School of Computer Science
Carnegie Mellon University

BatchBlend

A Batch-Aware Interpretable Probabilistic Graphical Model for Integrating Spatially Resolved Transcriptomes

Integrated Masters Thesis

presented to School of Computer Science of Carnegie Mellon University in partial fulfillment of the requirements for Research Honors in the Degree of *Master of Science in Computational Biology.*

by

Raehash V. Shah

Acknowledgments

I would like to express my deepest gratitude to my research advisor, Dr. Jian Ma, for his guidance, expertise, and support throughout my research journey. His intellectual and scientific curiosity has shaped my approach to computational biology. I am also grateful to the members of the Ma Lab, whose presentations during our weekly meetings consistently challenged my thinking and inspired me to elevate the quality of my work.

I am especially profoundly grateful to Shahul Alam, whose mentorship has been the cornerstone of my research experience. His guidance through complex computational challenges, feedback on countless iterations of my work, and willingness to engage in lengthy discussions about theoretical concepts were invaluable. Without his mentorship, this thesis would not have been possible, and his influence on my growth and development as a scientist will extend far beyond this work.

I extend my appreciation to the other members of my thesis committee, Dr. Andreas Pfenning and Dr. Russell Schwartz, for their questions, and valuable suggestions that enhanced the quality of this work.

The Computational Biology Department has been my academic home for over 5 years, and I am thankful to all faculty and staff who have contributed to my growth. Special thanks to my academic advisors, Dr. Dan DeBlasio and Dr. Phillip Compeau whose counsel and support helped me navigate the program and whose doors were always open when I needed guidance, whether for academic matters or personal advice. To my peers and friends in the department—your camaraderie, intellectual discussions, and mutual support created an environment where I could thrive amid challenges. The debugging sessions fueled by caffeine, impromptu whiteboard discussions, and shared celebrations of victories made this learning experience truly enjoyable.

I am also grateful to my friends throughout the CMU community who provided much-needed balance and perspective. Our "adventures" exploring Pittsburgh and weekend gatherings reminded me that there is a world outside computational biology. Your friendship has been a source of joy and rejuvenation throughout my time at CMU.

Finally, I owe an immeasurable debt of gratitude to my family, whose love and encouragement sustained me through the ups and downs of this academic pursuit. To my parents, my brother, and my extended family who consistently support me, give me advice and motivate me to be better, I am forever grateful.

Raehash V. Shah, 2025

Abstract

While multi-sample spatially-resolved transcriptomics (mSRT) technologies have enhanced our ability to compare different tissues across conditions and time points, they often introduce technical artifacts or batch effects that confound biological interpretability. Current batch effect correction methods either sacrifice interpretability for performance or fail to adequately leverage spatial context. BatchBlend addresses this gap by extending the interpretable framework of Popari with an explicit batch effect term, enabling the separation of technical variation from biological signal while maintaining an interpretable framework. The model is formulated as a non-negative matrix factorization with hidden Markov random fields (NMF-HMRF) similar to Popari with an added sample-specific batch effect term. Through comprehensive evaluation on simulated mSRT data, BatchBlend demonstrates competitive batch effect correction while maintaining moderate biological signal preservation compared to established methods: PyLiger, scVI, and STAligner. In addition, BatchBlend's performance in batch effect correction on mouse brain coronal slices was analyzed when integrating data from MERFISH, MERSCOPE and STARmap. While BatchBlend offers an interpretable approach to mSRT batch correction, further methodological refinements are needed to fully leverage spatial context during integration, presenting promising directions for future research to enhance both integration quality and biological signal preservation in spatially resolved transcriptomics.

Contents

Chapter 1

Introduction and Background

The spatial organization of gene expression is a fundamental determinant of phenotypic heterogeneity in various biological systems. Traditional approaches to quantifying *in vivo* transcriptomic information such as bulk or single-cell RNA sequencing fail to capture spatial context, as they require the dissociation of single cells from their tissue environment. This limitation prevents such technologies from characterizing key *in situ* aspects of cellular architecture such as cell-cell interactions and tissue organization. Recently, spatially-resolved transcriptomics (SRT) technologies have bridged this gap by preserving spatial information while quantifying gene expression.

As the field advances, experimental designs have evolved from single-sample to multi-sample SRT (mSRT) approaches, which allow comparison of different tissues across conditions and time points. While this development enhances statistical power and enables more robust biological comparisons, it can introduce variability driven by technical artifacts or batch effects which confound the biological interpretability of results. Therefore, batch effect correction is essential to mitigate these discrepancies, ensuring that inferred conclusions about the differences in gene expression between conditions are attributable to biological variation rather than technical noise.

1.1 scRNA-seq Data Batch Effect Correction

The challenge of batch effect correction for transcriptomic data integration has been extensively investigated. Numerous methods have been developed for scRNA-seq data. scVI [1] employs a variational autoencoder framework that models count distribution and technical effects in scRNA-seq data. While effective at batch effect correction, its complex neural network architecture and non-linear transformations create latent representations that lack direct biological interpretability. Similarly, Harmony [2] uses iterative clustering and correction to project cells into a shared embedding, but the transformations applied during this process cannot be easily attributed back to specific genes. BBKNN, Batch Balanced K-Nearest Neighbors, [3] and Scanorama [4] prioritize computational efficiency and correction performance through k-nearest neighbor graphs and panorama stitching approaches, respectively, but sacrifice interpretability in their integration processes.

Several interpretable methods have also been developed for scRNA-seq batch effect correction. PyComBat [5] implements a parametric empirical Bayes framework that estimates location and scale parameters for each batch to model gene expression as a function of the biological covariates with batch effect [6]. Although robust for bulk RNA-seq data, PyComBat cannot adequately capture the sparsity characteristic of single-cell and spatial transcriptomic data. PyLiger[7], which extends LIGER[8] to Python, uses integrative non-negative matrix factorization (iNMF) to learn shared and dataset-specific factors, thereby identifying common metagenes across batches.

Similarly, UINMF[9] extends the iNMF approach to find shared and dataset-specific factors for integration by using a union-intersection approach to identify similar biological signals while accounting for dataset-specific features. By maintaining the non-negativity constraints of transcriptomic data, PyLiger and UINMF preserve biological interpretability, where factors represent co-expressed genes. Seurat's [10] anchor-based integration methods, which align datasets based on shared cell states, maintain interpretability by operating directly on gene expression values, allowing users to trace integration effects back to specific genes across different batches. Despite their enhanced interpretability, these methods typically fail to incorporate spatial relationships between cells or spots in SRT data.

1.2 mSRT Data Batch Effect Correction

In mSRT analysis, correcting for batch effect while integrating transcriptomic and spatial data poses distinct challenges. Due to the reduced number of genes compared to scRNA-seq data [11], the lower spatial resolution of SRT technologies [12], and technical artifacts in the experimental procedures [11], SRT data typically exhibits a much lower signal-to-noise ratio than scRNA-seq data, thus decreasing the expected success rate of prior batch effect correction algorithms designed for the scRNA-seq modality.

PRECAST [13], an early attempt to mitigate these drawbacks, models spatial data as a probabilistic factor model and estimates low-dimensional embeddings that account for batch effects by unifying spatial factor analysis with clustering and embedding alignment. However, this approach does not explicitly capture the spatial interaction between the learned factors and does not enforce non-negativity of the learned factors, hindering the interpretability of the model.

Subsequently developed methods for mSRT analysis tend to leverage strong inductive biases in order to handle low data quality. For example, graph learning-based approaches have emerged as a dominant strategy to integrate SRT data. INSPIRE [14] is a deep learning-based approach that uses a graph neural network (GNN) and adversarial learning to adaptively remove batch effects from biological variation. By incorporating cell-cell interactions through spatial graphs, it recovers biological signals from technical variation, but the nonlinearities introduced by the GNN means that INSPIRE offers even less interpretability than PRECAST.

Similarly, SpatiAlign [15] employs a deep graph infomax (DGI) framework that captures intercellular relationships and uses self-supervised contrastive learning to align biologically similar cells. While SpatiAlign outperforms PRECAST [15], it also sacrifices interpretability due to its complex graph transformations. Building on these graph-based approaches, STAligner [16] introduces a spatial-temporal framework that incorporates both spatial coordinates and gene expression into a unified graph representation. Despite its effectiveness in capturing complex spatial patterns during integration, STAligner, like INSPIRE and SpatiAlign, employs multiple graph transformation steps and embedding alignments that obscure the direct relationship between input genes and integration results.

1.3 Gap in Current Approaches

The current landscape of mSRT batch effect correction methods predominantly relies on deep learning architectures that, while effective, sacrifice interpretability — a crucial aspect for generating biological insights. Alternatively, scRNA-seq methods that maintain interpretability often inadequately capture spatial interactions. This highlights a significant gap in the field: the need for methods that can both address the unique challenges of mSRT data while maintaining the interpretability that is crucial for biological insight.

Popari is a probabilistic graphical model designed specifically for mSRT analysis. It employs a non-negative matrix factorization coupled with a hidden Markov random field (NMF-HMRF) approach to capture both gene

expression patterns and spatial context. The framework represents gene expression data via a combination of shared metagenes and sample-specific spatial parameters, while incorporating spatial information through a graph structure that can model relationships between neighboring spatial entities. This interpretable approach allows researchers to identify co-expressed gene modules and visualize their spatial distribution across multiple tissue samples. The key strength of Popari is its ability to maintain biological interpretability through non-negative matrix factorization while accounting for the spatial context through the HMRF component. However, Popari lacks an explicit formulation for batch effect correction which limits its ability to integrate data from different experimental runs without pre-processing to mitigate the technical variation.

Therefore, BatchBlend directly addresses this gap by incorporating batch effect correction within the established Popari framework. By maintaining the original model's interpretable structure while adding a batch effect term, we can model the separability of technical variation from the biological representation of the data without compromising the interpretive power of the model.

Chapter 2

Formulation of BatchBlend

2.1 Formulation of BatchBlend as a Probabilistic Model

Multi-sample spatially-resolved transcriptomics (mSRT) data are comprised of spatial coordinates and gene expression profiles for cellular entities across multiple tissue samples. For an SRT sample indexed by t, the data $\mathbf{D}^t$ is represented by a tuple $\mathbf{D}^t = \{\mathbf{Y}^t, \mathbf{C}^t\}$ where $\mathbf{Y}^t \in \mathbb{R}_+^{G\times N}$ represents the gene expression profile for N spatial entities across G genes and $\mathbf{C}^t \in \mathbb{R}^{2\times N}$ represents the 2D Euclidean coordinates of the sample.

Building upon the non-negative matrix factorization (NMF) framework established by Popari, BatchBlend extends the model to explicitly account for batch effects. In the original Popari formulation, for a sample t, the gene expression matrix $\mathbf{Y}^t$ is represented as a function of shared K metagenes matrix $\mathbf{M} \in \mathbb{R}_+^{G\times K}$, sample-specific coefficients $\mathbf{X}^t \in \mathbb{R}_+^{K\times N}$, and unexplained variance $\mathbf{E}^t \in \mathbb{R}^{G\times N}$ where each column $e_i \sim \mathcal{N}(0, (\sigma_y^t)^2 I)$ drawn identically and independently distribution from a multivariate Gaussian with a shared variance parameter $(\sigma_y^t)^2$.

$$\mathbf{Y^t} = \mathbf{M}\mathbf{X^t} + \mathbf{E^t}$$

BatchBlend explicitly incorporates a batch effect term $\boldsymbol{\beta} \in \mathbb{R}_+^{T\times K}$ as an additive term which is unique to each sample t yielding:

$$\mathbf{Y^t} = \mathbf{M}(\mathbf{X^t} + \boldsymbol{\beta^t}) + \mathbf{E^t}$$

This formulation preserves the fundamental structure of the original model while introducing a sample-specific batch effect term $\boldsymbol{\beta}^t$. The key advantage of this approach is that it maintains the interpretability of shared metagenes $\mathbf{M}$ across samples while explicitly modeling batch-related variation.

Within this probabilistic graphical model, we define a node potential function (**Eq.** 2.1) that quantifies the difference between observed gene expression and our model's prediction for a sample t and spatial entity i.

$$\phi(x_i^t, y_i^t, \beta^t) = \exp\left(-\frac{\|y_i^t - \mathbf{M}(x_i^t + \boldsymbol{\beta^t})\|_2^2}{2(\sigma_y^t)^2}\right) \tag{2.1}$$

This function assigns higher values when the model's prediction closely matches the observed expression data, reflecting a higher probability of the observed data given the model parameters.

To capture spatial relationships between neighboring entities, we define edge potentials (**Eq.** 2.2) for any sample t and spatial entities i and j. The matrix, $\boldsymbol{\Lambda}^t \in \mathbb{R}^{K\times K}$ contains learnable parameters that express the pairwise spatial affinities of the K metagenes.

$$\varphi(x_i^t, x_j^t) = \exp\left(-\frac{(x_i^t)^\top}{\|x_i^t\|_1}\boldsymbol{\Lambda^t}\frac{(x_j^t)}{||x_j^t||_1}\right) \tag{2.2}$$

The complete joint probability distribution for our NMF-HMRF model can be expressed as:

$$\begin{aligned} P(\mathbf{Y}, \mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\Theta}) &= \prod_{t=1}^{T} P(\boldsymbol{Y^t}, \boldsymbol{X^t}, \boldsymbol{\beta^t}, \boldsymbol{\Theta^t}) \\ &= \prod_{t=1}^{T} P(\boldsymbol{Y^t}, \boldsymbol{X^t}, \boldsymbol{\beta^t}|\boldsymbol{\Theta^t}) P(\boldsymbol{\Theta^t}) \\ &= \prod_{t=1}^{T} P(\boldsymbol{Y^t}, \boldsymbol{X^t}, \boldsymbol{\beta^t}|\boldsymbol{\Theta^t}) P(\boldsymbol{\Lambda^t}) \end{aligned}$$

where $\boldsymbol{\Theta} = \{\mathbf{M}, \boldsymbol{\Lambda}, \sigma_y\}$ represents the learnable model parameters and $P(\Lambda^t)$ is a Gaussian prior distribution over $\boldsymbol{\Lambda}^t$.

By applying the Hammersley-Clifford theorem, we can express the model's likelihood as a product of the node and edge potentials:

$$P(\boldsymbol{Y^t}, \boldsymbol{X^t}, \boldsymbol{\beta^t}|\boldsymbol{\Theta^t}) = \frac{1}{Z(\boldsymbol{\Theta^t})} \prod_{i=1}^{N^t} \left[\phi(x_i^t, y_i^t, \boldsymbol{\beta^t}) \pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right]$$

where $Z(\Theta^t)$ is the partition function that ensures proper normalization of the probability distribution. In addition, we introduce a prior on the representation of node i relative to x_i^t: $\pi_x(x_i^t) = \exp(-\lambda\|x_i^t - \bar{x}_i^t\|_1)$ such that

$$\bar{x}_i^t = \frac{1}{T-1} \sum_{t'\in T\setminus t} \left[\frac{1}{|N^{t'}|} \sum_{i\in N^{t'}} x_i^{t'} \right]$$

Note that this term is constant with respect to x_i^t.

2.2 Alternating Coordinate Descent Optimization

Directly maximizing the likelihood function of the joint NMF-HMRF model with batch effect correction is computationally intractable. To address this challenge, we employ an alternating coordinate descent approach that optimizes each set of parameters while holding the others fixed.

