Federated MaxFuse

Diagonal Integration of Weakly Linked Spatial and Single-cell Data through Federated Learning

MSc Computer Science Thesis

Krzysztof Baran

Delft University of Technology

TUDelft

Federated MaxFuse

Diagonal Integration of Weakly Linked Spatial and Single-cell Data through Federated Learning

by

Krzysztof Baran

Student number
4662148

Thesis advisor:	Prof. dr. ir. Marcel Reinders
External advisor:	Dr. Jérémie Decouchant (Distributed Systems)
Daily supervisor:	MSc Swier Garst
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science
Research Group:	Pattern Recognition & Bioinformatics
Defense Date:	May 9, 2025

Cover Image:	Generated using DALL·E 3 by OpenAI
Cover:	TU Delft Report Style, with modifications by Daan Zwaneveld

Federated MaxFuse: Diagonal Integration of Weakly Linked Spatial and Single-cell Data through Federated Learning

Krzysztof Baran
Delft University of Technology
Delft, The Netherlands

Swier Garst
Delft University of Technology
Delft, The Netherlands

Marcel Reinders
Delft University of Technology Delft, The Netherlands

Jérémie Decouchant
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Integrating single-cell multi-omic data is crucial for comprehensive biological discovery, yet it remains challenging due to the weak correlation between modalities, data heterogeneity, and stringent privacy regulations. Conventional integration methods that depend on shared features or matched cells, which are rarely available in practice. While some diagonal integration approaches might mitigate some of these limitations, they are sensitive to noise, prone to overfitting, and challenging to validate, especially in the absence of centralized data access. This thesis introduces Federated **Ma**tching **x**cross modalities via **Fu**zzy **s**moothed **e**mbeddings (MaxFuse), a novel adaptation of MaxFuse within a Federated Learning (FL) framework, which enables privacy-preserving diagonal integration through fuzzy smoothing, federated Canonical Correlation Analysis (CCA), and iterative matching without exchanging raw data. We validate Federated MaxFuse on benchmark single-cell datasets, demonstrating that it achieves matching accuracy and embedding quality comparable to centralized baselines across supervised and unsupervised metrics. These findings establish Federated MaxFuse as a practical and scalable solution for privacy-preserving integration of multi-omic data, enabling robust cross-institutional analyses under real-world constraints.

Krzysztof Baran, Swier Garst, Marcel Reinders, and Jérémie Decouchant. 2025. Federated MaxFuse: Diagonal Integration of Weakly Linked Spatial and Single-cell Data through Federated Learning. 36 pages.

1 INTRODUCTION

Integrating single-cell data is a methodological approach to combine high-throughput sequencing datasets into a cohesive, harmonized dataset, originating from different batches, donors, or conditions. This integration enables comprehensive analysis while preserving authentic biological diversity, facilitating subsequent studies, and enhancing the precision of comparative investigations across diverse datasets [64, 3]. The ultimate goal of the integration is to align shared cell types and states across datasets, eliminate unwanted technical discrepancies among experiments, and enhance statistical power [47], making it a complex endeavor. It is a critical step in extracting meaningful insights from the wealth of heterogeneous single-cell data generated, enabling a more comprehensive understanding of complex biological systems [21, 9, 3, 2].

Furthermore, having biasless data integration means that one could perform downstream tasks such as differential expression, biomarker detection, or trajectory inference, which can lead to new biological discoveries about cellular function, development, and disease mechanisms [47, 2]. Additionally, more data is being produced with the recent rapid advancement of single-cell technologies. Nonetheless, the datasets are at best weakly linked due to differences in settings and protocols, highlighting the urgent need for new computational approaches to integrate these datasets [29].

Although new technological innovations in single-cell genomics assays help quantify and harness biological data, the challenge remains. There are many heterogeneous data modalities, ranging from CO-Detection by indEXing data (CODEX) and single-cell Assay for Transposase-Accessible Chromatin sequencing (scATAC-seq) to single-cell RNA sequencing (scRNA-seq), each presenting challenges such as high sparsity, high noise, high dimensionality, missing information for some properties, and low quantity. Therefore, weakly linked data requires new protocols.

Furthermore, they all need different preprocessing and normalization streams [3] [77]. Batch effects, sequencing depth, and data sparsity have a significant impact on the performance of analyses such as differential expression analysis [61]. These data quality challenges can often impair the performance of traditional machine learning methods, such as Deep Neural Network (DNN)s and Support Vector Machines (SVM)s, which are highly sensitive to noise, sparsity, and batch effects [3, 14, 58, 69]. Moreover, even when such issues are mitigated, these models often struggle with the inherent characteristics of biological datasets, namely high dimensionality and limited sample size, resulting in risks of overfitting or underfitting [14]. On the other hand, popular statistical dimensionality reduction methods and regularization methods, such as Least absolute shrinkage and selection operator (L1), Ridge regression (L2), or Principal Component Analysis (PCA), are effective for data analysis or downstream prediction tasks [17, 20, 69]. However, they do not handle missing information, assume linearity, and require some prior biological information, potentially limiting their ability to capture complex relationships in the data, which is prevalent in -omics applications [18, 33, 3, 58]. As a result, these methods will not effectively capture the coupling between different data modalities [3].

The integration methods can be divided into three main approaches: horizontal, vertical, and diagonal. Horizontal integration methods utilize genomic features with a mapping function, which is used to align different datasets despite the absence of matched cells in experiments [85, 89, 4]. It proves beneficial when experiments lack shared cells but possess overlapping features that can serve as anchors [84, 34]. Examples of such integration are Harmony [44], Linked Inference of Genomic Experimental Relationships (LIGER) [80], and Seurat v3 [73]. Likewise, vertical integration employs shared cells among various data modalities as integration anchors, allowing for interesting relationships between different molecular layers within the same cells locally (associations between specific features) or globally (identifying broader cellular states)[85, 89, 4, 84]. Examples of such integration are Canonical Correlation Analysis (CCA)[78], and integrative Non-negative Matrix Factorization (iNMF) [87].

Nevertheless, while these methods exhibit commendable efficacy in the integration of well-aligned datasets, mainly when prior knowledge regarding data linkage is available, they suffer from over-correction, highly dimensional data (possibly zero-inflated) getting severely distorted [66], batch effects, and forcibly merging non-matching sub-populations of cells [3, 85, 53]. Technological constraints exacerbate this problem, as existing single-cell multi-modal technologies are unable to efficiently gather data from multiple molecular layers of individual cells at scale [3]. This means that having shared samples (as in horizontal integration) or shared features (as in vertical integration) is not always feasible, necessitating integration strategies beyond these traditional paradigms. Due to these problems with horizontal and vertical integration, we are interested in diagonal integration.

Diagonal integration, alternatively termed cross-modal integration [3, 85], distinguishes itself from the techniques above by operating independently of anchors, shared features, or cellular structures. Instead, diagonal integration reconstructs a low-dimensional manifold that captures covariation patterns across diverse modalities [4], particularly when integrating data lacking explicit feature alignment. However, this method is hindered by challenges such as intrinsic heterogeneity of modalities, high dimensionality, noise, and sparsity. Conventional integration methods often struggle to address these complexities, and existing integration approaches encounter limitations, particularly when dealing with data that lacks precise feature matching. This method is inherently more complex, as it requires the direct learning of inter-modal relationships without the aid of anchors [85]. Consequently, innovative computational strategies are necessary, although existing methods have demonstrated limited success in addressing these complexities effectively [3, 85].

Additionally, the absence of cell or feature matching makes proper validation or interpretation of diagonal integration methods less straightforward compared to horizontal or vertical methods [12, 38]. Even partial correlations among modalities do not guarantee successful alignment of their latent manifolds [3, 38, 85]. Approaches like **M**anifold **A**lignment to **CH**aracterize **E**xperimental **R**elationships (MATCHER) [79], which employs a Gaussian process latent variable model, rely on strong biological assumptions about the data [4]. Several attempts have been made to integrate single-cell data from different modalities in recent years. One such approach is embedding spaces to map meaning and influences between modalities [35]. Although this approach is promising, it is not as feasible due to almost infinite and unseen sequence possibilities that naturally change over time, akin to those encountered in natural language processing [12]. Similarly, methods relying on 'weakly linked' features (those with limited known mappings) often result in significant mismatches or leave large portions of data unmatched [12, 54]. Addressing these shortcomings, the recently proposed **Ma**tching **x**cross modalities via **Fu**zzy **s**moothed **e**mbeddings (MaxFuse) method leverages techniques such as static data removal, iterative co-embedding, 'fuzzy smoothing' with NN graphs, de-noising through internal dimensional reductions through Singular Value Decomposition (SVD) (leveraging Arnoldi/Lanczos methods for sparse matrices [68]), and cell matching to leverage all information within each modality through iterative refinement via CCA. This ensures high-quality integration with little to no batch effects [12]. It was shown to surpass numerous contemporary methodologies in accuracy and scalability, thereby establishing it as a significant approach for integrating single-cell data across diverse modalities and offering potential insights into the intricate biological mechanisms associated with various diseases [12, 3].

