text{AE}}$. The learned latent representations $\{\mathbf{e}_i\}$ are used as the final CoxFormer gene embeddings.

In our implementation, the encoder comprises two stacked `TransformerEncoderLayer`s, and the decoder mirrors this structure with two stacked `TransformerDecoderLayer`s [3], forming a symmetric architecture. Each Transformer layer uses multi-head attention with $h = 8$ heads, a feed-forward dimension of 2048, and Rectified Linear Unit (ReLU) activations. The latent embedding dimension is set to 512.

The model is trained by minimizing the mean squared reconstruction loss using Adam

optimizer with an initial learning rate of $10^{-4}$. We adopt a cosine annealing schedule with $T_{\max} = 50$ and $\eta_{\min} = 10^{-6}$ to stabilize convergence. The batch size is set to 32, and the model is trained for 200 epochs.

# 3 Implementation details for CoxFormer multi-modal generative framework

The multi-modal generative framework is implemented with a model dimension $d_{\text{model}} = 512$. When histology images are available, image patches are encoded using a Vision Transformer (ViT) backbone [4]. Patch-level features are averaged within each spot and projected to $d_{\text{img}} = 512$ using a two-layer MLP with ReLU activations.

Spot coordinates are encoded using a Fourier feature encoder with exponentially spaced frequencies $\omega_f = 2^{f-1}\pi$ for $f = 1, \ldots, F$ [5]. We set $F = 32$, resulting in $4F$ positional features per spot, which are projected to $d_{\text{loc}}$ via a two-layer MLP with ReLU activations.

When both image and coordinate features are available, their projected outputs are concatenated and linearly mapped to $d_{\text{model}} = 512$. When only one spatial modality is available, it is directly projected to 512 dimensions.

The attention module uses multi-head self-attention with $h = 8$ heads and $d_k = d_v = d_{\text{model}}/h$. The feed-forward dimension is set to 1024 with dropout 0.1 and ReLU activations.

The gene-conditioning projection matrix $\mathbf{W}_g \in \mathbb{R}^{512 \times d_{\text{model}}}$ aligns the CoxFormer embedding with the spatial feature space. The prediction head consists of LayerNorm followed by four linear layers of width 256 with LeakyReLU activations (slope 0.1), and a final linear layer followed by an ELU activation ($\alpha = 0.01$) and an additive constant of 0.01.

To handle sparsity, we optimize a weighted Huber loss combined with an $\ell_1$ penalty ($\lambda_{\ell_1} = 10^{-3}$). The weighting coefficients for non-zero and zero entries are set to $\lambda_{nz} = 0.5$ and $\lambda_z = 0.5$. The Huber threshold $\delta$ is computed per mini-batch as the standard deviation of residuals.

Training is performed for up to 200 epochs using AdamW with learning rate $10^{-4}$ and weight decay $10^{-4}$. A cosine annealing schedule with $\eta_{\min} = 10^{-6}$ is applied. The batch size is 64. Early stopping is triggered when the exponential moving average of the training loss fails to improve by at least 0.2%.

# References


[1] L. Zhao, L. Akoglu, *PairNorm: Tackling Oversmoothing in GNNs*, in *8th International Conference on Learning Representations, ICLR 2020* (2020)

[2] K. Oono, T. Suzuki, *Graph Neural Networks Exponentially Lose Expressive Power for Node Classification*, in *8th International Conference on Learning Representations, ICLR 2020* (2020)

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need. Advances in neural information processing systems **30** (2017)

[4] R.J. Chen, C. Chen, Y. Li, T.Y. Chen, A.D. Trister, R.G. Krishnan, F. Mahmood, *Scaling vision transformers to gigapixel images via hierarchical self-supervised learning*, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 16144–16155

[5] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)