Our optimization strategy proceeds as follows:

1. Fix $\boldsymbol{\Theta^t}$ and $\boldsymbol{\beta^t}$, and optimize $\mathbf{X}^t$
2. Fix $\boldsymbol{\Theta^t}$ and $\mathbf{X}^t$, optimize $\boldsymbol{\beta^t}$
3. Fix $\mathbf{X}^t$ and $\boldsymbol{\beta^t}$, optimize $\boldsymbol{\Theta^t}$

For the latent embeddings $\mathbf{X}^t$, we derive the maximum a posteriori (MAP) estimate through a reparameterization of the posterior distribution, leading to a constrained optimization problem that can be solved using projected coordinate descent with acceleration. Similarly, for the batch effect parameters $\boldsymbol{\beta}^t$, we employ a different reparameterization of the posterior distribution, resulting in another constrained optimization problem solvable

via projected coordinate descent. When the optimization problem exhibits strong convexity, indicating by the positive definiteness of the Hessian, we leverage Nesterov's Accelerated Gradient method to achieve faster convergence. Nesterov's Accelerated Gradient improves upon standard gradient descent by incorporating momentum, which helps avoid oscillations in regions of high curvature while maintaining rapid convergence in flatter regions. This acceleration is particularly beneficial for these two strongly convex subproblems and hence we will verify the positive definiteness of the Hessian matrices. For each of the model parameters $\mathbf{\Theta}^t$, we employ different strategies. The shared metagenes $\mathbf{M}$ require joint optimization across all graphs and are estimated using projected gradient descent with acceleration, as the corresponding optimization problem is strongly convex. The spatial interaction parameters $\mathbf{\Lambda}^t$ are optimized using the Adam optimizer, as the MAP estimate leads to a convex optimization problem. The variance parameter σ_y^t has a closed-form solution derived from the likelihood function.

In the following sections, we derive the optimization procedures for each component of our model in detail.

2.3 Derivation of the Latent Space (X^t) Optimization

To optimize $\mathbf{X^t}$ we employ the MAP estimate:

$$\hat{X}^t_{MAP} = \underset{X^t \in \mathbb{R}_+^{K \times N}}{\operatorname{argmax}} P(X^t | Y^t, \beta^t, \Theta^t)$$

By Bayes' rule and taking the negative logarithm, this is equivalent to:

$$\begin{aligned}
\hat{X}^t &= \underset{X^t}{\operatorname{argmax}} P(Y^t, X^t, \beta^t | \Theta^t) P(\mathbf{\Lambda^t}) \\
&= \underset{X^t}{\operatorname{argmin}} \left[-\log(P(Y^t, X^t, \beta^t | \Theta^t)) - \log(P(\Lambda^t)) \right] \\
&= \underset{X^t}{\operatorname{argmin}} \left[-\log \left(\frac{1}{Z(\Theta^t)} \prod_{i=1}^{N^t} \left[\phi(x_i^t, y_i^t, \beta^t) \pi_x(x_i^t) \prod_{(i,j) \in \mathcal{E}^t} \varphi(x_i^t, x_j^t) \right] \right) \right] \\
&= \underset{X^t}{\operatorname{argmin}} \left[\sum_{i \in \mathcal{V}^t} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|x_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j) \in \mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{\|x_j^t\|_1} \right] + \log(Z(\Theta^t)) \right]
\end{aligned}$$

To facilitate optimization, we decompose x_i^t into $s_i^t z_i^t$, where $s_i^t = \|x_i^t\|_1 \in \mathbb{R}_+$ represents a size factor and $z_i^t \in \mathbb{S}_{K-1}$ is a point on the K-dimensional simplex. This decomposition preserves convexity and leads to a more efficient optimization procedure:

$$\begin{aligned}
\hat{x}_i^t &= \underset{x_i^t}{\operatorname{argmin}} \left[\frac{\|y_i^t - M(x_i^t + \boldsymbol{\beta^t})\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|x_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j) \in \mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{\|x_j^t\|_1} \right] \\
&= \underset{s_i^t \in \mathbb{R}_+, z_i^t \in \mathbb{S}_{k-1}}{\operatorname{argmin}} \left[\frac{\|y_i^t - s_i^t M z_i^t - M\beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|s_i^t \hat{z}_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j) \in \mathcal{E}^t} (z_i^t)^\top \Lambda^t (z_j^t) \right] \qquad (2.3)
\end{aligned}$$

With this reformulation, we can alternately optimize s_i^t and z_i^t. For updating s_i^t with fixed $z_i^t = \hat{z}_i^t$, we compute the partial derivative:

$$\frac{\partial}{\partial s_i^t}\left[\frac{\|y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|s_i^t\hat{z}_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j)\in\mathcal{E}^t} (\hat{z}_i^t)^\top \Lambda^t(z_j^t)\right]$$

$$= \frac{\partial}{\partial s_i^t}\left[\frac{\|y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|s_i^t\hat{z}_i^t - \bar{x}_i^t\|_1\right]$$

$$= \frac{\partial}{\partial s_i^t}\left[\frac{(y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)^\top(y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)}{2(\sigma_y^t)^2} + \lambda_x^\top \|s_i^t\hat{z}_i^t - \bar{x}_i^t\|_1\right]$$

$$= \frac{\partial}{\partial s_i^t}\left[\frac{(y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)^\top(y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)}{2(\sigma_y^t)^2} + \sum_k \lambda_x \cdot \operatorname{sgn}\left(s_i^t\hat{z}_{ik}^t - \overline{x}_{ik}^t\right)\cdot\left(s_i^t\hat{z}_{ik}^t - \overline{x}_{ik}^t\right)\right]$$

$$= \frac{-2(M\hat{z}_i^t)^\top y_i^t + 2s_i^t(M\hat{z}_i^t)^\top(M\hat{z}_i^t) + 2(M\hat{z}_i^t)^\top(M\beta^t)}{2(\sigma_y^t)^2} + \lambda_x \sum_k \cdot \operatorname{sgn}\left(s_i^t\hat{z}_{ik}^t - \overline{x}_{ik}^t\right)\cdot\hat{z}_{ik}^t$$

where sgn represents the signum function. Due to the presence of these terms, the value of the derivative depends on the region of $\mathbb{R}_+$ in which we constrain s_i^t to lie; for convenience, we constrain s_i^t to the region in which $\operatorname{sgn}\left(s_i^t - \frac{\overline{x}_{ik}^t}{\hat{z}_{ik}^t}\right) < 0\,\forall\, k$; this simplifies the derivative greatly, as the signum function always evaluates to -1.

Setting this expression to 0 and solving for s_i^t, we obtain:

$$\hat{s}_i^t = \frac{2(M\hat{z}_i^t)^\top(y_i^t) - 2(M\hat{z}_i^t)^\top(M\beta^t) + 2(\sigma_y^t)^2\lambda_x}{2(M\hat{z}_i^t)^\top(M\hat{z}_i^t)}$$

$$= \frac{(M\hat{z}_i^t)^\top(y_i^t) - (M\hat{z}_i^t)^\top(M\beta^t) + (\sigma_y^t)^2\lambda_x}{(M\hat{z}_i^t)^\top(M\hat{z}_i^t)}$$

Given our constraints, the final step is to clip $\hat{s}_i^t$ to lie within the convex constraint space:

$$m_i = \min_k\left(\frac{\overline{x}_{ik}^t}{\hat{z}_{ik}^t}\right)$$

$$\hat{s}_i^t = \min\left(\max\left(\frac{(M\hat{z}_i^t)^\top(y_i^t) - (M\hat{z}_i^t)^\top(M\beta^t) + (\sigma_y^t)^2\lambda_x}{(M\hat{z}_i^t)^\top(M\hat{z}_i^t)}, 0\right), m_i\right)$$

For optimizing z_i^t with fixed $s_i^t = \hat{s}_i^t$, we compute the both the gradient and the Hessian:

$$\frac{\partial^2}{\partial(z_i^t)^2}\left[\frac{\|y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_x\|s_i^t\hat{z}_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j)\in\mathcal{E}^t}(\hat{z}_i^t)^\top\Lambda^t(z_j^t)\right]$$

$$=\frac{\partial^2}{\partial(z_i^t)^2}\left[\frac{(y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)^\top(y_i^t - s_i^t M\hat{z}_i^t - M\beta^t)}{2(\sigma_y^t)^2} + \lambda_x\|s_i^t\hat{z}_i^t - \overline{x}_{ik}^t\|_1 + \sum_{(i,j)\in\mathcal{E}^t}(\hat{z}_i^t)^\top\Lambda^t(z_j^t)\right]$$

$$=\frac{\partial}{\partial z_i^t}\left[\frac{-\hat{s}_i^t M^\top(y_i^t) + (\hat{s}_i^t)^2(M^TM)(z_i^t) + (\hat{s}_i^t)(\beta^t)^\top(M^TM)}{(\sigma_y^t)^2} + \lambda_x s_i^t\cdot\operatorname{sgn}(s_i^t\hat{z}_i^t - \overline{x}_i^t) + \sum_{(i,j)\in\mathcal{E}^t}\Lambda^t(z_j^t)\right]$$

$$=\frac{2(\hat{s}_i^t)^2(M^TM)}{2(\sigma_y^t)^2}$$

$$=\frac{(\hat{s}_i^t)^2(M^TM)}{(\sigma_y^t)^2}$$

Since $\hat{s}_i^t \in \mathbb{R}_+$ and $\mathbf{M} \in \mathbb{R}_+^{G\times K}$, the Hessian is positive definite, confirming the problem is strongly convex. Thus, it is appropriate to use proximal Nesterov's Accelerated Gradient method for optimization.

2.4 Derivation of the Batch Effect (β^t) Optimization

To optimize the batch effect parameters $\boldsymbol{\beta}^t$, we again employ a MAP estimate:

$$\hat{\beta}_{MAP}^t = \underset{\beta^t\in\mathbb{R}^{T\times K}}{\operatorname{argmax}} P(\beta^t|X^t, Y^t, \Theta^t)$$

Following a similar approach as for $\mathbf{X}^t$, we can express this as:

$$\hat{\beta}^t = \underset{\beta^t}{\operatorname{argmax}} P(Y^t, X^t, \beta^t|\Theta^t)P(\Lambda^t)$$

$$= \underset{\beta^t}{\operatorname{argmin}}[-\log(P(Y^t, X^t, \beta^t|\Theta^t)) - \log(P(\Lambda^t))]$$

To simplify the optimization, we employ the mean-field approximation:

$$P(Y^t, X^t, \beta^t|\Theta^t) \approx P(\beta^t|Y^t, X^t, \Theta^t)\prod_{i=1}^{N^t}P(y_i^t, x_i^t|\beta^t, \Theta^t, X_{-i}^t, Y_{-i}^t) \tag{2.4}$$

$$= \frac{1}{\overline{Z}(\Theta^t)}\frac{1}{Z_{\beta^t}}\prod_{i=1}^{N^t}[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t)\prod_{(i,j)\in\mathcal{E}^t}\varphi(x_i^t, x_j^t)]$$

where

$$\psi(\beta^t) = \exp\left(-\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2}\right)$$

$$\phi(x_i^t, y_i^t) = \exp\left(-\frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2}\right)$$

The partition function is given by

$$\begin{aligned}
Z(\Theta^t) &= \int_{X^t, Y^t, \beta^t} P(Y^t, X^t, \beta^t | \Theta^t)\, \mathrm{d}X^t\, \mathrm{d}Y^t\, \mathrm{d}\beta^t \\
&= \int_{X^t, Y^t} \prod_{i=1}^{N^t} P(y_i^t, x_i^t | \beta^t, \Theta^t, X_{-i}^t, Y_{-i}^t)\, \mathrm{d}X^t\, \mathrm{d}Y^t \int_{\beta^t} P(\beta^t | Y^t, X^t, \Theta^t)\, \mathrm{d}\beta^t \\
&= \int_{X^t, Y^t} \prod_{i=1}^{N^t} \left[\phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t)\right] \mathrm{d}X^t\, \mathrm{d}Y^t \int_{\beta^t} \psi(\beta^t)\, \mathrm{d}\beta^t \qquad (2.5) \\
&= \bar{Z}(\Theta^t) Z_{\beta^t}
\end{aligned}$$

This leads to the following optimization problem for $\boldsymbol{\beta}^t$:

$$\begin{aligned}
\hat{\beta^t} &= \underset{\beta^t}{\operatorname{argmin}} \left[-\log\left(\frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z_{\beta^t}} \prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t)\right]\right)\right] \\
&= \underset{\beta^t}{\operatorname{argmin}} \sum_{i=1}^{N^t} [\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2} + \frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|x_i^t - \bar{x}_i^t\|_1 \\
&\quad + \sum_{(i,j)\in\mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{\|x_j^t\|_1}] + \log(\bar{Z}(\Theta^t) + \log(Z_{\beta^t}) \\
&= \underset{\beta^t}{\operatorname{argmin}} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2}\right]
\end{aligned}$$

For a specific node i, this becomes:

$$\beta^t = \underset{\beta^t}{\operatorname{argmin}} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2}\right]$$

The Hessian of this objective is given by

$$\frac{\partial^2}{\partial(\beta^t)^2}\left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top(M\beta^t)\|_1}{2(\sigma_y^t)^2}\right]$$

$$=\frac{\partial^2}{\partial(\beta^t)^2}\left[\frac{(M\beta^t)^\top(M\beta^t) - 2(y_i^t - Mx_i^t)^\top(M\beta^t)}{2(\sigma_y^t)^2}\right]$$

$$=\frac{\partial}{\partial\beta^t}\left[\frac{2(M^\top M)\beta^t - 2(y_i^t)^\top M + 2(x_i^t)^\top(M^\top M)}{2(\sigma_y^t)^2}\right]$$

$$=\frac{M^\top M}{(\sigma_y^t)^2}$$

Since $\mathbf{M} \in \mathbb{R}_+^{G\times K}$, the Hessian is positive definite, confirming that this problem is α-strongly convex. Therefore, we can employ the proximal Nesterov's Accelerated Gradient method for optimization.