While these techniques may work well, they all assume that the data is available in one lab and can be processed using a single method. This is a problem as datasets tend to be across silos which can be cross-institutional and cannot be shared. A solution to this is to establish an agreement between laboratories and data owners, allowing them to share the data, but this introduces substantial regulatory and logistical burdens. [62], and simply removing Personally Identifiable Information (PII) (a process known as pseudo-deanonymization) is ineffective due to a high risk of re-identification back to the data owner [43, 63]. On top of that, regulatory constraints such as GDPR or HIPAA restrict the sharing and centralization of sensitive data even when pseudonymized [43], and modern computation setups like Multi-Party Computation (MPC) are inefficient for large-scale systems with many participants and require special hardware [16], preventing traditional integration methods from being feasible. As a result, institutions often face barriers to collaboration. Federated Learning (FL) is a distributed model training framework involving multiple participants collaborating to train localized model versions. These participants train localized versions of the model, share model parameters (weights), aggregate them, and then return the aggregated parameters to every network participant [86]. FL's distributed nature allows for direct training models on federated networks, enabling the development of federated formats for tasks such as model pre-training, life-long learning, and multi-modal learning without having to share the data [1, 10, 56, 83, 88]. Hence, the Federated Learning (FL) can mitigate most privacy concerns associated with fully centralized learning [83].

Additionally, it is also vital to note that FL introduces novel challenges in the communication of 'data representation' that require a mental shift from traditional approaches and necessitate the utilization of the capabilities of connected network nodes [50]. Despite these hurdles, FL has successfully enabled collaborative, decentralized training of deep neural networks while preserving privacy by avoiding the sharing of raw data among network participants [51]. Given its success, a natural question arises: can similar principles be applied to perform diagonal data integration in a federated setting, thereby enabling privacy-preserving integration across distributed datasets. However, it still presents challenges in providing accurate integration, minimizing communication overhead, and ensuring that minimal information about the data is shared. This is especially important, as it is not as straightforward and standardized as sharing neural network weights across networks, typically through the network itself.

This study focuses on diagonal data integration using the single-cell MaxFuse algorithm, which we have converted into a federated method to address privacy concerns associated with data sharing. This approach aims to integrate single-cell data without sharing the actual data, while achieving the same results as the centralized MaxFuse algorithm.

2 METHODOLOGY

We developed a federated version of the diagonal integration technique. Specifically, we adapted the MaxFuse method to operate in a federated environment, which enables the integration of two different data modalities without requiring centralized data collection.

2.1 Data

We utilized publicly available datasets that were provided by the authors of the MaxFuse algorithm [12], sourced from the Wharton Research Data Services repository [70]. They comprise between 10,000 and 187,000 observations and range from 57 to 20,729 features.

We selected two modality-specific datasets from this collection for in-depth analysis.. The first consists of strongly matched human single-cell RNA and protein modalities [26] serving as a benchmark with known ground truth to assess the accuracy of the MaxFuse algorithm in modality matching which is a challenge faced by methods such as LIGER, Harmony, and Seurat V3 [73]. The second dataset comprises unmatched modalities between human intestine single-cell datasets, incorporating CODEX image data from Kennedy et al. [41] and scRNA-seq data from tonsil dissociated cells in King et al. [42]. This dataset enables evaluation of the robustness of integrating 'weakly' feature-linked modalities in final data embeddings to perform downstream analyses [30, 76].

For modalities with high signal-to-noise ratio, e.g., CITE-seq and scRNA-seq, we are constructing meta-cells, which are mean representations of 'communities' built by the Leiden clustering algorithm [75].

The labels overview is presented in Table 1, and the overview of preprocessing parameters used in this work is provided in Table 2.

Dataset	Data Type	Source	Label Counts
Antibodies	CITE-seq	[26]	*Levels (1-3)*: 8,31,58
	PBMC	[26]	
Tonsils	scRNA-seq	[42]	*Cluster labels*:6
	CODEX	[42]	

Table 1: Overview of datasets' labels.

2.1.1 Matching Benchmark Dataset: CITE-seq & PBMC Antibodies. The first dataset is a CITE-seq dataset that comprises human Peripheral Blood Mononuclear Cell (PBMC) cells with measurements from 228 antibodies, which enables a comprehensive analysis of both RNA and protein modalities [26]. The data were pre-matched through weighted-NN, which successfully captured extensive lymphoid diversity and helped map correspondences in immune responses [26]. The dataset from MaxFuse contains a subset of the original data of 10,000 cells, with a sparse RNA dataset of 1,707 features and a non-sparse Protein dataset of 224 features.

2.1.2 Integration Quality Dataset: CODEX & scRNA-seq Tonsils. The second dataset combines data from two separate studies. The first data modality is preprocessed CO-Detection by indEXing data (CODEX) Tonsil Images [1] from five human lymphoid tissues: three tonsils, one spleen, and one lymph node. These tissues are crucial sites for immune responses, containing B-cell follicles that play a key role in antibody production and the immune function [25, 41]. The second data modality is a scRNA-seq Human Tonsil Dissociated Cells, where the data was obtained from patients aged 3 to 7 years undergoing routine tonsillectomy, collected and analyzed by King et al. [42]. The dataset provided by MaxFuse contains a subset of the original data of 12,977 RNA points (i.e., scRNA-seq Tonsil Cells) with 5,000 features and 178,919 protein points (i.e., CODEX Tonsil Images) with 46 features. The authors have already pre-matched the labels as Cluster Info/Terms.

2.1.3 Preprocessing Pipeline. Both datasets were initially preprocessed by the authors of the dataset, facilitating tasks such as analysis and visualization, as well as by the authors of MaxFuse for further usability and reproducibility of the results. However, additional steps were required before it could be used, as each modality requires two input dataset types: an 'active dataset', i.e., a complete, modality-specific, preprocessed dataset, and a 'shared dataset', i.e., feature-aligned subsets across modalities. Each dataset has a dictionary-like correspondence. For instance, a protein marker name exists for an RNA gene name.

The general preprocessing pipeline for both datasets includes several common steps. Initially, the pipeline involves creating 'shared feature sets' by aligning both modalities based on the pre-established biological feature correspondences, which can be found in Appendix D. Subsequently, after creating 'shared feature sets', it undergoes variability feature filtering based on the standard deviation across all cells, retaining only features that surpass modality-specific thresholds specified in Table 2. After feature selection, both datasets have their cells normalized by target total, such that each cell is scaled to have the median total count of its respective dataset before normalization. A subsequent $\log(X + 1)$ transformation stabilizes variance and reduces skewness. Additionally, for specific 'active datasets' modalities like the scRNA-seq and CITE-seq, RNA-specific features are further filtered by identifying Highly Variable Genes (HVGs) using the Seurat V3 method [65, 73, 90], selecting genes with the highest normalized dispersion within expression-level bins. The number of top HVGs can be defined by the user, as defined in Table 2. Lastly, both datasets are scaled to zero mean and unit variance across features. Variability filtering is then applied to the 'active datasets' based on their standard deviation at the end.

The resulting preprocessed datasets ('shared' and 'active') then serve as inputs to the MaxFuse algorithm for generating embeddings and downstream inter-modality matching.

[1]**CODEX software**: https://github.com/nolanlab/CODEX

Dataset	Data Type	Top HVGs Count	Filtering Threshold		Initial Shape		Preprocessed Shape	
			Active	*Shared*	*Active*	*Shared*	*Active*	*Shared*
Antibodies	CITE-seq	-	1^{-5}	1^{-5}	$10{,}000 \times 20{,}729$	$10{,}000 \times 180$	$10{,}000 \times 1{,}707$	$10{,}000 \times 177$
	PBMC	n/a	1^{-5}	1^{-5}	$10{,}000 \times 224$	$10{,}000 \times 180$	$10{,}000 \times 224$	$10{,}000 \times 177$
Tonsils	scRNA-seq	5,000	1^{-5}	0.5	$12{,}977 \times 33{,}538$	$12{,}977 \times 53$	$12{,}977 \times 5{,}000$	$12{,}977 \times 32$
	CODEX	n/a	1^{-5}	0.1	$178{,}919 \times 57$	$178{,}919 \times 53$	$178{,}919 \times 46$	$178{,}919 \times 32$

Table 2: Overview of datasets preprocessing details, including data types, preprocessing parameters, initial and resulting data shapes. Each dataset is presented with its associated data type, the number of Highly Variable Genes (HVGs) selected (where applicable), filtering thresholds during preprocessing for active and shared modalities, and dataset shapes before and after preprocessing (samples $\times$ features).