2.5 Derivation of Parameters (Θ^t) Optimization

Having established the optimization procedures for the latent space variables $\mathbf{X}^t$ and batch effect parameters $\boldsymbol{\beta}^t$, we now turn our attention to optimizing the model parameters $\boldsymbol{\Theta}^t = \{\mathbf{M}, \boldsymbol{\Lambda}^t, \sigma_y^t\}$. For the model parameters $\boldsymbol{\Theta}^t$, we maximize the posterior probability:

$$\begin{aligned}\hat{\Theta}_{MAP} &= \underset{\Theta}{\operatorname{argmax}}\, P(Y, X, \beta, \Theta) \\ &= \underset{\Theta}{\operatorname{argmax}} \prod_{t=1}^{T} P(Y^t, X^t, \beta^t|\Theta^t)P(\Lambda^t) \\ &= \underset{\Theta}{\operatorname{argmax}} \sum_{t=1}^{T} \log(P(Y^t, X^t, \beta^t|\Theta^t)) + \log(P(\Lambda^t))\end{aligned}$$

Using the mean-field approximation in **Eq.** 2.4 and the Hammersley-Clifford theorem, we can express the optimization for a sample t as:

$$\hat{\Theta}^t \approx \underset{\Theta^t}{\text{argmax}} \left[\log \left(P(\beta^t | Y^t, X^t, \Theta^t) \prod_{i=1}^{N^t} P(y_i^t, x_i^t | \beta^t, \Theta^t, X_{-i}^t, Y_{-i}^t) \right) + \log(P(\Lambda^t)) \right]$$

$$= \underset{\Theta^t}{\text{argmax}} \sum_{i=1}^{N^t} \left[\log(\psi(\beta^t)) + \log(\phi(x_i^t, y_i^t)) + \log(\pi_x(x_i^t)) + \sum_{(i,j)\in\mathcal{E}^t} \log(\varphi(x_i^t, x_j^t)) \right] - \log(\bar{Z}(\Theta^t)) - \log(Z_{\beta^t}) + \log(P(\Lambda^t))$$

$$= \underset{\Theta^t}{\text{argmax}} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2} + \frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|x_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j)\in\mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{\|x_j^t\|_1} \right] - \log(\bar{Z}(\Theta^t) - \log(Z_{\beta^t}) + \log(P(\Lambda^t))$$

$$= \underset{\Theta^t}{\text{argmin}} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2} + \frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2} + \lambda_x^\top \|x_i^t - \bar{x}_i^t\|_1 + \sum_{(i,j)\in\mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{\|x_j^t\|_1} \right] + \log(\bar{Z}(\Theta^t) + \log(Z_{\beta^t}) - \log(P(\Lambda^t))$$

The optimization of $\mathbf{\Theta}^t$ involves calculating the partition functions $\bar{Z}(\mathbf{\Theta}^t)$ and Z_{β^t}, and then separately optimizing each of the parameters $\mathbf{M}$, $\mathbf{\Lambda}^t$, and σ_y^t.

2.5.1 Calculation of Partition Functions

Using the partition function expressions derived earlier in **Eq.** 2.5, we can compute the analytical form for Z_{β^t}:

$$Z_{\beta^t} = \int_{\beta^t} \psi(\beta^t)\, \mathrm{d}\beta^t$$

$$= \int_{\beta^t} \exp\left(-\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^T (M\beta^t)\|_1}{2(\sigma_y^t)^2} \right) \mathrm{d}\beta^t$$

$$= \int_{\beta^t} \exp\left(-\frac{(M\beta^t)^\top (M\beta^t) - 2\sum_{i\in\mathcal{V}^t} (y_i^t - Mx_i^t)^\top (M\beta^t)}{2(\sigma_y^t)^2} \right) \mathrm{d}\beta^t$$

Defining $\mathbf{c} = \sum_{i \in V^t}(y_i^t - \mathbf{M}x_i^t)$, which is a constant with respect to $\boldsymbol{\beta}^t$, we have:

$$Z_{\beta^t} = \int_{\beta^t} \exp\left(-\frac{(M\beta^t)^\top (M\beta^t) - 2c^\top (M\beta^t)}{2(\sigma_y^t)^2}\right) \mathrm{d}\beta^t$$

$$= \int_{\beta^t} \exp\left(-\frac{(\beta^t)^\top (M^\top M)\beta^t - 2(c^\top M)\beta^t)}{2(\sigma_y^t)^2}\right) \mathrm{d}\beta^t$$

By completing the square in the exponent:

$$Z_{\beta^t} = \int_{\beta^t} \exp\left(-\frac{(\beta^t - (M^\top M)^{-1}(M^\top c))^\top (M^\top M)(\beta^t - (M^\top M)^{-1}(M^\top c)) - (M^\top c)^\top (M^\top M)^{-1}(M^\top c)}{2(\sigma_y^t)^2}\right) \mathrm{d}\beta^t$$

The term $(\mathbf{M}^T\mathbf{c})^T(\mathbf{M}^T\mathbf{M})^{-1}(\mathbf{M}^T\mathbf{c})$ is a constant, so we can factor it out:

$$Z_{\beta^t} \propto \int_{\beta^t} \exp\left(-\frac{(\beta^t - (M^\top M)^{-1}(M^\top c))^\top (M^\top M)(\beta^t - (M^\top M)^{-1}(M^\top c))}{2(\sigma_y^t)^2}\right) \mathrm{d}\beta^t$$

This integral is a multidimensional Gaussian integral, which has the closed-form solution:

$$Z_{\beta^t} = (2\pi(\sigma_y^t)^2)^{\frac{K}{2}} \det(M^\top M)^{-\frac{1}{2}}$$

For the derivation of the node-specific partition function we can first decompose the integral:

$$\bar{Z}_i(\Theta^t) = \int_{x_i^t, y_i^t} \phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t)\,\mathrm{d}x_i^t\,\mathrm{d}y_i^t$$

$$= \int_{x_i^t} \pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \int_{y_i^t} \phi(x_i^t, y_i^t)\,\mathrm{d}y_i^t\,\mathrm{d}x_i^t$$

The inner integral ${\bar{Z}_i}^Y(\Theta^t)$ over y_i^t can be computed as a Gaussian integral:

$${\bar{Z}_i}^Y(\Theta^t) = \int_{y_i^t} \phi(x_i^t, y_i^t)\,\mathrm{d}y_i^t$$

$$= \int_{y_i^t} \exp\left(-\frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2}\right) \mathrm{d}y_i^t$$

$$= (2\pi(\sigma_y^t)^2)^{\frac{G}{2}}$$

For the outer integral over x_i^t, we use the same reparameterization as in **Eq.** 2.3, writing $x_i^t = s_i^t z_i^t$ where $s_i^t = |x_i^t|_1 \in \mathbb{R}_+$ and $z_i^t \in \mathbb{S}_{K-1}$ is a point on the K-dimensional simplex. The Jacobian of this transformation is given by $(s_i^t)^{K-1}$, yielding:

$$\bar{Z_i}^X(\Theta^t) = \int_{x_i^t} \pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t)\,\mathrm{d}x_i^t$$

$$= \int_{s_i^t, z_i^t} (s_i^t)^{K-1} \exp(-\lambda_x(s_i^t)) \prod_{(i,j)\in\mathcal{E}^t} \exp(-(z_i^t)^\top \Lambda^t(z_j^t))\,\mathrm{d}s_i^t\,\mathrm{d}z_{i,1}^t \cdots \mathrm{d}z_{i,K-1}^t$$

$$= \int_{s_i^t} (s_i^t)^{K-1} \exp(-\lambda_x(s_i^t))\,\mathrm{d}s_i^t \int_{z_i^t} \exp\left(-\sum_{(i,j)\in\mathcal{E}^t} (z_i^t)^\top \Lambda^t(z_j^t)\right) \mathrm{d}z_{i,1}^t \cdots \mathrm{d}z_{i,K-1}^t$$

The integral over s_i^t can be computed to a closed form as follows:

$$\int_{s_i^t} (s_i^t)^{K-1} \exp(-\lambda_x(s_i^t))\,\mathrm{d}s_i^t = \lambda_x^{-K}(K-1)!$$

To find the analytic form for the integral over z_i^t, let $\eta_{h,i}^t := \Lambda_h^t \sum_{j\in\eta(i)} z_{h,j}^t$. Then we define a function $f_{ik}^t : z_{ik}^t \mapsto \exp\left(-\eta_{ik}^t \cdot z_{ik}^t\right)$ for each dimension of $\eta_{h,i}^t$. Then we can rewrite the integral as follows where the $*$ symbol represents the convolution operator.

$$Z_i^Z(\Theta_h^t) := \int_{z_{h,i}^t \in \mathbb{S}_{K-1}} \exp\left(-\sum_{j\in\eta(i)} (z_{h,i}^t)^\top \Lambda_h^t z_{h,j}^t\right) dz_{h,i(K-1)}^t \cdots dz_{h,i1}^t$$

$$= \int_{z_{h,i}^t \in \mathbb{S}_{K-1}} \exp\left(-(z_{h,i}^t)^\top \eta_{h,i}^t\right)\, dz_{h,i(K-1)}^t \cdots dz_{h,i1}^t$$

$$= \int_0^q \cdots \int_0^{q-\sum_{k=1}^{K-2} z_{h,ik}^t} \exp\left(-(z_{h,i}^t)^\top \eta_{h,i}^t\right)\, dz_{h,i(K-1)}^t \cdots dz_{h,i1}^t \Big|_{q=1}$$

$$= (*_{k=1}^K f_{h,ik}^t)(1)$$

This means that the integral represents the convolution of the functions f evaluated at 1. From this step, we can apply the convolution theorem with the Laplace transform $\mathcal{L}$:

$$\mathcal{L}[(*_{k=1}^K f_{h,ik}^t)(1)] = \prod_{k=1}^K \mathcal{L}[f_{h,ik}^t] = s \mapsto \prod_{k=1}^K \frac{1}{s+\eta_{ik}^t}$$

We can then apply a partial fraction decomposition:

$$\prod_{k=1}^{K} \frac{1}{s+\eta_{ik}^t} = \sum_{k=1}^{K} \frac{a_k \prod_{j\neq k}(s+\eta_{ij}^t)}{s+\eta_{ik}^t}$$

$$\implies a_k \cdot \prod_{j\neq k}(-\eta_{ik}^t + \eta_{ij}^t) = 1$$

$$\implies \prod_{k=1}^{K} \frac{1}{s+\eta_{ik}^t} = \sum_{k=1}^{K} \frac{1}{(s+\eta_{ik}^t)\prod_{j\neq k}(\eta_{ij}^t - \eta_{ik}^t)}$$

Now, since the inverse Laplace transform $\mathcal{L}^{-1}$ is linear, we can then compute this as follows:

$$\mathcal{L}^{-1}\left[\mathcal{L}\left[\left(*_{k=1}^{K} f_{h,ik}^t\right)(1)\right]\right] = \sum_{k=1}^{K} \frac{1}{\prod_{j\neq k}(\eta_{ij}^t - \eta_{ik}^t)} \cdot \mathcal{L}^{-1}\left[\frac{1}{(s+\eta_{ik}^t)}\right]$$

$$= \sum_{k=1}^{K} \frac{\exp(-\eta_{ik}^t)}{\prod_{j\neq k}(\eta_{ij}^t - \eta_{ik}^t)}$$

Therefore the full analytic form of the integral over z_i^t is as follows and can be computed efficiently in each iteration of the parameter update steps.

$$\int_{z_i^t} \exp\left(-\sum_{(i,j)\in\mathcal{E}^t} (z_i^t)^\top \Lambda^t (z_j^t)\right) \mathrm{d}z_{i,1}^t \cdots \mathrm{d}z_{i,K-1}^t = \sum_{k=1}^{K} \frac{\exp(-\Lambda^t \sum_{i,k} \hat{z}_k)}{\prod_{j\neq k}((\Lambda^t \sum_{i,j} \hat{z}_j) - (\Lambda^t \sum_{i,k} \hat{z}_k))}$$

Therefore, the complete node-specific partition function is:

$$\bar{Z}_i(\Theta^t) = \int_{x_i^t} \pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \int_{y_i^t} \phi(x_i^t, y_i^t)\,\mathrm{d}y_i^t\,\mathrm{d}x_i^t$$

$$= \int_{x_i^t} \pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \bar{Z}_i{}^Y(\Theta^t)\,\mathrm{d}x_i^t$$

$$= \bar{Z}_i{}^Y(\Theta^t)\bar{Z}_i{}^X(\Theta^t)$$

$$= (2\pi(\sigma_y^t)^2)^{\frac{G}{2}} \lambda_x^{-K}(K-1)! \sum_{k=1}^{K} \frac{\exp(-\Lambda^t \sum_{i,k} \hat{z}_k)}{\prod_{j\neq k}((\Lambda^t \sum_{i,j} \hat{z}_j) - (\Lambda^t \sum_{i,k} \hat{z}_k))}$$

With these partition functions, we can now proceed to deriving the optimization of each of the model parameters.

2.5.2 Optimization of the Metagene Matrix M

The optimization of the shared metagene matrix $\mathbf{M}$ across all samples is formulated as:

$$\widehat{M} = \underset{M}{\operatorname{argmax}} \prod_{t=1}^{T} \frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z_{\beta^t}} \prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right]$$

$$= \underset{M}{\operatorname{argmax}} \sum_{t=1}^{T} \log \left[\prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right] \right] - \log(\bar{Z}(\Theta^t)) - \log(Z_{\beta^t})$$

$$= \underset{M}{\operatorname{argmax}} \sum_{t=1}^{T} \sum_{i=1}^{N^t} \left[\log(\psi(\beta^t)) + \log(\phi(x_i^t, y_i^t)) + \log(\pi_x(x_i^t)) + \sum_{(i,j)\in\mathcal{E}^t} \log(\varphi(x_i^t, x_j^t)) \right] - \log(\bar{Z}(\Theta^t)) - \log(Z_{\beta^t})$$

$$= \underset{M}{\operatorname{argmax}} \sum_{t=1}^{T} \sum_{i=1}^{N^t} \left[\log(\psi(\beta^t)) + \log(\phi(x_i^t, y_i^t)) \right] - \log(Z_{\beta^t})$$

$$= \underset{M}{\operatorname{argmin}} \sum_{t=1}^{T} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2} + \frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2} \right] + \log((2\pi(\sigma_y^t)^2)^{\frac{K}{2}} \det(M^\top M)^{-\frac{1}{2}})$$

$$= \underset{M}{\operatorname{argmin}} \sum_{t=1}^{T} \sum_{i=1}^{N^t} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} \right] - \frac{1}{2} \log(\det(M^\top M))$$

To optimize this expression with respect to $\mathbf{M}$, we compute the derivative for a specific sample t and node i:

$$\frac{\partial}{\partial M}\left[\frac{\|y_i^t - M(x_i^t+\beta^t)\|_2^2}{2(\sigma_y^t)^2} - \frac{1}{2}\log(\det(M^\top M))\right]$$

$$= \frac{\partial}{\partial M}\left[\frac{(y_i^t - M(x_i^t+\beta^t))^\top(y_i^t - M(x_i^t+\beta^t))}{2(\sigma_y^t)^2} - \frac{1}{2}\log(\det(M^\top M))\right]$$

$$= \frac{\partial}{\partial M}\left[\frac{(y_i^t)^\top(y_i^t) - (y_i^t)^\top(Mx_i^t) - (y_i^t)^\top(M\beta^t) - (Mx_i^t)^\top(y_i^t) + (Mx_i^t)^\top(Mx_i^t) + (Mx_i^t)^\top(M\beta^t)}{2(\sigma_y^t)^2}\right.$$
$$\left. - \frac{(M\beta^t)^\top(y_i^t) + (M\beta^t)^\top(Mx_i^t) + (M\beta^t)^\top(M\beta^t)}{2(\sigma_y^t)^2} - \frac{1}{2}\log(\det(M^\top M))\right]$$

$$= \frac{-(y_i^t)(x_i^t)^\top - (y_i^t)(\beta^t)^\top - (y_i^t)(x_i^t)^\top + 2M(x_i^t)(x_i^t)^\top + 2M(x_i^t)(\beta^t)^\top - (\beta^t)(y_i^t)^\top}{2(\sigma_y^t)^2}$$
$$+ \frac{2M(\beta^t)(x_i^t)^\top + 2M(\beta^t)(\beta^t)^\top}{2(\sigma_y^t)^2} - M(M^\top M)^{-1}$$

$$= \frac{-(y_i^t)(x_i^t)^\top - (y_i^t)(\beta^t)^\top + M(x_i^t)(x_i^t)^\top + M(x_i^t)(\beta^t)^\top + M(\beta^t)(x_i^t)^\top + M(\beta^t)(\beta^t)^\top}{(\sigma_y^t)^2}$$
$$- M(M^\top M)^{-1}$$

We can use this derivative to perform gradient descent to minimize the expression. Proximal Nesterov's Accelerated Gradient method was later used heuristically to solve the problem.