2.2 Algorithm & Federation

The MaxFuse algorithm is systematically structured into six distinct phases as illustrated in Algorithm 1. It begins with an initial preparation step that sets the groundwork for the subsequent five stages.

Firstly, the shared feature datasets $\mathbf{Y}^\circ$ and $\mathbf{Z}^\circ$, obtained through preprocessing (see subsubsection 2.1.3), are constructed using known biological mappings described in Appendix D. This preprocessing procedure is denoted as `GetSharedFeatures` for their respective dataset.

Then, in the second step, we optionally perform meta-cell cluster aggregation using the Leiden clustering algorithm [75] for a dataset exhibiting a high signal-to-noise ratio, producing datasets $\mathbf{Y}^m$ and $\mathbf{Z}^m$. The idea is to create 'fake' cells representing community clusters of cells of sizes n_y and n_z, which is optional to perform for each modality. The aggregated modality dataset, which we call meta-modality, clusters are averaged out to get a center of all clusters. The original data is retained unchanged for the non-aggregated modality dataset, which we refer to as non-meta-modality.

$$\rho(X,Y) = \frac{\sum_{k=1}^{p}(X_k - \bar{X})(Y_k - \bar{Y})}{\sqrt{\sum_{k=1}^{p}(X_k - \bar{X})^2}\sqrt{\sum_{k=1}^{p}(Y_k - \bar{Y})^2}}, \qquad (1)$$

where: $\bar{X}$ and $\bar{Y}$ are means for p features

The third step is fuzzy smoothing, for which we first construct k-NN graphs G_Y and G_Z for each modality. Fuzzy smoothing [12, 39, 40] (see Algorithm 2 for the implementation) uses these graphs to mitigate uncertainty and noise in biological data by adjusting each cell towards the average of its neighborhood. This smoothing allows each cell's features to adopt some of the characteristics of its neighbors, reflecting real-world ambiguity. This denoising step reinforces biological structure and improves generalizability by filtering out irregularities and aligning with local patterns. Furthermore, this prevents analyses from being overly influenced by outliers while accommodating heterogeneity in the dataset [39, 40]. Consequently, fuzzy smoothing improves the quality and reliability of the initial matching Π°, corrects noisy or erroneous structures originating from the CCA method during creation of iteratively refined matching Π.

The fourth step involves computing an initial matching between the smoothed 'shared' features, denoted as $\widetilde{\mathbf{Y}^\circ}$ and $\widetilde{\mathbf{Z}^\circ}$, obtained through the previously mentioned "fuzzy smoothing" method [39, 40] (see Algorithm 2). Using these smoothed datasets, we calculate Pearson correlation $\rho(\widetilde{\mathbf{Y}^\circ}, \widetilde{\mathbf{Z}^\circ})$, as illustrated in Equation 1. We then convert this correlation matrix into a distance matrix $\mathbf{D}^\circ$ by subtracting the correlation values from a matrix of ones. This inversion step transforms the matching problem from a correlation maximization task into a distance minimization task, known as the linear sum assignment problem [7], defined in Equation 2. To solve this minimization efficiently, we employ the Jonker-Volgenant algorithm [15]. The objective is to find the $\mathtt{min}(n_\mathrm{Y}, n_\mathrm{Z})$ best-matching pairs of cells. This results in a binary assignment matrix M, where each entry indicates whether the i^{th} element from $\widetilde{\mathbf{Y}^\circ}$ is matched to the j^{th} element from $\widetilde{\mathbf{Z}^\circ}$. From this matrix M, we extract pairs of matched indices $\{(i_k, j_k)\}_{k=1}^{n} \subset \mathbb{Z}^+ \times \mathbb{Z}^+$ along with corresponding weights $\{(w_k)\}_{k=1}^{n} \in \mathbb{R}$, , forming the initial match $\Pi^\circ = \{(i_k, j_k, w_k)\}_{k=1}^{n}$.

$$\begin{aligned} \min_{M \in \{0,1\}^{n_\mathrm{Y} \times n_\mathrm{Z}}} \quad & \sum_{i,j} M_{ij}\, \mathbf{D}_{ij} \\ \text{such that} \quad & \sum_{i} M_{ij} \le 1, \quad \forall j, \\ & \sum_{j} M_{ij} \le 1, \quad \forall i, \\ & \sum_{i,j} M_{ij} = \mathtt{min}(n_\mathrm{Y}, n_\mathrm{Z}) \end{aligned} \qquad (2)$$

Subsequently, in the fifth step, we enter an iterative refinement phase. We alternate between integrating the sample and feature spaces, iteratively using the improved sample space integration to get a better feature embedding, and vice versa. The output of the initial matching step Π° is set to Π used to align the cells from the data modalities $\mathbf{Y}^{cc}$ and $\mathbf{Z}^{cc}$ which are initialized as fuzzily smoothed meta and non-meta modalities $\mathbf{Y}^m$ and $\mathbf{Z}^m$ respectively. After this, a refinement loop begins where the algorithm aligns the datasets with Π to produce aligned datasets of equal number of rows Y^{aligned} and Z^{aligned} through indexing the matching with $\Pi_0 = \{(i_k)_{k=1}^{n}\}$ and $\Pi_1 = \{(j_k)_{k=1}^{n}\}$. Then it iteratively learns a linear joint embedding $\mathbf{Y}^{cc}$ and $\mathbf{Z}^{cc}$ of cells across modalities. This is done by computing a Canonical Correlation Analysis (CCA) based on all features of the cross-modal matched cell pairs $\mathbf{Y}^{\text{aligned}}$ and $\mathbf{Z}^{\text{aligned}}$. Finally, we apply fuzzy smoothing (Algorithm 2) to produce $\widetilde{\mathbf{Y}^{cc}}$ and $\widetilde{\mathbf{Z}^{cc}}$, and again applies Jonker–Volgenant algorithm (to solve for linear sum assignment) on the updated distance matrix $\mathbf{D}^{cc}$ and using Equation 2 to create a new iteration of refined matching Π. The refinement loop continues for a predetermined number of iterations, T, yielding a final, optimized matching Π.

Finally, step six performs percentile-based filtering and propagation of matches. The refined matching Π, produced through iterative refinement, is filtered using a threshold α to remove low-quality matches. This results in Π^{pivot}, allowing for a more accurate representation of the relationships between the datasets. From these filtered pivots, CCA is performed using the filtered pivots aligned datasets to produce the embedding weights $\mathbf{Y}^{m,e}$ and $\mathbf{Z}^{m,e}$, which are used in propagation and for getting the final full data embeddings. The propagation step produces Π^{prop}, where for any unmatched cell in either modality, we identify nearest cells NN within the same modality and assign the same link to the opposite modality cell for the unmatched cell as the neighboring cell. After that, we calculate the Pearson correlation (using Equation 1) of the propagation-aligned datasets and multiply these correlations with each other to get the new weights for the propagated matching Π^{prop}. Then the redundant matches that match to the same cells are removed to create $\Pi^{\text{prop}*}$ and filtered again to create Π^{final} for weight β. The final output includes a list of matched pairs matching with their weights Π^{final} and joint embeddings $\mathbf{Y}_e, \mathbf{Z}_e$ of all cells in both modalities that can be utilized for downstream analysis.