2.5.3 Optimization of the Variance Parameter $(\sigma_y^t)^{-1}$

For the sample-specific variance parameter $(\sigma_y^t)^{-1}$, we formulate:

$$\widehat{(\sigma_y^t)}^{-1} = \underset{(\sigma_y^t)^{-1}}{\operatorname{argmax}} \frac{1}{\bar{Z}(\Theta^t)}\frac{1}{Z_{\beta^t}}\prod_{i=1}^{N^t}\left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t)\prod_{(i,j)\in\mathcal{E}^t}\varphi(x_i^t, x_j^t)\right]$$

$$= \underset{(\sigma_y^t)^{-1}}{\operatorname{argmax}} \log\left(\prod_{i=1}^{N^t}\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t)\prod_{(i,j)\in\mathcal{E}^t}\varphi(x_i^t, x_j^t)\right) - \log(\bar{Z}(\Theta^t)) - \log(Z_{\beta^t})$$

$$= \underset{(\sigma_y^t)^{-1}}{\operatorname{argmax}} \sum_{i=1}^{N^t}\left[\log(\psi(\beta^t)) + \log(\phi(x_i^t, y_i^t)) + \log(\pi_x(x_i^t)) + \sum_{(i,j)\in\mathcal{E}^t}\log(\varphi(x_i^t, x_j^t))\right] - \log(\bar{Z}(\Theta^t)) - \log(Z_{\beta^t})$$

$$= \underset{(\sigma_y^t)^{-1}}{\operatorname{argmin}} \sum_{i=1}^{N^t}\left[\frac{\|y_i^t - M(x_i^t+\beta^t)\|_2^2}{2(\sigma_y^t)^2}\right] + \frac{G}{2}\log(2\pi(\sigma_y^t)^2) + \frac{K}{2}\log(2\pi(\sigma_y^t)^2) + \log(\det(M^\top M)^{-\frac{1}{2}})$$

Following similar steps we can compute the partial derivative of the expression for a specific node i with respect to $(\sigma_y^t)^{-1}$ while only focusing on the terms that depend on $(\sigma_y^t)^{-1}$:

$$\frac{\partial}{\partial(\sigma_y^t)^{-1}}\left[\sum_{i=1}^{N^t}\left[\frac{\|y_i^t - M(x_i^t+\beta^t)\|_2^2}{2(\sigma_y^t)^2}\right] + \frac{G}{2}\log(2\pi(\sigma_y^t)^2) + \frac{K}{2}\log(2\pi(\sigma_y^t)^2)\right]$$

$$\implies \frac{\partial}{\partial(\sigma_y^t)^{-1}}\left[\frac{\|y_i^t - M(x_i^t+\beta^t)\|_2^2}{2(\sigma_y^t)^2} + \frac{G}{2}(\log(2\pi) - \log((\sigma_y^t)^2)) + \frac{K}{2}(\log(2\pi) - \log((\sigma_y^t)^2))\right]$$

$$= \frac{\|y_i^t - M(x_i^t+\beta^t)\|_2^2}{(\sigma_y^t)} - G(\sigma_y^t) - K(\sigma_y^t)$$

Computing the derivative with respect to $(\sigma_y^t)^{-1}$ and setting it to zero:

$$0 = \sum_{i=1}^{N^t}\left[\frac{\|y_i^t - M(x_i^t+\beta^t)\|_2^2}{(\sigma_y^t)}\right] - N^t G(\sigma_y^t) - N^t K(\sigma_y^t)$$

$$= \sum_{i=1}^{N^t}\left[\|y_i^t - M(x_i^t+\beta^t)\|_2^2\right] - (N^t G + N^t K)(\sigma_y^t)^2$$

Solving for $(\sigma_y^t)^{-1}$:

$$\widehat{(\sigma_y^t)}^{-1} = \sqrt{\frac{1}{(N^t G + N^t K)}\sum_{i=1}^{N^t}[\|y_i^t - M(x_i^t+\beta^t)\|_2^2]}$$

This closed-form solution provides the optimal value for the variance parameter.

2.5.4 Optimization of the Spatial Interaction Matrix Λ^t

For the sample-specific spatial interaction matrix $\mathbf{\Lambda}^t$, our optimization is:

$$\widehat{\Lambda^t} = \underset{\Lambda^t}{\text{argmax}} \frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z_{\beta^t}} \prod_{i=1}^{N^t} [\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t)] P(\Lambda^t)$$

$$= \underset{\Lambda^t}{\text{argmax}} \log \left[\prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right] \right] - \log(\bar{Z}(\Theta^t)) - \log(Z_{\beta^t}) + \log(P(\Lambda^t))$$

$$= \underset{\Lambda^t}{\text{argmax}} \sum_{i=1}^{N^t} \left[\sum_{(i,j)\in\mathcal{E}^t} \log(\varphi(x_i^t, x_j^t)) \right] - \log(\bar{Z}^X(\Theta^t)) + \log(P(\Lambda^t))$$

$$= \underset{\Lambda^t}{\text{argmin}} \sum_{i=1}^{N^t} \left[\sum_{(i,j)\in\mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{||x_j^t||_1}) \right] + \log(\bar{Z}^X(\Theta^t)) - \log(P(\Lambda^t))$$

Assuming a Gaussian prior on the magnitude and the difference in average spatial affinity for $\mathbf{\Lambda}^t$:

$$P(\Lambda^t) \propto \exp(-\lambda_\Lambda \|\Lambda^t\|_F^2 + \lambda_{\bar{\Lambda}} \|\Lambda^t - \bar{\Lambda}\|_F^2)$$

where $\|\cdot\|_F$ denotes the Frobenius norm, λ_Λ and $\lambda_{\bar{\Lambda}}$ are regularization parameters, and $\bar{\mathbf{\Lambda}}$ is a prior mean matrix.

Therefore the final optimization problem becomes:

$$\hat{\Lambda}^t = \underset{\Lambda^t}{\text{argmin}} \sum_{i=1}^{N^t} \left[\sum_{(i,j)\in\mathcal{E}^t} \frac{(x_i^t)^\top}{\|x_i^t\|_1} \Lambda^t \frac{(x_j^t)}{||x_j^t||_1} \right] + \log(\bar{Z}^X(\Theta^t)) + \lambda_\Lambda \|\Lambda^t\|_F^2 + \lambda_{\bar{\Lambda}} \|\Lambda^t - \bar{\Lambda}\|_F^2$$

This optimization problem is solved using the Adam optimizer, which efficiently handles the potentially complex landscape of the objective function.

2.6 Derivation of Optimization with β^t Prior

After implementing the derivation described above, we found the formulation leading to the embeddings no longer capturing the biological signal and rather all of the signal to be in the introduced batch effect term. Therefore, we suggest the introduction of a direct regularization on the $\boldsymbol{\beta^t}$ through a prior similar to PyLiger [8] (e.g. $\pi_\beta(\beta^t) = \exp(-\lambda_\beta \|M\beta^t\|_2^2)$). This approach would necessitate revisions to our mathematical formulation and optimization procedure, described below.

With the introduction of this new prior on $\boldsymbol{\beta^t}$, our new likelihood function with using the Hammersley-Clifford Theorem is represented as:

$$P(Y^t, X^t, \beta^t | \Theta^t) = \frac{1}{Z(\Theta^t)} \prod_{i=1}^{N^t} \left[\phi(x_i^t, y_i^t, \beta^t)\pi_x(x_i^t)\pi_\beta(\beta^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right]$$

2.6.1 Derivation of the Latent Space (X^t) Optimization

Note that since we are still utilizing the alternating coordinate descent optimization procedure, our optimization of $\boldsymbol{X^t}$ doesn't change as a result of this new prior and follows as derived in **Section** 2.3.

2.6.2 Derivation of the Batch Effect (β^t) Optimization

For the optimization of the batch effect term $\boldsymbol{\beta^t}$, we once again can employ the MAP estimate:

$$\hat{\beta}^t = \underset{\beta^t \in \mathbb{R}^{T \times K}}{\operatorname{argmax}} P(\beta^t | X^t, Y^t, \Theta^t)$$

$$= \underset{\beta^t}{\operatorname{argmax}} P(Y^t, X^t, \beta^t | \Theta^t) P(\Lambda^t)$$

$$= \underset{\beta^t}{\operatorname{argmin}} [-\log(P(Y^t, X^t, \beta^t | \Theta^t)) - \log(P(\Lambda^t))]$$

To simplify the optimization, we employ the mean-field approximation described in **Eq.** 2.4:

$$P(Y^t, X^t, \beta^t | \Theta^t) \approx P(\beta^t | Y^t, X^t, \Theta^t) \prod_{i=1}^{N^t} P(y_i^t, x_i^t | \beta^t, \Theta^t, X_{-i}^t, Y_{-i}^t)$$

$$= \frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z'_{\beta^t}} \prod_{i=1}^{N^t} [\psi(\beta^t) \phi(x_i^t, y_i^t) \pi_x(x_i^t) \pi_\beta(\beta^t) \prod_{(i,j) \in \mathcal{E}^t} \varphi(x_i^t, x_j^t)]$$

This leads to the following optimization problem for $\boldsymbol{\beta^t}$:

$$\hat{\beta}^t = \underset{\beta^t}{\operatorname{argmin}} \left[-\log \left(\frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z_{\beta^t}} \prod_{i=1}^{N^t} \left[\psi(\beta^t) \phi(x_i^t, y_i^t) \pi_x(x_i^t) \pi_\beta(\beta^t) \prod_{(i,j) \in \mathcal{E}^t} \varphi(x_i^t, x_j^t) \right] \right) \right]$$

$$= \underset{\beta^t}{\operatorname{argmin}} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2} + \lambda_\beta \|M\beta^t\|_2^2 \right]$$

$$= \underset{\beta^t}{\operatorname{argmin}} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1 + 2\lambda_\beta (\sigma_y^t)^2 \|M\beta^t\|_2^2}{2(\sigma_y^t)^2} \right]$$

For a specific node i, this becomes:

$$\beta^t = \underset{\beta^t}{\operatorname{argmin}} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1 + 2\lambda_\beta (\sigma_y^t)^2 \|M\beta^t\|_2^2}{2(\sigma_y^t)^2} \right]$$

The Hessian of this objective is given by

$$\frac{\partial^2}{\partial(\beta^t)^2}\left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top(M\beta^t)\|_1 + 2\lambda_\beta(\sigma_y^t)^2\|M\beta^t\|_2^2}{2(\sigma_y^t)^2}\right]$$

$$=\frac{\partial^2}{\partial(\beta^t)^2}\left[\frac{(M\beta^t)^\top(M\beta^t) - 2(y_i^t - Mx_i^t)^\top(M\beta^t) + 2\lambda_\beta(\sigma_y^t)^2(M\beta^t)^\top(M\beta^t)}{2(\sigma_y^t)^2}\right]$$

$$=\frac{\partial}{\partial\beta^t}\left[\frac{2(M^\top M)\beta^t - 2(y_i^t)^\top M + 2(x_i^t)^\top(M^\top M) + 2\lambda_\beta(\sigma_y^t)^2(M^\top M)\beta^t}{2(\sigma_y^t)^2}\right]$$

$$=\frac{M^\top M + \lambda_\beta(\sigma_y^t)^2(M^\top M)}{(\sigma_y^t)^2}$$

Since $\mathbf{M} \in \mathbb{R}_+^{G\times K}$, this Hessian is positive definite, meaning we can again employ the same proximal Nesterov's Accelerated Gradient method for optimization

2.6.3 Derivation of Parameters (Θ^t) Optimization

To derive the optimization of the parameters in this re-defined model we need to follow similar computations that are described in **Section** 2.5.

Calculation of Partition Function

For the optimization of our parameters, we need to once again compute an analytic form for our new partition function:

$$Z(\Theta^t) = \int_{X^t,Y^t,\beta^t} P(Y^t, X^t, \beta^t|\Theta^t)\,\mathrm{d}X^t\,\mathrm{d}Y^t\,\mathrm{d}\beta^t$$

$$= \int_{X^t,Y^t}\prod_{i=1}^{N^t} P(y_i^t, x_i^t|\beta^t, \Theta^t, X_{-i}^t, Y_{-i}^t)\,\mathrm{d}X^t\,\mathrm{d}Y^t \int_{\beta^t} P(\beta^t|Y^t, X^t, \Theta^t)\,\mathrm{d}\beta^t$$

$$= \int_{X^t,Y^t}\prod_{i=1}^{N^t}\left[\phi(x_i^t, y_i^t)\pi_x(x_i^t)\prod_{(i,j)\in\mathcal{E}^t}\varphi(x_i^t, x_j^t)\right]\mathrm{d}X^t\,\mathrm{d}Y^t \int_{\beta^t}\psi(\beta^t)\pi_\beta(\beta^t)\,\mathrm{d}\beta^t$$

$$= \bar{Z}(\Theta^t)Z'_{\beta^t}$$

Therefore, the computation of Z'_{β^t} follows as:

$$
\begin{aligned}
Z'_{\beta^t} &= \int_{\beta^t} \psi(\beta^t)\pi_\beta(\beta^t)\,\mathrm{d}\beta^t \\
&= \int_{\beta^t} \exp\left(-\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^T(M\beta^t)\|_1}{2(\sigma_y^t)^2}\right)\exp\left(-\lambda_\beta\|M\beta^t\|_2^2\right)\mathrm{d}\beta^t \\
&= \int_{\beta^t} \exp\left(-\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^T(M\beta^t)\|_1 - 2\lambda_\beta(\sigma_y^t)^2\|M\beta^t\|_2^2}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t \\
&= \int_{\beta^t} \exp\left(-\frac{(M\beta^t)^\top(M\beta^t) - 2\sum_{i\in\mathcal{V}^t}(y_i^t - Mx_i^t)^\top(M\beta^t) - 2\lambda_\beta(\sigma_y^t)^2(M\beta^t)^\top(M\beta^t)}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t \\
&= \int_{\beta^t} \exp\left(-\frac{(1+2\lambda_\beta(\sigma_y^t)^2)(M\beta^t)^\top(M\beta^t) - 2\sum_{i\in\mathcal{V}^t}(y_i^t - Mx_i^t)^\top(M\beta^t)}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t
\end{aligned}
$$

Defining $\mathbf{c} = \sum_{i\in V^t}(y_i^t - \mathbf{M}x_i^t)$ and $\mathbf{a} = (1+2\lambda_\beta(\sigma_y^t)^2)^{\frac{1}{2}}$ which are constant with respect to $\boldsymbol{\beta}^t$, we have:

$$
\begin{aligned}
Z'_{\beta^t} &= \int_{\beta^t} \exp\left(-\frac{(aM\beta^t)^\top(aM\beta^t) - 2c^\top(M\beta^t)}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t \\
&= \int_{\beta^t} \exp\left(-\frac{(\beta^t)^\top(a^2M^\top M)\beta^t - 2(c^\top M)\beta^t)}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t
\end{aligned}
$$

By completing the square in the exponent:

$$
\begin{aligned}
Z'_{\beta^t} &= \int_{\beta^t} \exp\left(-\frac{(\beta^t - (a^2M^\top M)^{-1}(M^\top c))^\top(a^2M^\top M)(\beta^t - (a^2M^\top M)^{-1}(M^\top c)) - (M^\top c)^\top(a^2M^\top M)^{-1}(M^\top c)}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t \\
&\propto \int_{\beta^t} \exp\left(-\frac{(\beta^t - (a^2M^\top M)^{-1}(M^\top c))^\top(a^2M^\top M)(\beta^t - (a^2M^\top M)^{-1}(M^\top c))}{2(\sigma_y^t)^2}\right)\mathrm{d}\beta^t
\end{aligned}
$$

Similar to the computation in **Section** 2.5.1, this integral is a multidimensional Gaussian integral, which means we can compute a closed-form solution:

$$
\begin{aligned}
Z'_{\beta^t} &= (2\pi(\sigma_y^t)^2)^{\frac{K}{2}}\det(a^2M^\top M)^{-\frac{1}{2}} \\
&= (2\pi(\sigma_y^t)^2)^{\frac{K}{2}}(a^2)^{-\frac{K}{2}}\det(M^\top M)^{-\frac{1}{2}} \\
&= (2\pi(\sigma_y^t)^2)^{\frac{K}{2}}(1+2\lambda_\beta(\sigma_y^t)^2)^{-\frac{K}{2}}\det(M^\top M)^{-\frac{1}{2}}
\end{aligned}
$$

Optimization of the Metagene Matrix M

With this new defined analytic form of the partition functions, the optimization of the metagene matrix, $\mathbf{M}$ changes to the following:

$$\widehat{M} = \underset{M}{\operatorname{argmax}} \prod_{t=1}^{T} \frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z'_{\beta^t}} \prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t)\pi_\beta(\beta^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right]$$

$$= \underset{M}{\operatorname{argmax}} \sum_{t=1}^{T} \log \left[\prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t)\pi_\beta(\beta^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right] \right] - \log(\bar{Z}(\Theta^t)) - \log(Z'_{\beta^t})$$

$$= \underset{M}{\operatorname{argmin}} \sum_{t=1}^{T} \sum_{i=1}^{N^t} \left[\frac{\|M\beta^t\|_2^2 - 2\|(y_i^t - Mx_i^t)^\top (M\beta^t)\|_1}{2(\sigma_y^t)^2} + \frac{\|y_i^t - Mx_i^t\|_2^2}{2(\sigma_y^t)^2} + \lambda_\beta \|M\beta^t\|_2^2 \right] + \log((2\pi(\sigma_y^t)^2)^{\frac{K}{2}} (1 + 2\lambda_\beta(\sigma_y^t)^2)^{-\frac{K}{2}} \det(M^\top M)^{-\frac{1}{2}})$$

$$= \underset{M}{\operatorname{argmin}} \sum_{t=1}^{T} \sum_{i=1}^{N^t} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_\beta \|M\beta^t\|_2^2 \right] - \frac{1}{2} \log(\det(M^\top M))$$

The derivative for a specific sample t and node i to optimize the expression with respect to $\mathbf{M}$ is:

$$\frac{\partial}{\partial M} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} + \lambda_\beta \|M\beta^t\|_2^2 - \frac{1}{2} \log(\det(M^\top M)) \right]$$

$$= \frac{\partial}{\partial M} \left[\frac{(y_i^t - M(x_i^t + \beta^t))^\top (y_i^t - M(x_i^t + \beta^t))}{2(\sigma_y^t)^2} + \lambda_\beta (M\beta^t)^\top (M\beta^t) - \frac{1}{2} \log(\det(M^\top M)) \right]$$

$$= \frac{\partial}{\partial M} [\frac{(y_i^t)^\top (y_i^t) - (y_i^t)^\top (Mx_i^t) - (y_i^t)^\top (M\beta^t) - (Mx_i^t)^\top (y_i^t) + (Mx_i^t)^\top (Mx_i^t) + (Mx_i^t)^\top (M\beta^t)}{2(\sigma_y^t)^2} - \frac{(M\beta^t)^\top (y_i^t) + (M\beta^t)^\top (Mx_i^t) + (M\beta^t)^\top (M\beta^t)}{2(\sigma_y^t)^2} + \lambda_\beta (M\beta^t)^\top (M\beta^t) - \frac{1}{2} \log(\det(M^\top M))]$$

$$= \frac{-(y_i^t)(x_i^t)^\top - (y_i^t)(\beta^t)^\top - (y_i^t)(x_i^t)^\top + 2M(x_i^t)(x_i^t)^\top + 2M(x_i^t)(\beta^t)^\top - (\beta^t)(y_i^t)^\top}{2(\sigma_y^t)^2} + \frac{2M(\beta^t)(x_i^t)^\top + 2M(\beta^t)(\beta^t)^\top}{2(\sigma_y^t)^2} + 2M(\beta^t)(\beta^t)^\top - M(M^\top M)^{-1}$$

$$= \frac{-(y_i^t)(x_i^t)^\top - (y_i^t)(\beta^t)^\top + M(x_i^t)(x_i^t)^\top + M(x_i^t)(\beta^t)^\top + M(\beta^t)(x_i^t)^\top + M(\beta^t)(\beta^t)^\top}{(\sigma_y^t)^2} + 2M(\beta^t)(\beta^t)^\top - M(M^\top M)^{-1}$$

Like above we can use this derivative to perform gradient descent or Proximal Nesterov's Accelerated Gradient (NAG) heuristically to solve the problem.

Optimization of the Variance Parameter $(\sigma_y^t)^{-1}$

For the sample-specific variance parameter $(\sigma_y^t)^{-1}$, we formulate:

$$\widehat{(\sigma_y^t)}^{-1} = \underset{(\sigma_y^t)^{-1}}{\operatorname{argmax}} \frac{1}{\bar{Z}(\Theta^t)} \frac{1}{Z'_{\beta^t}} \prod_{i=1}^{N^t} \left[\psi(\beta^t)\phi(x_i^t, y_i^t)\pi_x(x_i^t)\pi_\beta(\beta^t) \prod_{(i,j)\in\mathcal{E}^t} \varphi(x_i^t, x_j^t) \right]$$

$$= \underset{(\sigma_y^t)^{-1}}{\operatorname{argmax}} \sum_{i=1}^{N^t} \left[\log(\psi(\beta^t)) + \log(\phi(x_i^t, y_i^t)) + \log(\pi_x(x_i^t)) + \log(\pi_\beta(\beta^t)) + \sum_{(i,j)\in\mathcal{E}^t} \log(\varphi(x_i^t, x_j^t)) \right] - \log(\bar{Z}(\Theta^t)) - \log(Z'_{\beta^t})$$

$$= \underset{(\sigma_y^t)^{-1}}{\operatorname{argmin}} \sum_{i=1}^{N^t} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} \right] + \frac{G}{2}\log(2\pi(\sigma_y^t)^2) + \frac{K}{2}\log(2\pi(\sigma_y^t)^2) - \frac{K}{2}\log(1 + 2\lambda_\beta(\sigma_y^t)^2) + \log(\det(M^\top M)^{-\frac{1}{2}})$$

Following similar steps we can compute the partial derivative of the expression for a specific node i with respect to $(\sigma_y^t)^{-1}$ while only focusing on the terms that depend on $(\sigma_y^t)^{-1}$:

$$\frac{\partial}{\partial(\sigma_y^t)^{-1}} \left[\sum_{i=1}^{N^t} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} \right] + \frac{G}{2}\log(2\pi(\sigma_y^t)^2) + \frac{K}{2}\log(2\pi(\sigma_y^t)^2) - \frac{K}{2}\log(1 + 2\lambda_\beta(\sigma_y^t)^2) \right]$$

$$\implies \frac{\partial}{\partial(\sigma_y^t)^{-1}} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{2(\sigma_y^t)^2} + \frac{G}{2}(\log(2\pi) - \log((\sigma_y^t)^2)) + \frac{K}{2}(\log(2\pi) - \log((\sigma_y^t)^2)) - \frac{K}{2}(\log(2\lambda_\beta) - \log((\sigma_y^t)^2)) \right]$$

$$= \frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{(\sigma_y^t)} - G(\sigma_y^t) - K(\sigma_y^t) + K(\sigma_y^t)$$

$$= \frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{(\sigma_y^t)} - G(\sigma_y^t)$$

Setting this derivative to zero, and solving for $(\sigma_y^t)^{-1}$ we get a closed-form solution that provides the optimal value for the variance parameter

$$0 = \sum_{i=1}^{N^t} \left[\frac{\|y_i^t - M(x_i^t + \beta^t)\|_2^2}{(\sigma_y^t)} \right] - N^t G(\sigma_y^t)$$

$$\widehat{(\sigma_y^t)}^{-1} = \sqrt{\frac{1}{N^t G} \sum_{i=1}^{N^t} [\|y_i^t - M(x_i^t + \beta^t)\|_2^2]}$$

Optimization of the Spatial Interaction Matrix Λ^t

The optimization of the spatial interaction matrix $\boldsymbol{\Lambda^t}$ is not affected by this introduction of the batch effect term prior and thus the optimization problem stays the same as described in **Section** 2.5.4.

This derivation suggests a possible solution to the challenge later described where the burden fell entirely on the batch effect term $\boldsymbol{\beta^t}$. By introducing an explicit regularization term $\pi_\beta(\beta^t) = \exp(-\lambda_\beta \|M\beta^t\|_2^2)$, we provide a mechanism to control the magnitude of the batch effect parameters during optimization. This approach differs from the above formulation by ensuring that the model cannot simply shift all variation into the batch effect term while allowing embeddings to collapse. The modified optimization framework maintains the same alternating coordinate descent structure while encouraging a more balanced distribution of variation between the biological embeddings $\boldsymbol{X^t}$ and the batch effect $\boldsymbol{\beta^t}$. Implementation of this regularized approach represents a promising direction for achieving more robust integration that preserves both the integrity of biological signals and the effectiveness of batch effect correction.

2.7 Future Directions

During model implementation, we encountered significant challenges with the prior distribution formulation. In the original Popari framework, the prior on node representations was a simple exponential prior, given by $\pi_x(x_i^t) = \exp(-\lambda_x \|x_i^t\|_1)$ designed to constrain the magnitudes of x_i^t for simplex projection. However, this formulation led to a problematic outcome: embeddings consistently collapsed to zero during training, shifting the optimization burden entirely to the batch effect term β^t. While this technically resulted in data integration, it did so by effectively bypassing the biological signal rather than properly aligning it.

To address this limitation, we proposed a new prior that incorporates cross-dataset information (see **Section** 2.1), computing an average across all datasets and spatial entities. This prior encourages embeddings to remain similar to the average embedding profile across all datasets, meaning that although it is possible for the embeddings to collapse to zero, it is also possible for the embeddings to integrate at a non-zero value which would allow for batch-specific variation. This approach while maintaining biological relevance integration, had the additional benefit of minimal disruption to our original mathematical formulation, as it remains approximately constant with respect to x_i^t. This is because we are computing the average across all other datasets except the dataset of interest t.

Although this improved prior demonstrated better performance, further enhancements are possible. A particularly promising future direction involves developing a prior that explicitly encourages embeddings to align with their nearest neighbors from different datasets, which would likely correspond to similar cell types. This approach would simultaneously preserve biological information while promoting batch integration. Since our optimization approach requires the prior to be approximately constant with respect to x_i^t, implementing this enhancement would require careful formulation, but it represents a relatively contained modification of the current framework.

Chapter 3

BatchBlend's Performance on mSRT Data

We conducted a comprehensive evaluation of BatchBlend's batch effect correction capabilities through a series of controlled experiments using simulated mSRT data. In developing and evaluating these benchmarks, we first sought to compare BatchBlend against the original Popari framework and then benchmark the methodology against established batch effect correction methods discussed previously, including PyLiger, scVI and STAligner.

To generate synthetic datasets for this purporse, we extended the mSRT simulator introduced in Popari, by introducing parameters for simulating batch effect of varying magnitudes for different samples. This simulation framework enabled us to create "realistic" mSRT data that exhibit technical artifacts observed in real-world SRT experiments. By using such synthetic mSRT data with user-specified batch effects, we were able to quantitatively assess the correction performance in a controlled environment. Using the ground truth simulated cell type and batch labels, we also assessed how effectively each method removed technical variation while still preserving the true biological signal.

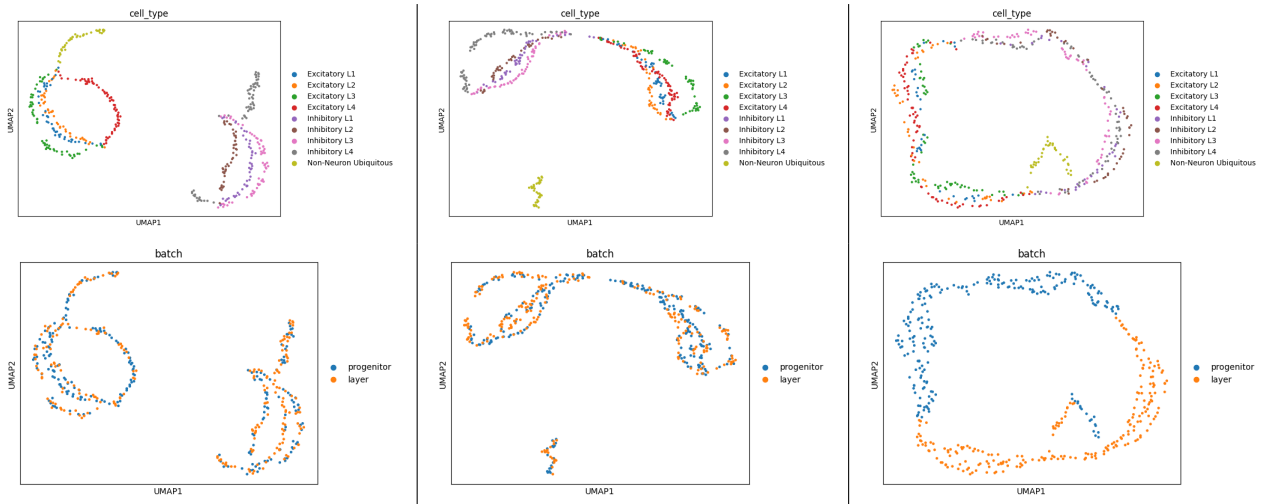
3.1 Simulation of mSRT Data with Batch Effect

In the context of the Popari simulation framework, simulated mSRT expression is drawn from a generative, metagene-based model, whereas simulated spatial coordinates are generated according to highly-configurable user inputs. Popari samples a metagene matrix M for all samples, while it generates cell embeddings independently X^t for each sample. Subsequently, for each sample these quantities are multiplied together to generate expression values Y^t:

$$Y^t := M \times X^t + \text{noise term}$$

In particular, $M \in \mathbb{R}_+^{G \times K}$ is generated by sequentially sampling gene weights from a Gamma distribution while also ensuring a proportion of the gene weights are shared with previous metagenes. Then, within each spatial domain, each spatial entity is assigned a cell-type label, randomly sampled from a user-specified cell type distribution. Given these labels, $X^t \in \mathbb{R}_+^{K \times N^t}$ is then generated in a spatially-informed manner. Their product yields the gene expression distribution parameters $\tilde{Y}^t \in \mathbb{R}_+^{G \times N^t}$ for each cell. From these parameters, the expression value for each entity i, y_i^t, is then sampled from a truncated multivariate Gaussian with mean given by $\tilde{y}_i^t$ and covariance matrix $\sigma_y^t I \in \mathbb{R}_+^{G \times G}$.