Algorithm 1 Centralised MaxFuse Algorithm

```
Input:
  • Two datasets $\mathbf{Y} \in \mathbb{R}^{N_y \times p_y}$, $\mathbf{Z} \in \mathbb{R}^{N_z \times p_z}$
  • Transformation functions $\texttt{GetSharedFeatures}_y : \mathbb{R}^{N_y \times p_y} \to \mathbb{R}^{N_y \times s}$ and $\texttt{GetSharedFeatures}_z : \mathbb{R}^{N_z \times p_z} \to \mathbb{R}^{N_z \times s}$ ($s$ being number of shared components)
  • Meta cell counts $n_y, n_z \in \mathbb{Z}^+ \cup \varnothing$ (if $\varnothing$, do not create meta-cells)
  • NN count $k \in \mathbb{Z}^+$
  • Smoothing interpolation weight $w \in (0.0, 1.0)$
  • Pivot and propagation filtering weight $\alpha, \beta \in (0.0, 1.0)$
Output:
  • Final embeddings $\mathbf{Y}_e \in \mathbb{R}^{N_y \times s}$ and $\mathbf{Z}_e \in \mathbb{R}^{N_z \times s}$
  • Final matching $\Pi^{\text{final}} = \{(i_k,\ j_k,\ w_k)\}_{k=1}^{n} \subset \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{R}$ where $n \leq min(N_y, N_z)$

  // Step 1: Transform to Shared Features
1:  $\mathbf{Y}^{\circ} \leftarrow \texttt{GetSharedFeatures}_y(\mathbf{Y})$
2:  $\mathbf{Z}^{\circ} \leftarrow \texttt{GetSharedFeatures}_z(\mathbf{Z})$
  // Step 2: Construct Meta-cells
3:  $\mathbf{Y}^{m} \leftarrow \texttt{ConstructMetaCells}(\mathbf{Y}, n_y)$
4:  $\mathbf{Z}^{m} \leftarrow \texttt{ConstructMetaCells}(\mathbf{Z}, n_z)$
  // Step 3: Create Nearest-Neighbor Graphs
5:  $\mathbf{G}_Y \leftarrow \texttt{NearestNeighborGraph}(\mathbf{Y}^{m}, k)$
6:  $\mathbf{G}_Z \leftarrow \texttt{NearestNeighborGraph}(\mathbf{Z}^{m}, k)$
  // Step 4: Create Initial Shared Data Matching $\Pi^{\circ}$
7:  $\widetilde{\mathbf{Y}^{\circ}} \leftarrow \texttt{FuzzySmoothing}(\mathbf{Y}^{\circ}, \mathbf{G}_Y, w)$
8:  $\widetilde{\mathbf{Z}^{\circ}} \leftarrow \texttt{FuzzySmoothing}(\mathbf{Z}^{\circ}, \mathbf{G}_Z, w)$
9:  $\mathbf{D}^{\circ} \leftarrow 1.0 - \rho(\widetilde{\mathbf{Y}^{\circ}},\ \widetilde{\mathbf{Z}^{\circ}})$ // Inverse of Pearson
10: $\{(i_k, j_k)\}_{k=1}^{n} \leftarrow \texttt{LinearSumAssignment}(\mathbf{D})$ // Equation 2
11: $\{w_k\}_{k=1}^{n} \leftarrow \mathbf{D}_{\{(i_k, j_k)_{k=1}^{n}\}}$ // Get matching scores
12: $\Pi^{\circ} \leftarrow \{(i_k,\ j_k,\ w_k)_{k=1}^{n}\}$
  // Step 5: Joint Embedding and Iterative Refinement
13: $\Pi \leftarrow \Pi^{\circ}$
14: $\mathbf{Y}^{cc} \leftarrow \texttt{FuzzySmoothing}(\mathbf{Y}^{m}, \mathbf{G}_Y, w)$
15: $\mathbf{Z}^{cc} \leftarrow \texttt{FuzzySmoothing}(\mathbf{Z}^{m}, \mathbf{G}_Z, w)$
16: for $t = 1$ to $T$ do
17:     $\mathbf{Y}^{\text{aligned}}, \mathbf{Z}^{\text{aligned}} \leftarrow \mathbf{Y}^{cc}(\Pi_0), \mathbf{Z}^{cc}(\Pi_1)$
18:     $\mathbf{Y}^{cc}, \mathbf{Z}^{cc} \leftarrow \texttt{CCA}(\mathbf{Y}^{\text{aligned}}, \mathbf{Z}^{\text{aligned}})$
19:     $\widetilde{\mathbf{Y}^{cc}} \leftarrow \texttt{FuzzySmoothing}(\mathbf{Y}^{cc}, \mathbf{G}_Y, w)$
20:     $\widetilde{\mathbf{Z}^{cc}} \leftarrow \texttt{FuzzySmoothing}(\mathbf{Z}^{cc}, \mathbf{G}_Z, w)$
21:     $\mathbf{D}^{cc} \leftarrow 1.0 - \rho(\widetilde{\mathbf{Y}^{cc}},\ \widetilde{\mathbf{Z}^{cc}})$
22:     $\{(i_k^{cc}, j_k^{cc})\}_{k=1}^{n} \leftarrow \texttt{LinearSumAssignment}(\mathbf{D}^{cc})$
23:     $\{w_k^{cc}\}_{k=1}^{n} \leftarrow \mathbf{D}^{cc}_{\{(i_k^{cc}, j_k^{cc})_{k=1}^{n}\}}$ // Get matching scores
24:     $\Pi \leftarrow \{(i_k^{cc},\ j_k^{cc},\ w_k^{cc})_{k=1}^{n}\}$
25: end for
  // Step 6: Filtering and Final Embedding
26: $\Pi^{\text{pivot}} \leftarrow \texttt{FilterOutBadMatches}(\Pi, \alpha)$
27: $\mathbf{Y}^{m,e}, \mathbf{Z}^{m,e} \leftarrow \texttt{FitEmbeddingsCCA}(\mathbf{Y}^{m}(\Pi_0^{\text{pivot}}),\ \mathbf{Z}^{m}(\Pi_1^{\text{pivot}}))$
28: $\Pi^{\text{prop}} \leftarrow \texttt{Propagate}(\Pi^{\text{pivot}}, \mathbf{Y}^{m}, \mathbf{Z}^{m}, \mathbf{Y}^{m,e}, \mathbf{Z}^{m,e})$
29: $\Pi^{\text{prop}*} \leftarrow \texttt{RemoveRedundantConnections}(\Pi^{\text{prop}})$
30: $\Pi^{\text{final}} \leftarrow \texttt{FilterOutBadMatches}(\Pi^{\text{prop}*}, \beta)$
31: $\mathbf{Y}_e, \mathbf{Z}_e \leftarrow \texttt{GetEmbeddingsCCA}(\mathbf{Y}, \mathbf{Z}, \mathbf{Y}^{m,e}, \mathbf{Z}^{m,e})$
32: return $\mathbf{Y}_e,\ \mathbf{Z}_e,\ \Pi^{\text{final}}$
```

Lastly, an important step not explicitly reflected in the pseudocode is the optional application of truncated SVD, particularly via Arnoldi/Lanczos methods optimized for sparse matrices [68]. This step can be applied before computationally intensive procedures or when denoising is desired. Reducing the dataset's dimensionality significantly lowers computational costs and mitigates noise, which is especially critical for large datasets where excessive noise can obscure generalizable covariance patterns. Further details are provided in Appendix B.

Algorithm 2 Fuzzy Smoothing

```
Input:
  • Data matrix $X \in \mathbb{R}^{n \times d}$
  • NN graph edges $G = (e_1, e_2, w) \subset \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{R}$ with incidences pair and distance
  • Smoothing weight $\omega \in [0.0,\ 1.0]$
Output: Smoothed data matrix $\widetilde{X} \in \mathbb{R}^{n \times d}$

1:  Initialize adjacency matrix $A \in \mathbb{R}^{n \times n}$ with 0s
2:  for $i = 1$ to length$(W)$ do
      // Accumulate edge weights
3:      $A_{e_1[i], e_2[i]} \leftarrow A_{e_1[i], e_2[i]} + w[i]$
4:  end for
5:  $D \leftarrow \sum_{j=1}^{n} A_{:,j}$ // Row sums for normalization
6:  $C \leftarrow A \cdot X$ // Weighted sum of neighbors

7:  for $i = 1$ to $n$ do
8:      $C_{i,:} \leftarrow \frac{C_{i,:}}{D_i}$ // Get neighborhood average
9:  end for

10: return $\omega \cdot X + (1 - \omega) \cdot C$ // Interpolation
```

2.2.1 Proposed Federated version of MaxFuse. The federated version of the algorithm exhibits certain resemblances to its centralized counterpart. Nonetheless, it is crucial to recognize the differences in their operational mechanisms. The premise is that the data is distributed across organizations that are prohibited from sharing it [72] (see Figure 1). Here, the variable i $\in$ 1, 2 denotes the node index (node 1 or node 2). As for the matching, if we denote centralized matching as $\Pi = \{(i_k,\ j_k,\ w_k)_{k=1}^n\} \subset \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{R}$, we divide this into a matching belonging to node one is denoted as $\Pi_{(0,2)} = \{(i_k,\ w_k)\}_{k=1}^n \subset \mathbb{Z}^+ \times \mathbb{R}$ and for node two $\Pi_{(1,2)} = \{(j_k\ w_k)\}_{k=1}^n \subset \mathbb{Z}^+ \times \mathbb{R}$. These can be generalized as $\Pi_{(\mathrm{i}-1,2)}$. Each node has its indices and shares the matching weight.