For BatchBlend, we introduce an additive batch effect parameter by sampling a batch effect vector



(a) No simulated batch effect

(b) Simulated batch effect strength ($\sigma_\beta^t = 1$)

(c) Simulated batch effect strength ($\sigma_\beta^t = 2$)

Figure 3.1: Visualization of cell type and batch classifications under different levels of simulated batch effect based on the scale of the standard deviation of the truncated gaussian for 2 datasets each where $\hat{K} = 50\%$. Top row shows cell type classification, while bottom row shows batch classification with increasing batch effect strength from left to right.

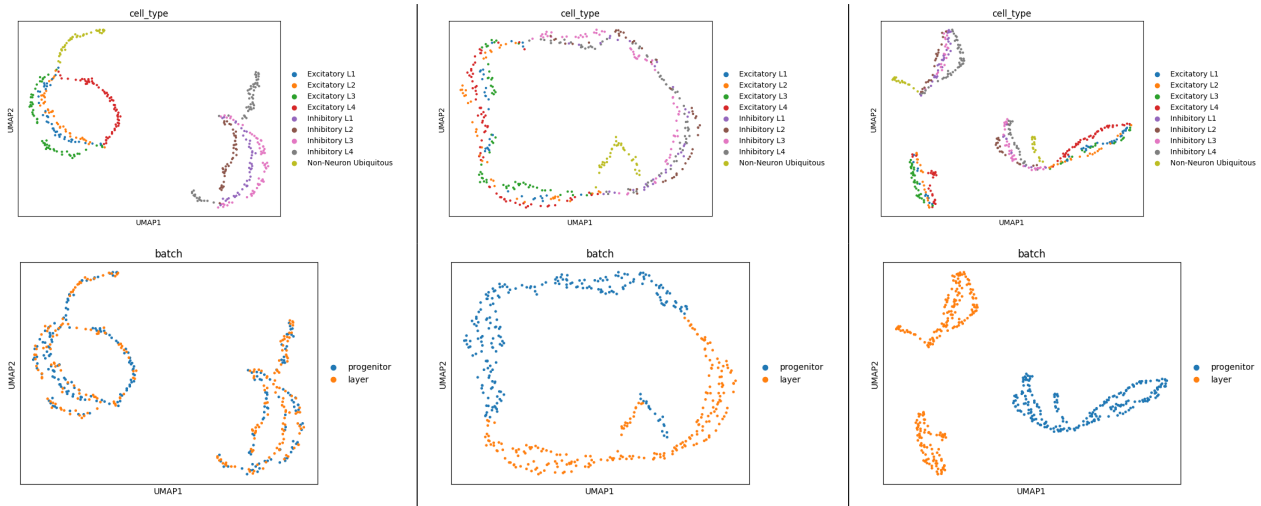
$\beta^t \in \mathbb{R}_+^K$ for each sample t according to a multivariate truncated Gaussian distribution with standard deviation, σ_β^t. However, to further control the batch effect strength, only a subset of the K metagenes ($\hat{K}$) expressed an added batch effect. For the other metagenes, the batch effect vector is assigned a value of 0. This vector was directly added to X^t, representing a batch-specific linear shift in the data. This yields the following mean parameters/generative mode:

$$Y^t = M(X^t + B^t) + \text{noise term},$$

which aligns closely with the BatchBlend paradigm. We then selected hyperparameters for the simulation data to ensure that this simulated batch effect is of a similar scale to that found in real mSRT data but still demonstrates a significant batch effect. We visualized the batch effect using Uniform Manifold Approximation and Projection (UMAP) plots. This non-linear dimensionality reduction allows us to analyze the data based on technical variation via batch labels and biological preservation via cell type labels. Batch effect can be observed since when increasing the intensity of the batch effect, data points cluster by batch identity rather than by cell type. In the absence of batch effect, points are mixed by batch but clustered by cell type, while with strong batch effect, distinct regions become dominated by specific batches, compromising the biological signal.

As illustrated in **Fig.** 3.1, we analyzed the batch effect strength by varying the standard deviation (σ_β^t) of the truncated Gaussian distribution used for generation. Without simulated batch effect (**Fig.** 3.1a), cell types form distinct, well-defined clusters while batch information appears randomly distributed, indicating no batch effect across datasets. With a moderate batch effect ($\sigma_\beta^t = 1$, **Fig.** 3.1b), cell type boundaries begin to blur and batch-specific clusters emerge. At stronger batch effect levels ($\sigma_\beta^t = 2$, **Fig.** 3.1c), we observe substantial disruption of biological clustering patterns alongside pronounced batch-specific segregation.

Similarly, **Fig.** 3.2 illustrates the impact of varying the percentage of metagenes ($\hat{K}$) affected by batch effect. In the absence of batch effect (**Fig.** 3.2a), cell types remain clearly differentiated while batch information shows no clustering pattern. When batch effect, $\hat{K} = 50\%$ of metagenes (**Fig.** 3.2b), cell type boundaries become



(a) No batch effect

(b) Simulated batch effect ($\hat{K} = 50\%$)

(c) Simulated batch effect ($\hat{K} = 100\%$)

Figure 3.2: Visualization of cell type and batch classifications under different percentages of metagenes with batch effect strength ($\sigma_\beta^t = 2$) providing additive batch effect for 2 datasets. Top row shows cell type classification, while bottom row shows batch classification with increasing batch effect strength from left to right.

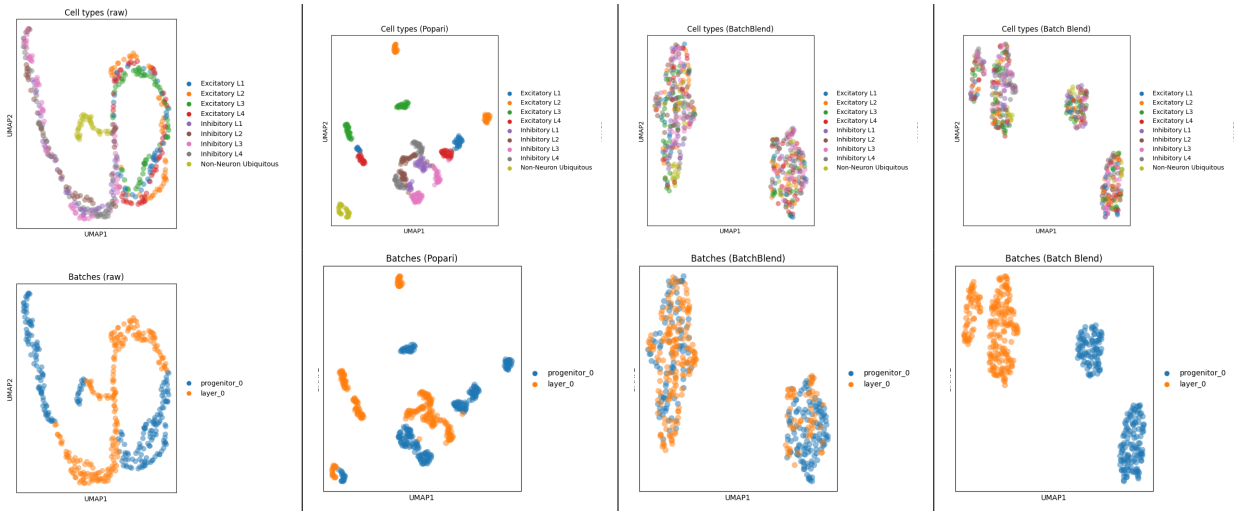
partially obscured as batch-specific clusters begin to form. With $\hat{K} = 100\%$ (**Fig.** 3.2c), biological signal is severely compromised as batch-specific clustering becomes the dominant organizational pattern in the data.

Based on these comprehensive simulations, we selected hyperparameters corresponding to 50% of metagenes ($\hat{K} = 50\%$) receiving an additive batch effect with a standard deviation of 2, as these parameters produced the strongest batch effect yet realistic conditions for evaluating batch effect correction performance. In addition as seen in **Fig.** 3.5, this was the set of hyperparameters of batch effect that showed the most significant drop off in batch correction and biological conservation scores in Popari which would be the gap that BatchBlend would hope to fill.

3.2 Evaluation of BatchBlend on Simulated mSRT Data

For qualitative evaluation of BatchBlend to other methods, we visualized the embeddings using UMAP plots overlayed with batch information and biological annotations. After successful batch effect correction, we expect the data to cluster primarily by cell type rather than by batch identity. For quantitative assessment, we used the standardized single-cell integration benchmarking suite (scIB) [17] to measure both batch effect removal efficacy and biological signal preservation.

To evaluate batch effect removal, we computed several metrics that address different aspects of integration quality. The Silhouette Batch score provides a global measure of batch mixing, with higher values indicating successful integration between batches. The Integration Local Inverse Simpson Index (iLISI) offers finer granularity by quantifying the effective number of batches represented in local neighborhoods after correction. Similarly, the k-nearest neighbor Batch Effect Test (kBET) evaluates batch mixing at the local level by assessing whether batch distribution in local neighborhoods matches the global batch distribution, with higher values indicating better mixing. Graph Connectivity determines whether cells from the same batch form disconnected subgraphs. Principal Component Regression (PCR) measures the correlation between principal components of



(a) Raw Simulated Data with Batch Effect

(b) POPARI Learned Embeddings

(c) BATCHBLEND Learned Embeddings

(d) BATCHBLEND with Added Batch Effect

Figure 3.3: Visualization of cell type and batch classifications of raw simulated data and the learned embeddings from POPARI and BATCHBLEND. The learned batch effect from optimization was added back to the embeddings of BATCHBLEND.

the integrated data and batch variables, where larger scores indicate greater differences in variance contribution before and after integration.

To evaluate biological signal preservation, we used multiple complementary metrics. The Isolated Labels Score quantifies how well cells of the same cell type remain grouped after integration, indicating the preservation of biologically relevant population structures. The kMeans Adjusted Rand Index (ARI) and kMeans Normalized Mutual Information (NMI) measure concordance between corrected data clusters and known biological cell types, with higher values indicating better preservation. The Silhouette Label metric assesses cell type separation after batch effect correction, where higher values indicate clearer boundaries between cell types. Cell-type Local Inverse Simpson's Index (cLISI) measures how effectively distinct cell types remain separated after batch correction.

Each of the scIB metrics is normalized and scaled to be between 0 and 1 where values closer to 1 suggest better performance. Optimal scores for each of the metrics comparing across methods were bolded to demonstrate better performance. Once each of the individual metrics are computed, the scIB framework also provides an aggregate score that combines batch correction efficiency and biological signal preservation, offering a comprehensive assessment of integration performance that balances the trade-off between removing technical variation and maintaining biological information.

3.3 BATCHBLEND's Batch Effect Correction Performance with Respect to POPARI

To assess BATCHBLEND's capacity for correcting batch effects, we employed a strong set of batch effect simulation parameters ($\hat{K} = 50\%$ and $\sigma_\beta^t = 2$) to generate five experimental datasets. Each dataset consisted of 2 batches containing 225 cells and 100 genes, providing sufficient complexity to evaluate the performance of the method

Table 3.1: Average Batch integration metrics for benchmark methods on simulated mSRT data. Bold values show better performance between BatchBlend and Popari; † indicates best overall value.

Method	Silhouette batch	iLISI	kBET	Graph connectivity	PCR comparison
PyLiger	0.486	0.489	0.259	0.526	0.964†
scVI	0.932†	0.836†	0.907†	0.383	0.909
STAligner	0.680	0.657	0.644	0.410	0.872
Popari	0.527	0.076	0.018	**0.781**†	0.373
BatchBlend	**0.833**	**0.442**	**0.578**	0.273	**0.547**

Table 3.2: Average Biological Conservation metrics for benchmark methods on simulated mSRT data. Bold values show better performance between BatchBlend and Popari; † indicates best overall value.

Method	Isolated labels	kMeans NMI	kMeans ARI	Silhouette label	cLISI
PyLiger	0.603	0.627	0.431	0.589	0.925
scVI	0.482	0.090	0.031	0.482	0.481
STAligner	0.406	0.271	0.123	0.411	0.658
Popari	**0.610**†	**0.651**†	**0.459**†	**0.605**†	**0.954**†
BatchBlend	0.480	0.061	0.012	0.479	0.512

against the original Popari framework. The scIB metrics were calculated for each dataset to evaluate biological conservation and batch integration, then averaged across the five experimental datasets to identify general trends for each method.

The optimization procedures for both methods begin with NMF training before transitioning to the full NMF-HMRF procedure. However, our experimental analyses revealed that BatchBlend demonstrated a diminished performance in maintaining the learned batch effect parameters during the NMF-HMRF optimization phase compared to its efficiency during the preliminary NMF phase. Therefore, we restricted BatchBlend's training to 50 iterations of NMF optimization, while Popari underwent 10 NMF iterations followed by 50 NMF-HMRF iterations. This limitation highlights a potential future direction of enhancing BatchBlend's NMF-HMRF optimization or, alternatively, positioning BatchBlend as a preprocessing step prior to Popari's NMF-HMRF phase to leverage the complementary strengths of both approaches.

After optimization, we generated UMAP plots based on the PCA (Principal Component Analysis) of the embeddings to demonstrate BatchBlend's batch effect correction efficacy. PCA is used as an initial dimensionality reduction to truly capture the most variance signal in the data while reducing noise. **Fig.** 3.3c demonstrates that for an example dataset, **Fig.** 3.3a, BatchBlend successfully integrated points from different batches, as indicated by the mixing of the cells in different datasets in the batch classification plot. In addition to integrating cell embeddings, BatchBlend effectively modeled the batch effect. When the learned batch effect was introduced back to the learned embeddings, the dataset was once again separated by batch (see **Fig.** 3.3d). Note that the separation of the blue dataset into 2 clusters could be a consequence of the original raw data separating the same dataset into 2 clusters.

Popari, conversely, lacks an explicit batch effect correction mechanism in its framework. Therefore, as seen in **Fig.** 3.3b, the learned embeddings still seem to have batch-specific clustering rather than integration evidenced by the distinct dataset clusters in the batch classification plot. However, Popari demonstrates superior preservation of the biological signal seen by the well-defined cell type clusters shown in the cell type labeling of the learned embeddings. In fact, although the batches aren't as integrated as BatchBlend, Popari's representation

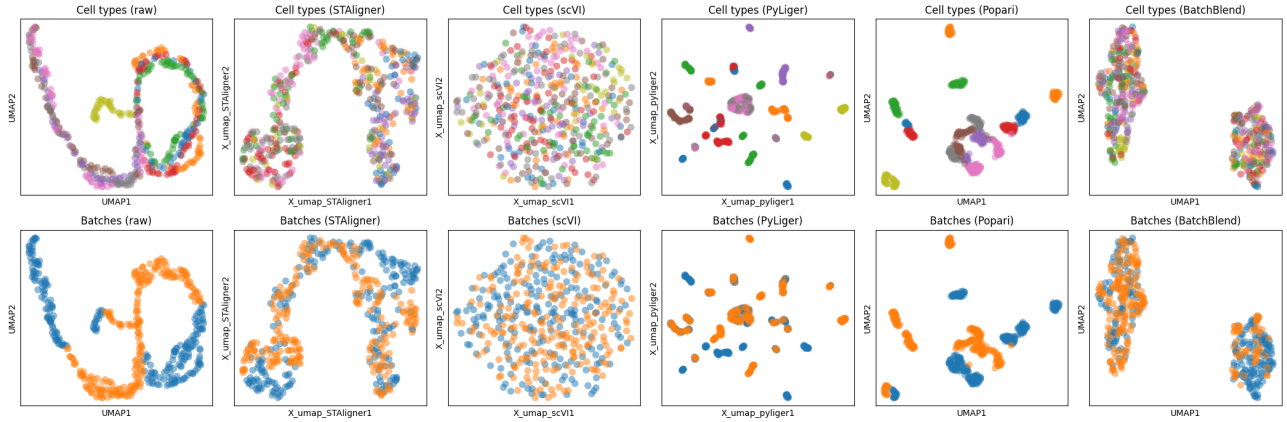


Figure 3.4: Benchmarking BatchBlend's batch effect correction against established batch effect correction methods, PyLiger, scVI and STAligner using cell type and batch classification to show integration of the data.

shows cell types from different datasets positioned relatively closer to each other (i.e., the Non-Neuron Ubiquitous cells).

This trade-off between biological signal preservation in Popari and batch effect correction in BatchBlend is also outlined in the quantitative metrics shown in **Table** 3.1 and **Table** 3.2. Quantitatively, BatchBlend performed better than Popari at batch integration. Particularly, BatchBlend seemed to perform better at mixing in local neighborhoods. While Popari far exceeded BatchBlend's biological conservation. This could perhaps be explained by the lack of NMF-HMRF training iterations with BatchBlend which may have been critical to Popari preserving the biological signal.

3.4 BatchBlend's Batch Effect Correction Performance with Respect to State-of-the-Art Methods

To comprehensively evaluate BatchBlend performance, we performed a comparative analysis against several established batch effect correction methods: PyLiger, scVI, and STAligner. This benchmarking allowed us to assess how BatchBlend performs relative to methods that are interpretable but not specifically designed for mSRT data (PyLiger), or methods that are specialized for mSRT data but lack interpretability (STAligner) or both (scVI).

As shown in **Table** 3.1, BatchBlend demonstrated competitive performance compared to some batch effect correction methods. BatchBlend performed well at global mixing based on the silhouette batch score. For local mixing, although the iLISI and kBET scores were lower than those of other established methods, BatchBlend still achieved reasonably high values. This is most apparent in PCR comparison, where the established methods changed the variance contribution more than BatchBlend, indicating they were more effective at removing batch effects. Overall, scVI and STAligner performed better at batch effect correction than the more interpretable methods. This was also observed in the UMAP (**Fig.** 3.4), where STAligner and scVI had clear integration of the different cells in two distinct datasets. Similarly, PyLiger's clusters can still be seen as batch-specific.

For preservation of the biological signal, BatchBlend shows competitive performance with respect to selected metrics, as seen in **Table** 3.1. PyLiger had similar performance to Popari where cell type conservation was preserved seen by the Isolated Labels, kMeans NMI, kMeans ARI metrics in **Table** 3.2 which exceeded

Table 3.3: Combined weighted aggregate score of the biological conservation and batch integration. Bold values show better performance between BatchBlend and Popari; † indicates best overall value.

Method	Batch Integration	Biological Conservation	Total
PyLiger	0.52	0.62	0.60†
scVI	0.78†	0.32	0.50
STAligner	0.48	0.37	0.47
Popari	0.40	**0.67**†	**0.54**
BatchBlend	**0.59**	0.31	0.40

BatchBlend's performance. This can also be seen in **Fig.** 3.4, where, like Popari, PyLiger had distinct cell type specific clusters formed in the UMAP with cell type annotation. However, BatchBlend performed competitively compared to STAligner and scVI in biological conservation, where these methods prioritized batch effect correction. This is evident in both the quantitative metrics and the UMAP visualization, where despite successful batch integration, specific cell types become difficult to distinguish in the embeddings. In fact, scVI seemed to project all cell embeddings into one cloud without any sense of a further cell-type-specific organization. This perhaps may be an indication of why scVI excels in the quantification of batch effect correction: since it compresses everything together in one space without considering the biological signal at all.

Comparison of BatchBlend with established batch correction methods is revealing a nuanced performance profile, as seen in **Table** 3.3. In batch correction, BatchBlend performs better than Popari and is similar to PyLiger but does not achieve the integration quality of scVI and STAligner, which may leverage their complex, non-interpretable architectures to better address technical variation. Similarly, in biological conservation, BatchBlend performs on par with scVI and STAligner but is limited in the preservation of the biological signals demonstrated by PyLiger and Popari. This balance of strengths and limitations positions BatchBlend's aggregate score below other methods.

3.5 Robustness Analysis of BatchBlend on Different Batch Effect Simulation Hyperparameters

While **Section** 3.3 and **Section** 3.4 examined specific hyperparameter settings, we wanted to systematically evaluate if the performance of BatchBlend is conditional to the hyperparameters picked in **Section** 3.2. Under varying batch effect conditions, we conducted a comprehensive robustness analysis by manipulating the two key simulation hyperparameters: batch effect strength and the percentage of metagenes affected by batch effect (see **Section** 3.1). In addition to the data generated for the previous sections, we generated datasets for five different random seeds with 225 cells and 100 genes for different choices of batch effect standard deviation values of the multivariate Gaussian $\sigma_\beta^t = \{0, 1, 1.5, 2, 3 \text{ and } 5\}$ and the percentage of metagenes to have simulated batch effect, $\hat{K} = \{0\%, 25\%, 50\%, 75\%, 100\%\}$.

The results (as seen in **Fig.** 3.5) reveal consistent patterns across methods. Batch effect correction and biological conservation performance significantly deteriorated as the strength of the batch effect increased and slightly deteriorated as the percentage of affected metagenes increases. This likely occurs because these methods were developed for existing technologies with less severe batch effects than those in our simulations. In particular, while Popari maintains superior preservation of biological signals in all combinations of parameters, its batch correction capability drops drastically with increasing batch effect strength, while BatchBlend still maintains the capacity to correct for batch effect.

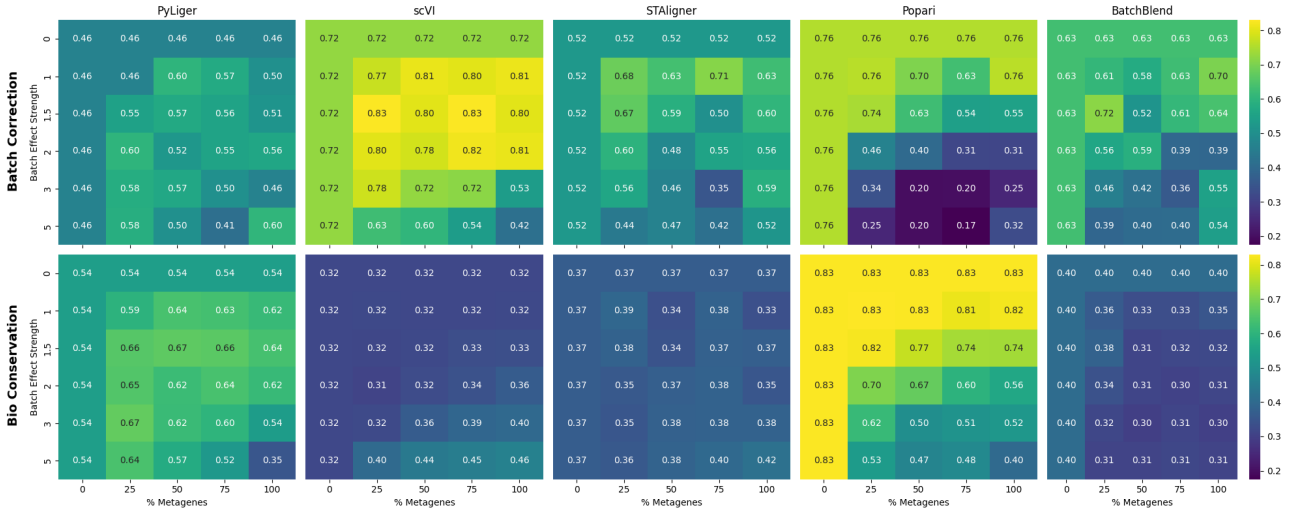


Figure 3.5: Average Batch Correction and Bio Conservation Metrics across varying simulated Batch Effect strength parameters (σ_β^t and $\hat{K}$). Note that for a batch effect strength of a multivariate Gaussian with $\sigma_\beta^t = 0$ and where $\hat{K} = 0\%$ of the metagenes are introduced with the added batch effect, the dataset would be the same since they all depict no batch effect introduced to the dataset. Hence, the scores are the same along these axes.

This is further supported by **Fig.** 3.6, where for a $\sigma_\beta^t = 2$, BatchBlend consistently outperformed Popari in batch effect correction. When 50% of the metagenes are introduced with batch effect, BatchBlend performed better than Popari at batch effect strengths $\sigma_\beta^t \geq 2$. This confirms our hypothesis that explicit modeling of batch effect through BatchBlend provides more robust correction compared to Popari when technical variation becomes more pronounced with batch effect strength. At the same time, across all percentages of batch effect metagenes and standard deviations for the multivariate gaussians, Popari performed better than BatchBlend. However, the difference in biological conservation gets smaller as batch effect strength decreases.

Compared to the other benchmark methods, BatchBlend demonstrates competitive performance relative to PyLiger and STAligner in batch effect correction of most parameter combinations, where even in some cases it outperforms these methods, as seen in the standard deviations in **Fig.** 3.7. In addition, similar to what is observed in **Fig.** 3.6, we see that the batch correction performance decreases as the simulated batch strength increases. However, as observed in the single case with batch effect strength of 2 and 50% of metagenes, scVI outperforms these methods by far in batch effect correction. However, as mentioned in **Section** 3.4, it is possible that this is because the method does not consider the retention of biological signal, which is seen in the methods corresponding values of biological conservation metric.

For biological conservation, across different choices of the batch effect simulation hyperparameters, the pattern still suggest that Popari and PyLiger preserve the biological signal better than BatchBlend, STAligner and scVI. However, as batch effect signal increases Popari and PyLiger preservation of biological signal decreases as seen in **Fig.** 3.7. Although worse than Popari and PyLiger, BatchBlend seems to do as well as scVI and STAligner, again highlighting the persistent tradeoff between batch effect removal and biological signal preservation.

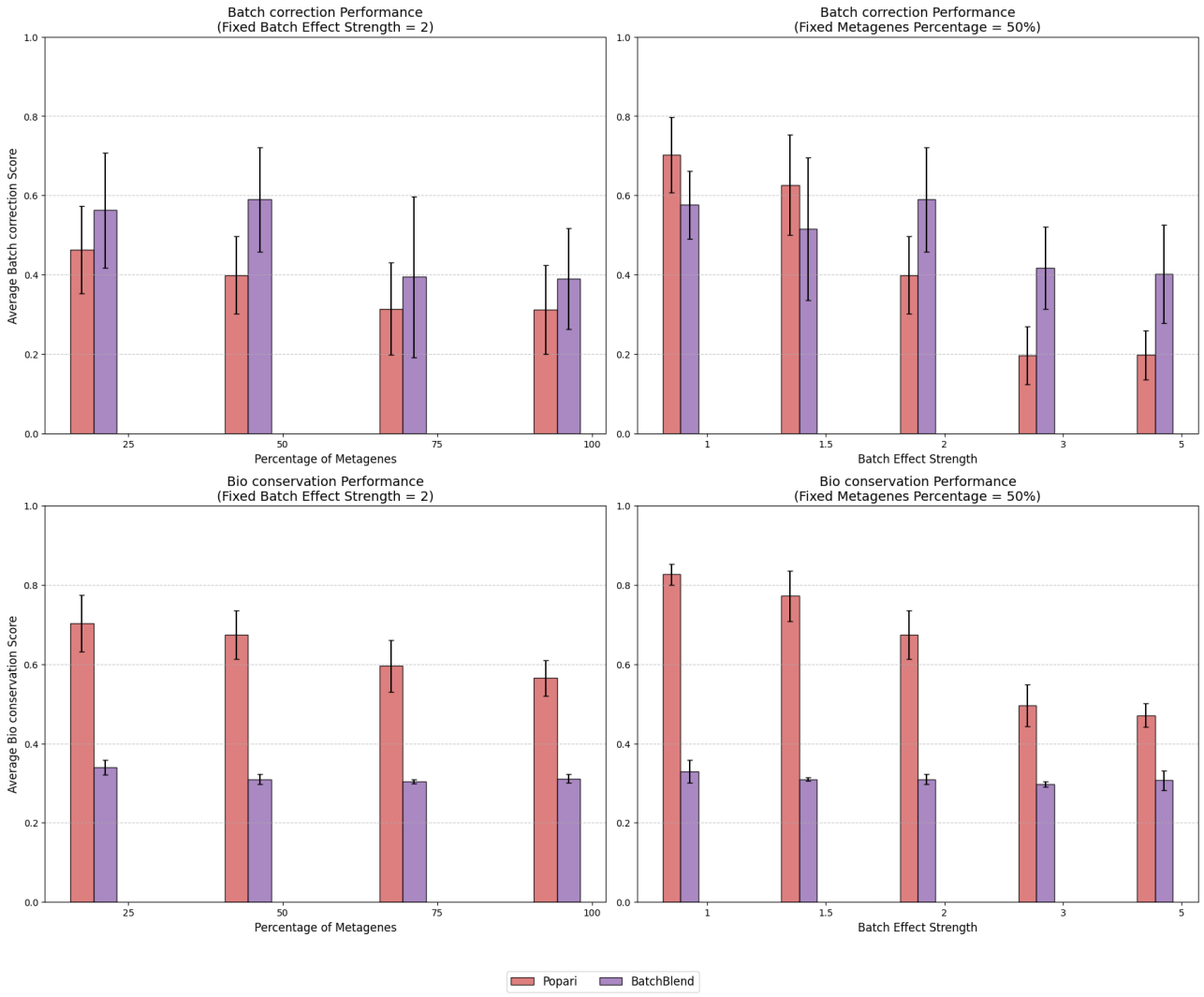


Figure 3.6: Batch Correction Score and Biological Conservation Scores for Selected Choice of Batch Effect Parameters comparing Popari and BatchBlend. The bar represents the average value across the five seeds with the error bars representing the standard deviation of the values.