Algorithm 3 Federated MaxFuse Algorithm Initialization

```
For i = (1, 2) which denotes a node index:
Input:
  • Dataset $\mathbf{X}_\mathrm{i} \in \mathbb{R}^{N_{x_\mathrm{i}} \times p_{x_\mathrm{i}}}$
  • Transformation functions GetSharedFeatures : $\mathbb{R}^{N_{x_\mathrm{i}} \times p_{x_\mathrm{i}}} \to \mathbb{R}^{N_{x_\mathrm{i}} \times s}$
  • Meta cell count $n_{x_\mathrm{i}} \in \mathbb{Z}^+ \cup \varnothing$ (if $\varnothing$, do not create meta-cells)
  • NN count $k \in \mathbb{Z}^+$,
  • Smoothing interpolation weight $w \in (0.0, 1.0)$
Output:
  • Initial matching $\Pi^{\mathrm{i},\circ} = \{(i_k/j_k,\ w_k)\}_{k=1}^n \subset \mathbb{Z}^+ \times \mathbb{R}$
  • Meta-cell data $\mathbf{X}_\mathrm{i}^m$
  • NN graph $\mathrm{G}_{X_i}$

   // NODE 1&2
   // Step 1: Transform to Shared Features
1: $\mathbf{X}_\mathrm{i}^\circ \leftarrow$ GetSharedFeatures($\mathbf{X}_\mathrm{i}$)
   // Step 2: Construct Meta-cells
2: $\mathbf{X}_\mathrm{i}^m \leftarrow$ ConstructMetaCells($\mathbf{X}_\mathrm{i}, n_{x_\mathrm{i}}$)
   // Step 3: Create Nearest-Neighbor Graphs
3: $\mathrm{G}_{X_i} \leftarrow$ NearestNeighborGraph($\mathbf{X}_\mathrm{i}^m, k$)
   // Step 4: Create Initial Shared Data Matching
4: $\widetilde{\mathbf{X}_\mathrm{i}^\circ} \leftarrow$ FuzzySmoothing($\mathbf{X}_\mathrm{i}^\circ, \mathrm{G}_{X_\mathrm{i}}, w$)
   // Send normalized shared data $\widetilde{\mathbf{X}_\mathrm{i}^\circ}'$
5: Send($\widetilde{\mathbf{X}_\mathrm{i}^\circ}'$, to='SERVER', action='MATCHING')

   // SERVER: Perform matching from Algorithm 4
6: await ServerMatching($\widetilde{\mathbf{X}_1^\circ}', \widetilde{\mathbf{X}_2^\circ}'$)

   // NODE 1: Receive indices and matching score
7: $\Pi^{1,\circ} \leftarrow$ Receive(
      $\{(i_k,\ w_k)\}_{k=1}^n$,
      from='SERVER',
      action='MATCHING'
   )
8: return $\Pi^{1,\circ}$, $\mathbf{X}_1^m$, $\mathrm{G}_{X_1}$

   // NODE 2: Receive indices and matching score
9: $\Pi^{2,\circ} \leftarrow$ Receive(
      $\{(j_k,\ w_k)\}_{k=1}^n$,
      from='SERVER',
      action='MATCHING'
   )
10: return $\Pi^{2,\circ}$, $\mathbf{X}_2^m$, $\mathrm{G}_{X_2}$
```

Firstly, the initial steps of the federated MaxFuse algorithm are executed locally and in parallel on each participating modality node, as shown in Algorithm 3. Each node begins with input data Xi $\in \mathbb{R}^{N_{x_\mathrm{i}} \times p_{x_\mathrm{i}}}$, for which the shared feature representation Xi$^\circ \in \mathbb{R}^{N_{x_\mathrm{i}} \times s}$ has already been prepared during preprocessing as described in subsubsection 2.1.3 (Step 1) . Subsequently, meta-cells are optionally calculated, resulting in the meta-cell representation $\mathbf{X}_\mathrm{i}^m$ (Step 2). Next, it constructs a nearest-neighbor graph $\mathrm{G}_{X_\mathrm{i}}$ based on the meta-cell representation and a chosen neighbor count $k \in \mathbb{Z}^+$ (Step 3). Lastly, the algorithm applies 'fuzzy smoothing' as shown in Algorithm 2, yielding smoothed data $\widetilde{\mathbf{X}_\mathrm{i}^\circ} \in \mathbb{R}^{N_{x_\mathrm{i}} \times s}$. Steps 1-3 are performed locally by each node/participant. Next, the initial matching step, analogous to the centralized setting (see Algorithm 1), is performed on the server. Since each modality node does not have access to the other node's shared data, it first computes the L2-normalized and fuzzy smoothed shared feature matrix $\widetilde{\mathbf{X}_\mathrm{i}^\circ}'$ and transmits it to the server via the Send procedure (Step 4). Each node then receives its local assignment, $\Pi^{\mathrm{i},\circ}$, containing the modality indices and matching weights

$$\begin{aligned}
\rho(X, Y) &= X' \cdot Y' \\
&= \frac{(X - \bar{X})}{\|X - \bar{X}\|_2} \cdot \frac{(Y - \bar{Y})}{\|Y - \bar{Y}\|_2} \\
\textbf{where:} & \\
X' &= \frac{X - \bar{X}}{\|X - \bar{X}\|_2}, \quad Y' = \frac{Y - \bar{Y}}{\|Y - \bar{Y}\|_2}, \\
\bar{X} &= \text{row-wise mean}, \\
\bar{Y} &= \text{row-wise mean}, \\
\|X - \bar{X}\|_2 &= \text{L2 norm}, \\
\rho(X, Y) &= \text{Pearson correlation matrix}
\end{aligned} \tag{3}$$

On the server side (see Algorithm 4), the received normalized datasets $\widetilde{\mathbf{X}_1^\circ}'$ and $\widetilde{\mathbf{X}_2^\circ}'$ are used to compute the correlation matrix following Equation 3. This calculation, mathematically equivalent to $\rho(\widetilde{\mathbf{X}_1^\circ}, \widetilde{\mathbf{X}_2^\circ})$ (see Equation 1), makes explicit the normalization and mean-centering steps that underlie Pearson correlation. Then matching $\Pi = (i_k, j_k, w_k)_{k=1}^n$ is produced from the distance matrix $\mathbf{D}$, analogous to the centralized. The assignments are subsequently redistributed back to the respective nodes.

Algorithm 4 Federated Sever Matching Procedure

```
Input: Normalized Datasets: $\mathbf{X}_1' \in \mathbb{R}^{N_{x_1} \times p_{x_1}}$ and $\mathbf{X}_2' \in \mathbb{R}^{N_{x_2} \times p_{x_2}}$
Output: Matching $\{(i_k,\ w_k)\}_{k=1}^n \subset \mathbb{Z}^+ \times \mathbb{R}$ for Node 1 and $\{(j_k,\ w_k)\}_{k=1}^n \subset \mathbb{Z}^+ \times \mathbb{R}$ for Node 2

1: Receive($\mathbf{X}_1$, from='1', action='MATCHING')
2: Receive($\mathbf{X}_2$, from='2', action='MATCHING')

3: $\mathbf{D} \leftarrow 1.0 - \mathbf{X}_1' \cdot (\mathbf{X}_2')^\top$ // Inverse of Pearson
4: $\{(i_k, j_k)\}_{k=1}^n \leftarrow$ LinearSumAssignment($\mathbf{D}$) // Equation 2
5: $\{w_k\}_{k=1}^n \leftarrow \mathbf{D}_{\{(i_k, j_k)_{k=1}^n\}}$ // Get matching scores
6: Send($\{(i_k,\ w_k)\}_{k=1}^n$, to='1', action='MATCHING')
7: Send($\{(j_k,\ w_k)\}_{k=1}^n$, to='2', action='MATCHING')
8: return // To the higher-level coordinating logic
```

In the next stage, the algorithm prepares for the main refinement loop, which is orchestrated by the server and runs for T iterations, as shown in Algorithm 5. Locally, the initial matching $\Pi_{(\mathrm{i}-1,\ 2)}^\circ$ is assigned to a loop matching $\Pi_{(\mathrm{i}-1,\ 2)}$ and the modality data $\mathbf{X}_\mathrm{i}^m$ is smoothed and assigned to loop embedding X_i^{cc}. Within this loop, the CCA is computed in a federated manner, denoted as `FederatedCCA`, by exchanging only the necessary weights between the nodes and the server. This step is the core of the federated process and requires careful reflection, as it involves sharing model weights. A detailed breakdown of this procedure is provided in subsubsection 2.2.2. After `FederatedCCA` is finished, just like in the initial matching, the received embeddings X_i^{cc} are locally fuzzily smoothed, yielding $\widetilde{X_i^{cc}}$, which is then L2 normalized (denoted as $\widetilde{X_i^{cc}}'$) and sent to the server through `Send` procedure where new matching $\Pi_{(i-1,2)}$ is calculated as shown in Algorithm 4. This matching is used to align the newly calculated X_i^{cc}, and this repeats until the loop terminates.

Algorithm 5 Federated MaxFuse Loop

```
For i = (1, 2) which denotes a node index:
Input:
  • Meta-cell dataset $X_i^m \in \mathbb{R}^{N_{x_i}^m \times p_{x_i}}$
  • Initial Matching $\Pi^\circ_{(i-1,\,2)} \subset \mathbb{Z}^+ \times \mathbb{R}$
  • NN graph for fuzzy smoothing $G_{X_i}$ together with interpolation
    weight $w \in (0.0, 1.0)$
Output: Pivot Matching $\Pi_{(i-1,\,2)} \subset \mathbb{Z}^+ \times \mathbb{R}$

// Step 5: Iterative Refinement
// NODE 1&2
1:  $X_i^{cc} \leftarrow$ FuzzySmoothing($X_i^m, G_{X_i}, w$)
2:  $\Pi_{(i-1,\,2)} \leftarrow \Pi^\circ_{(i-1,\,2)}$

// SERVER: orchestrated loop
3:  for $t = 1$ to $T$ do
      // NODE 1&2
4:    $X_i^{\text{aligned}} \leftarrow X_i^{cc}[\Pi_{i-1}]$

      // SERVER: CCA from Algorithm 7
      // Federated Learning Orchestration
5:    $(X_1^{cc}, X_2^{cc}) \leftarrow$ FederatedCCA($X_1^{\text{aligned}}, X_2^{\text{aligned}}$)

      // NODE 1&2
6:    $\widetilde{X_i^{cc}} \leftarrow$ FuzzySmoothing($X_i^{cc}, G_{X_i}, w$)
      // Send normalized data
7:    Send($\widetilde{X_i^{cc}}'$, to='SERVER', action='MATCHING')

      // SERVER: Perform matching from Algorithm 4
8:    await ServerMatching($\widetilde{X_1^{cc}}', \widetilde{X_2^{cc}}'$))

      // NODE 1&2
9:    Receive($\Pi'_{(i-1,\,2)}$, from='SERVER')
10:   $\Pi_{(i-1,\,2)} \leftarrow \Pi'_{(i-1,\,2)}$
11: end for
    // NODE 1&2 : Return pivot matching
12: return $\Pi_{(i-1,\,2)}$
```

Lastly, in step 6 in Algorithm 6, the refined matching $\Pi_{(i-1,\,2)}$ undergoes an additional filtering process where the fraction α of the weakest matches are discarded using a `FilterOutBadMatches` procedure by looking at their weights in Π_2, which are the same across nodes. CCA is recalculated on this filtered subset through using `FitEmbeddingCCA`, and the resulting final weights can be used to generate updated modality-specific embeddings, denoted $X_{i,e}$.

From there, just like in the centralized algorithm, the unmatched cells are assigned based on the closest NN through a `Propagation` procedure. They are added to the matching, yielding Π_{i-1}^{prop}, which contains the existing indices and positions of new indices that are closest neighbors of existing indices. Then, these matches are sent to the server for alignment and redistributed back to the nodes. Once received, we used the previously calculated embeddings X_i^m to align with Π_{i-1}^{prop} and normalize it. Then, the matching from Algorithm 4 is performed, which ensures match uniqueness, redundant connections (duplicate connections with lower weight) are removed via the `RemoveRedundantConnections` step, yielding Π^*.

The resulting matches are further percentile filtered using a second threshold β to obtain a final matching Π^{final}, which is then redistributed to local nodes as $\Pi^{\text{final}}_{(i-1,2)}$. Once received, each node leverages the CCA weights $X_i^{m,e}$ calculated on the filtered pivots matching to compute the final embeddings $X_{i,e}$.

Algorithm 6 Federated MaxFuse Final Filtering

```
For i = (1, 2) which denotes a node index:
Input:
  • Meta-cell dataset $X_i^m \in \mathbb{R}^{N_{x_i}^m \times p_{x_i}}$
  • Active Dataset $X_i \in \mathbb{R}^{N_{x_i} \times p_{x_i}}$
  • NN graph for fuzzy smoothing $G_{X_i}$ together with interpolation
    weight $w \in (0.0, 1.0)$
  • Refined Matching $\Pi_{(i-1,\,2)} = \{(i_k/j_k,\ w_k)\}_{k=1}^n \subset \mathbb{Z}^+ \times \mathbb{R}$
  • Pivot and propagation filtering weight $\alpha, \beta \in (0.0, 1.0)$
Output:
  • Final matching $\Pi^{\text{final}}_{(i-1,\,2)}$
  • Final embedding $X_{i,\,e}$

// Step 6: Filtering and Final Embedding
// NODE 1&2
1:  $\Pi^{\text{pivot}}_{(i-1,\,2)} \leftarrow$ FilterOutBadMatches($\Pi_{(i-1,\,2)}, \alpha$)

// SERVER: CCA embedding fitting
// Federated Learning Orchestration
2:  $(X_1^{m,e}, X_2^{m,e}) \leftarrow$ FitEmbeddingsCCA($X_1^m[\Pi_0^{\text{pivot}}], X_2^m[\Pi_1^{\text{pivot}}]$)

// NODE 1&2
3:  $\Pi_{i-1}^{\text{prop}} \leftarrow$ Propagate($\Pi^{\text{pivot}}_{(i-1,2)}$, $X_i^{m,e}$)
4:  Send(
      $\Pi_{i-1}^{\text{prop}}$,
      to ='SERVER',
      action='PROP_MATCHING'
    )
// SERVER: Join Propagation indices
5:  Receive($\Pi_{i-1}^{\text{prop}}$, from='i', action='PROP_MATCHING')
6:  $\Pi^{\text{prop}} \leftarrow$ JoinAndAlign($\Pi_0^{\text{prop}}, \Pi_1^{\text{prop}}$)
7:  Send($\Pi_{i-1}^{\text{prop}}$, to='i', action='PROP_MATCHING')

// NODE 1&2
```

```
 8: Receive($\Pi^{\text{prop}}_{i-1}$, from='SERVER', action='PROP_MATCHING')
 9: $\widetilde{X}_i'$ ←FuzzySmoothing($X^m_i(\Pi^{\text{prop}}_{i-1})'$, $G_{X_i}$, $w$)
10: Send($\widetilde{X}_i'$, to='SERVER', action='FINAL_MATCHING')

    // SERVER: Orchestrate Pivot Propagation
11: Receive($\widetilde{X}_i'$, from='i', action='FINAL_MATCHING')
12: $\{(i_k, j_k, w_k)\}_{k=1}^n$ ← await ServerMatching($\widetilde{X_1^\circ}'$, $\widetilde{X_2^\circ}'$)
13: $\Pi^*$ ← RemoveRedundantConnections($\{(i_k, j_k, w_k)\}_{k=1}^n$)
14: $\Pi^{\text{final}}$ ← FilterOutBadMatches($\Pi^*, \beta$)
15: Send(($\Pi^{\text{final}}_{i-1}$, $\Pi^{\text{final}}_2$), to='i')

    // NODE 1&2
16: Receive($\Pi^{\text{final}}_{(i-1,\,2)}$, from='SERVER')
17: $X_{i,\,e}$ ← GetEmbeddingsCCA($X_i$, $X_i^{m,e}$)
18: return $\Pi^{\text{final}}_{(i-1,\,2)}$, $X_{i,\,e}$
```