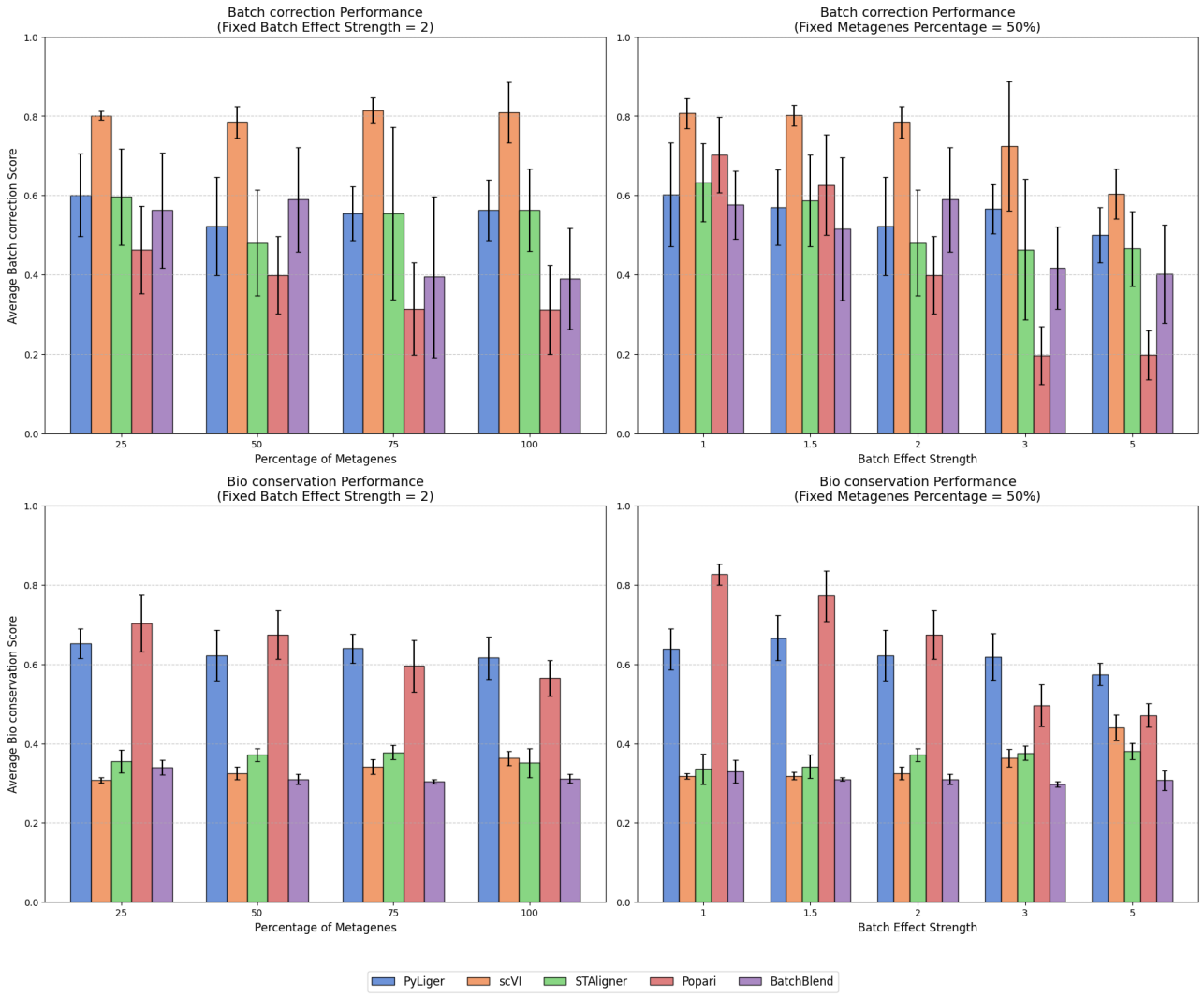
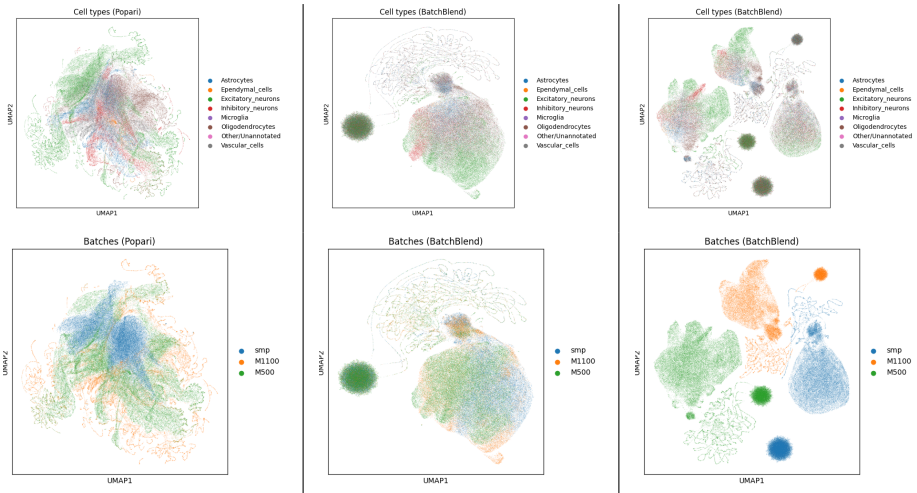


Figure 3.7: Batch Correction Score and Biological Conservation Scores for Selected Choice of Batch Effect Parameters comparing all benchmark methods to BatchBlend. The bar represents the average value across the five seeds with the error bars representing the standard deviation of the values.



(a) POPARI Learned Embeddings

(b) BATCHBLEND Learned Embeddings

(c) BATCHBLEND with Added Batch Effect

Figure 3.8: Visualization of cell type and batch classifications of raw coronal mice data and the learned embeddings from POPARI and BATCHBLEND. The learned batch effect from optimization was then added back to the latent space embeddings of BATCHBLEND.

3.6 Assessment of BATCHBLEND on Mice Brain Slices

To further analyze BATCHBLEND, we performed integration of high-quality mSRT datasets from three different spatial transcriptomics technologies. These include: (1) MERFISH data (M500) comprising 218 coronal mouse brain slices with 1,122 genes; (2) MERSCOPE (M1100) data consisting of 53 coronal slices from the whole mouse brain with 550 genes; and (3) STARmap (SMP) data containing 17 coronal slices with 1,022 genes. To perform integration of these technologies, we needed to subset the genes to only include the 258 overlapping genes. Then for three different seed runs of POPARI and BATCHBLEND, we performed a UMAP on the PCA of the embeddings and then performed scIB metrics on the batch and higher level annotations of cell types to assess batch correction integration.

When integrating these technologies, we observed clear separation by technology type, even while the data clustered by cell types. When performing analysis through POPARI (seen in **Fig.** 3.8a), we see some clear clustering of the data by cell types, but still separation by batches indicating that the batch signal is still strongly present in the dataset. BATCHBLEND demonstrates superior performance in this aspect, effectively integrating batches (X^t) as evidenced by the large cluster in **Fig.** 3.8b. When the learned batch effect is added back ($X^t + \beta^t$) in **Fig.** 3.8c, the resulting separation by spatial transcriptomic technology confirms that BATCHBLEND effectively captures technology-specific batch effects. This highlights again the power of this existing framework of the model. However, as seen in the simulated data, when looking at the cell type, the batch integration does seem to lose some of this biological signal. Unlike the simulation setting, there is some signal preservation, perhaps similar to POPARI. For example there does seem to be some clustering of the excitatory neurons of all three datasets seen in the bottom of the large cluster.

When quantifying the observed performance difference in the UMAPs, for batch integration (**Table** 3.4) and biological conservation (**Table** 3.5) we see a more fine-tuned aspect of the discrepancy in batch integration and biological conservation. The global mixing demonstrated by the Silhouette batch and PCR comparison scores show that BATCHBLEND successfully removes the batch-specific variance from the overall data structure. In

Table 3.4: Average batch integration metrics for each of the benchmark methods on coronal mice slices. Bold values show better performance between BatchBlend and Popari

Method	Silhouette batch	iLISI	kBET	Graph connectivity	PCR comparison
Popari	**0.831**	0.010	0.005	**0.633**	0.148
BatchBlend	0.815	**0.266**	**0.201**	0.341	**0.439**

Table 3.5: Average biological conservation metrics for each of the benchmark methods on coronal mice slices. Bold values show better performance between BatchBlend and Popari

Method	Isolated labels	kMeans NMI	kMeans ARI	Silhouette label	cLISI
Popari	**0.495**	**0.229**	**0.113**	**0.477**	**0.982**
BatchBlend	0.464	0.109	0.021	0.451	0.726

addition, when looking at the local mixing metrics (iLISI and kBET), BatchBlend shows that it can create neighborhoods where cells from different batches are well-mixed which we observed in **Fig.** 3.8. Aggregating all of the batch correction metrics, seen in **Fig.** 3.9, shows that BatchBlend performing better than Popari is consistent across all random seed runs of the methods. However, an observation is that this batch correction score for BatchBlend seemed to have a relatively greater standard deviation than Popari which perhaps indicates that it is possible for a single run of BatchBlend, the performance could be worse than the average outcome. However, for batch correction, BatchBlend's performance is still better than Popari.

Similarly, when quantifying biological conservation, Popari excels in preserving the biological signal across all metrics. Clustering-based metrics (kMeans NMI and ARI) indicate that Popari maintains a cell-type structure that aligns well with cell type labels. The separation metrics (silhouette label and cLISI) suggest that Popari keeps distinct cell types well separated while ensuring that cells of the same type remain grouped together (isolated labels). This meant that the total biological conservation score for Popari was consistently greater than BatchBlend across all random seed runs of the models as seen in **Fig.** 3.9.

Interestingly, although BatchBlend does not have competitive performance to Popari in biological conservation, the quantitative metrics difference is not as significant as when we quantified BatchBlend's performance on simulated data. This perhaps indicates that our simulations that are supposed to reflect batch effect found when integrating spatial transcriptomic technologies may be slightly less strong than the

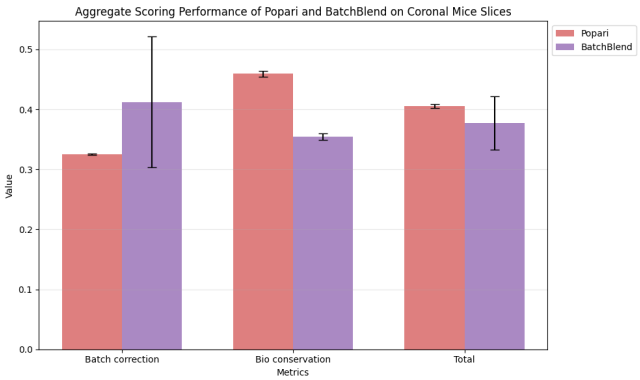


Figure 3.9: Aggregate Batch Correction and Biological Conservation Scores for 3 Random Seed Runs

hyperparameters we estimated. This, perhaps suggests that the gap between BatchBlend and Popari is smaller than previously expected when analyzing the performance with simulated mSRT data. This was seen in the total score shown in **Fig.** 3.9. In some cases, the current formulation of BatchBlend performs better than Popari in integration of data from different technologies. However, on average, due to the biological conservation performance, BatchBlend performs worse than Popari. This stark contrast suggests that BatchBlend's batch effect correction mechanism may be overly aggressive, successfully removing technical variation but also disrupting biological signals in the process. Conversely, Popari's lack of explicit batch correction preserves biological information at the expense of leaving technical artifacts intact. This trade-off highlights a fundamental challenge in batch effect correction: finding the right balance between removing technical noise and maintaining biological signal integrity.

3.7 Future Directions

Although BatchBlend demonstrates competitive performance against established batch effect correction methods, several promising research directions could enhance its capabilities. First, incorporating spatial information into the batch effect correction process represents a critical advancement opportunity. In the current implementation, although the mathematical framework for spatial interaction is derived in the model formulation, BatchBlend does not fully leverage this spatial context during batch correction.

This limitation stems from a technical challenge: while the NMF iterations effectively learn batch effect parameters, subsequent spatial iterations cause the embeddings to collapse to zero values (as detailed in **Section** 2.7). Addressing this collapse through modifications to the model formulation would enable BatchBlend to simultaneously correct batch effects while preserving spatial relationships, potentially improving both integration quality and biological signal preservation. A possible solution would be implementing the optimization with a β^t prior as described in **Section** 2.6. Although this has been derived, this has not yet been implemented and is an immediate next step for BatchBlend.

In addition, further validation of BatchBlend on high-quality real-world mSRT datasets represents another next step. Thus far, we have described integration of MERFISH, MERSCOPE and STARmap spatial transcriptomics technologies. However, further technologies can be integrated for coronal mouse brain slices like CosMx, and Xenium. Beyond successful integration of these datasets, further analysis on potential novel spatial and biological insights about the mouse cortex would be valuable next steps. This would show the added value of BatchBlend since prior to this integration, these insights would be inaccessible due to technical variation or batch effect.

Bibliography

[1] Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nature Methods* **15**, 1053–1058 (2018).

[2] Korsunsky, I. *et al.* Fast, sensitive, and accurate integration of single cell data with Harmony. *bioRxiv* (2019). URL `https://doi.org/10.1101/461954`. Preprint.

[3] Polański, K. *et al.* Bbknn: fast batch alignment of single cell transcriptomes. *Bioinformatics* **36**, 964–965 (2020).

[4] Hie, B. L., Kim, S., Rando, T. A., Bryson, B. & Berger, B. Scanorama: integrating large and diverse single-cell transcriptomic datasets. *Nature Protocols* **19**, 2283–2297 (2024).

[5] Behdenna, A. *et al.* Pycombat, a python tool for batch effects correction in high-throughput molecular data using empirical bayes methods. *BMC Bioinformatics* **24**, 459 (2023).

[6] Johnson, W. E., Li, C. & Rabinovic, A. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics* **8**, 118–127 (2007).

[7] Lu, L. & Welch, J. D. Pyliger: scalable single-cell multi-omic data integration in python. *Bioinformatics* **38**, 2946–2948 (2022).

[8] Welch, J. D. *et al.* Single-cell multi-omic integration compares and contrasts features of brain cell identity. *Resource* (2019).

[9] Kriebel, A. R. & Welch, J. D. Uinmf performs mosaic integration of single-cell multi-omic datasets using nonnegative matrix factorization. *Nature Communications* **13**, 780 (2022).

[10] Hao, Y. *et al.* Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nature Biotechnology* **42**, 293–304 (2024).

[11] Fang, S. *et al.* Computational approaches and challenges in spatial transcriptomics. *Genomics, Proteomics & Bioinformatics* **21**, 24–47 (2022).

[12] Li, S., Shen, Q. & Zhang, S. Spatial transcriptomics-aided localization for single-cell transcriptomics with STALocator. *Cell Systems* **16**, 101195 (2025).

[13] Liu, W. *et al.* Probabilistic embedding, clustering, and alignment for integrating spatial transcriptomics data with precast. *Nature Communications* **14**, 296 (2023).

[14] Zhao, J. *et al.* Inspire: interpretable, flexible and spatially-aware integration of multiple spatial transcriptomics datasets from diverse sources. *bioRxiv* (2024).

[15] Zhang, C. *et al.* spatialign: an unsupervised contrastive learning model for data integration of spatially resolved transcriptomics. *GigaScience* **13**, giae042 (2024).

[16] Zhou, X., Dong, K. & Zhang, S. Integrating spatial transcriptomics data across different conditions, technologies and developmental stages. *Nature Computational Science* **3**, 894–906 (2023). URL `https://www.nature.com/articles/s43588-023-00469-4`.

[17] Luecken, M. D. *et al.* Benchmarking atlas-level data integration in single-cell genomics. *Nature Methods* **19**, 41–50 (2022).