2.2.2 Federated CCA breakdown. Canonical Correlation Analysis (CCA) is a statistical method used to identify and quantify associations between two sets of multivariate random variables. For instance, $X = \{X_1, X_2, ..., X_n\} \in \mathbb{R}^{N\times n}$ and $Y = \{Y_1, Y_2, ..., Y_m\} \in \mathbb{R}^{N\times m}$ where both sets have the same number of observations N and possibly different number of features n and m. The goal is to find linear combinations, known as canonical variates, $u = X\mathbf{a}$ and $v = Y\mathbf{b}$, that are maximally correlated [31]. This is achieved by finding canonical weight vectors $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ that solve the optimization problem shown in Equation 4.

$$\max_{\mathbf{a},\mathbf{b}} \ \rho(u = X\mathbf{a},\ v = Y\mathbf{b}) \tag{4}$$

However, in practice, it has been shown that regular Canonical Correlation Analysis (CCA) often struggles with overfitting when the sample size is small relative to the number of variables, leading to spurious associations that do not generalize well [27, 57, 28]. To address these limitations, Two-block Mode B Partial Least Squares (PLS) [24, 81] offers an alternative to traditional Canonical Correlation Analysis (CCA) computation. For C shared components, it constructs latent score vectors $\xi_c = X\cdot u_c \in \mathbb{R}^N$ and $\omega_c = Y\cdot v_c \in \mathbb{R}^N$ through iterative weighted linear combinations, with the weight vectors $u_c \in \mathbb{R}^n$ and $v_c \in \mathbb{R}^m$ estimated to prioritize indicators that contribute most to the predictive association. Additionally, loadings are calculated γ_c and δ_c, which are coefficients that map latent variables back to the original data dimensions as presented in Equation 5 and 6, where there is some small error ε_X and ε_Y if we were to run the algorithm for all features for their respective data matrices [59]. This predictive associativity is especially advantageous when indicator quality is heterogeneous [57] and has been applied in fields such as chemometrics, where it has helped model relationships between spectral data and chemical properties [82] and omics-type data [59]. Due to these reasons, the authors of MaxFuse [12] use Two-block Mode B PLS as proposed by Wegelin [78], which is the part we are making federated as seen in Algorithm 7.

$$X = \xi\cdot\gamma^\top + \varepsilon_X \tag{5}$$

$$Y = \omega\cdot\delta^\top + \varepsilon_Y \tag{6}$$

In Algorithm 7, each node normalizes their data in parallel which creates their first residuals $X_1^1 \in \mathbb{R}^{N\times n}$ and $X_1^2 \in \mathbb{R}^{N\times m}$, where N is the number of samples and n and m the number of features in each node (also referred to as view). Then a loop begins in which we calculate all the approximate singular vectors (u_c and v_c), weights (ξ_c and ω_c), and loadings (γ_c and δ_c) for index $c \in \{1, \ldots, C\}$, are computed to identify directions of shared variation. The approximate singular vectors are calculated using the singular vector power method [78] (see subsubsection 2.2.3), where the goal is to calculate the maximized cross-correlation $\mathbf{C}$ matrix between X^1 and X^2 where $\mathbf{C} = (X_c^1)^\top X_c^2$ such that $u_c^\top\cdot u_c = v_c^\top\cdot v_c = 1$ and $(u_c)_i > 0$, where $i = \arg\max|(\mathbf{u}_r)_i|$. In other words, find the index i of the element in $\mathbf{u}_r$ with the most significant absolute value while ensuring that at the end, it will become a positive value after a possible flip [78].

In the following stages of the loop, the latent score vectors ξ_c and ω_c represent projections of the data onto the latent space. This weighting is applied by regressing the original data onto the latent scores, loading vectors γ_c and δ_c. After projecting and reconstructing the approximated views $\hat{X}^1{}_c = \xi_c\cdot\gamma_c^\top$ and $\hat{X}^2{}_c = \omega_c\cdot\delta_c^\top$, residuals X_{c+1} and Y_{c+1} are updated through regression from which the next residual which will be used to calculate singular vectors $\mathbf{u}_{c+1}$ and $\mathbf{v}_{c+1}$. This continues until the C number of components is calculated, which is defined by the algorithm orchestrator, i.e., the server. Lastly, every score and loading vector is placed into its component index c, and combined to form the transformation matrices $U\cdot\Gamma^\top$ for X^1 and $V\cdot\Delta^\top$ for X^2. To apply the CCA transformation, each node multiplies its local data by the corresponding transformation matrix.

Algorithm 7 Federated Canonical Correlation Analysis

```
Input (Node 1 & 2): For i = (1, 2) which denotes a node index:
  • Data $X^i \in \mathbb{R}^{N\times n_i}$
  • Boolean $b_i$ to determine if only fitting should be performed
Input (Server): shared components number $C$
Output: Fitted embedding or full data embeddings:
  • Node 1: $U\cdot\Gamma^\top$ or $U\cdot\Gamma^\top\cdot X^1$ depending on $b_1$
  • Node 2: $V\cdot\Delta^\top$ or $V\cdot\Delta^\top\cdot X^2$ depending on $b_2$

    // NODE 1&2: Normalize data
 1: $X_1^i$ ← normalize($X^i$)
    // SERVER: Initialize main loop
 2: for $c$ ← 1 to $C$ do
    // Coordinate finding eigenvectors
 3:     $\mathbf{u}_c, \mathbf{v}_c$ ← SingularVectorPowerMethod($X_c^1, X_c^2$)

    // NODE 1: calculate scores and weights
 4:     $\xi_c \leftarrow X_c^1\cdot\mathbf{u}_c$
 5:     $\gamma_c^\top \leftarrow \xi_c(\xi_c^\top\xi_c)^{-1}\xi_c^\top X_c^1$
 6:     $\hat{X}^1{}_c(\xi_c) \leftarrow \xi_c\cdot\gamma_c^\top$
 7:     $X_{c+1}^1 \leftarrow X_c^1 - \hat{X}^1{}_c(\xi_c)$

    // NODE 2: calculate scores and weights
 8:     $\omega_c \leftarrow X_c^2\cdot\mathbf{v}_c$
 9:     $\delta_c^\top \leftarrow \omega_c(\omega_c^\top\omega_c)^{-1}\omega_c^\top X_c^2$
10:     $\hat{X}^2{}_c(\omega_c) \leftarrow \omega_c\cdot\delta_c^\top$
```

```
11:     $X^2_{c+1} \leftarrow X^2_c - \hat{X^2}_c(\omega_c)$
12: end for

    // NODE 1: Transform data into embedding
13: $U \leftarrow \begin{bmatrix} u_1 & u_2 & \dots & u_C \end{bmatrix}$
14: $\Gamma \leftarrow \begin{bmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_C \end{bmatrix}$
15: if $b_1 == 1$ then
16:     return $U \cdot \Gamma^\top$
17: else
18:     return $U \cdot \Gamma^\top \cdot X^1$
19: end if

    // NODE 2: Transform data into embedding
20: $V \leftarrow \begin{bmatrix} v_1 & v_2 & \dots & v_C \end{bmatrix}$
21: $\Delta \leftarrow \begin{bmatrix} \delta_1 & \delta_2 & \dots & \delta_C \end{bmatrix}$
22: if $b_2 == 1$ then
23:     return $V \cdot \Delta^\top$
24: else
25:     return $V \cdot \Delta^\top \cdot X^2$
26: end if
```

2.2.3 *Singular Vectors Power Method Federation.*

To support the distributed computation of canonical vectors in a federated setting with multiple message exchanges, we need to adopt the Singular Vectors Power Method [78] and modify it into a federated version, as shown in Algorithm 8. As mentioned in the previous paragraph, the goal is to approximate the singular vectors of the cross-correlation matrix **C** between the residual data blocks X^1_c and X^2_c. This implementation supports Mode B PLS (CCA-equivalent) version of the algorithm, which requires the use of Moore–Penrose pseudo-inverses [19, 78] locally on each node through the PseudoInverse procedure as described by Wegelin [78] and shown in Equation 7.

$$X^\dagger = (X^\top \cdot X)^{-1} \cdot X^\top \tag{7}$$

The initialization of the latent scores on each node with a random vector is interpolated with weight w with the first feature column $X^{\mathrm{i}}_c[:, 0]$ by utilizing Equation 8, so that a column of real data is not shared. The picking of a good interpolation value w is explored further in subsubsection 3.2.4. The nodes then send to server that they want to begin the loop which ends either when we iterated for K times (20,000 was recommended by MaxFuse) or once the absolute dot product difference between u_k and u_{k-1}, $\delta = \|u_k - u_{k-1}\|$ is smaller than some small tolerance value ϵ (1^{-6} was recommended by the authors of MaxFuse). This loop limit means we have approximated the eigenvectors of **C** closely. Therefore, a large K is recommended. During each iteration, nodes exchange their current projected latent scores ξ_k and ω_k, respectively, via the central server. Each node then updates its salience vectors u_k and v_k, using the pseudo-inverse of its local residual data block and the received latent score. These updated saliences are then normalized to unit length to ensure convergence stability. Once convergence is achieved, each node returns its final weight vector, corresponding to the cross-correlation structure's first singular direction between X_c and Y_c. It is used further inside CCA component.

$$\begin{aligned} \texttt{NoiseInterpolation}(v, w) &= (1-w)\cdot v + w \cdot \mathcal{N}(0,1)^n \\ \textbf{where:}\quad & v \in \mathbb{R}^n, \quad w \in (0.0, 1.0] \end{aligned} \tag{8}$$

Algorithm 8 Federated Singular Vectors Power Method

```
For i = (1, 2) which denotes a node index:
Input (Node 1&2): Residuals $X^{\mathrm{i}}_c \in \mathbb{R}^{N \times n_i}$
Input (Server):
  • Maximum number of iterations $K$
  • Small tolerance $\epsilon$
  • Interpolation weight $w$
Output: Weight vector $u_k \in \mathbb{R}^{n_1}$ or $v_k \in \mathbb{R}^{n_2}$ depending on Node

    // NODE 1&2: Initialize scores and pseudo-inverse
1:  $\xi_1$ or $\omega_1 \leftarrow$ NoiseInterpolation($X^{\mathrm{i}}_c[:, 0], w$)
2:  $X^{\mathrm{i},\dagger}_c \leftarrow$ PseudoInverse($X^{\mathrm{i}}_c$)

    // Server: Initialize loop
3:  $k \leftarrow 1$
4:  $\delta \leftarrow \infty$
5:  while $\delta > \epsilon$ and $k < K$ do
    // NODE 1
6:      Send($\xi_k$, to='2', via='SERVER')

    // NODE 2
7:      Send($\omega_k$, to='1', via='SERVER')

    // NODE 1&2: Update salience $u_k$ or $v_k$
8:      Receive($\omega_k$ or $\xi_k$, from='2 or 1')
9:      $l_k \leftarrow \omega_k$ or $\xi_k$
10:     $u_k$ or $v_k \leftarrow X^{\mathrm{i},\dagger}_c \cdot l_k (l_k^\top l_k)^{-1}$
11:     $u_k$ or $v_k \leftarrow$ Normalize($u_k$ or $v_k$)
12:     $\omega_{k+1}$ or $\xi_{k+1} \leftarrow X^{\mathrm{i}}_c \cdot (u_k$ or $v_k)$

    // NODE 1: Send convergence differential
13:     $\delta \leftarrow \|u_k - u_{k-1}\|$
14:     Send($\delta$, to='SERVER')

    // SERVER: increment loop counter
15:     $k \leftarrow k + 1$
16: end while
    // NODE 1&2: Return weight vector
17: return $u_k$ or $v_k$
```

2.2.4 *Message safety analysis.*

The federated algorithm was run on the cluster within a single process that simulates a centralized FL system, where each message passes through the centralized server, mimicking the architecture shown in Figure 1. However, 'fuzzily smoothed' and normalized subsets of data and CCA scores are passed instead of sending the neural network weights as in conventional federated neural network training, as defined in Algorithm 5, 6, 7, and 8.

We assume that communication between clients and the central server occurs over a secure channel, such as a VPN or other private network [72], to minimize the risk of interception by external parties.

Additionally, we assume that participants are non-malicious and that the central server follows an honest-but-curious model, meaning it follows the protocol but may attempt to infer private information from the received data [5]. As discussed in subsubsection 2.2.4, extracting meaningful information under these assumptions is mathematically nontrivial but still warrants further investigation.

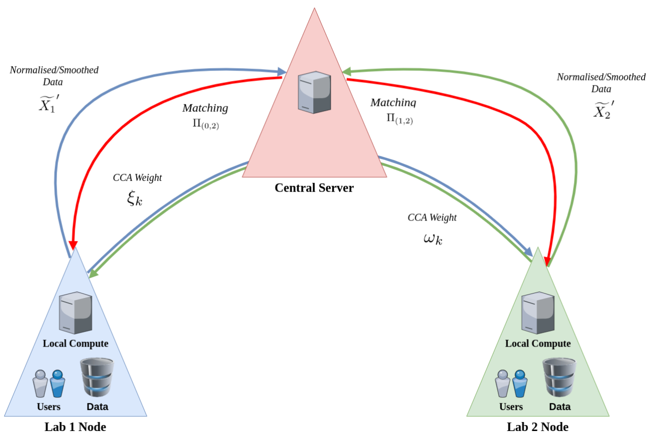

Figure 1: Federated MaxFuse architecture showing the communication between nodes and the central server.

Despite Federated MaxFuse circumventing direct data sharing between participating nodes, intermediary communications such as smoothed features, canonical correlation projections, and matching scores are still exchanged. Understanding the nature and potential risks associated with these shared messages is crucial, particularly when sensitive biological information is involved. This subsection scrutinizes the categories of messages transmitted during the federated integration process, examines what underlying information they may reveal, and discusses the trade-offs between communication efficiency, model performance, and data privacy. By systematically evaluating each shared variable and its associated algorithmic operation, we aim to assess the potential vulnerabilities and propose safeguards where necessary. In Table 3, we investigate this through the lens of the variables transmitted in the algorithm.

2.3 Evaluation

Both supervised and unsupervised measures were used to assess the efficacy of federated MaxFuse. We use cell labels to evaluate supervised alignment accuracy on both datasets (Antibodies and Tonsils), based on shared cellular classification labels across modalities and the Antibodies benchmark dataset, which features explicit cellular-level correspondences for alignment quality. In this context, federated MaxFuse embeddings generated across distributed silos are aggregated centrally, alongside their known ground-truth correspondences. The unsupervised evaluation techniques are also implemented to alleviate potential constraints associated with supervised methods, such as erroneous labels or inconsistent labeling across datasets. These techniques evaluate the embeddings produced by MaxFuse by scrutinizing the consistency and structural integrity in relation to the intrinsic characteristics of the original datasets [37].

2.3.1 Supervised. Supervised measures assess the quality of the produced matching and embeddings based on pre-existing knowledge, such as cell sample labels and existing matches. For this category, there are two sub-categories for evaluation: data with existing matching and per-sample labels, and data with only per-sample labels.

Firstly, we have measurements that measure the quality of the 1–1 sample-level ground-truth matching as provided by researchers associated with MaxFuse: Fraction of Samples Closer Than True Match (FOSCTTM) [52, 12] and Fraction Of Samples whose true matches are among their K-Nearest Neighbors (FOSKNN) [12]. For clarity, we assume that benchmarked evaluation occurs on a single computer, where matching and data are represented as distances between points that must be calculated. The FOSCTTM score, as seen in Equation 9, is calculated by getting Euclidean distances $\|\cdot\|_2$ between two modalities, which subsequently are used to calculate the proportion of samples that are situated nearer to the predetermined sample x and y than to their authentic correspondences $(x_{\mathtt{match}}, y_{\mathtt{match}})$ for $N = |X| = |Y|$ samples. The $\mathbb{I}(\cdot)$ is an indicator function, evaluating to 1 if the condition holds and 0 otherwise. In an optimal alignment setting, all samples would exhibit proximity to their genuine match, resulting in a zero score. Therefore, a lower FOSCTTM indicates superior alignment efficacy. Conversely, a high score suggests that incorrectly matched cells are nearer to one another than to the accurate ground-truth matches. Likewise, FOSKNN assesses alignment effectiveness at the single-cell level by determining whether, for each cell embedding $x_i \in X$ from one modality, its actual corresponding embedding $z_i \in Z$ from the alternate modality is among its k-NNs, denoted as $\mathbb{I}(x_i \in \mathrm{kNN}(y_i, X, k))$ in Equation 10, in the joint embedding space, and vice versa, denoted as $\mathbb{I}(y_i \in \mathrm{kNN}(z_i, Y, k))$ in Equation 10. Formally, FOSKNN computes the average proportion of successful matches across all cells, where a higher score indicates better integration quality. Following the recommendation by the authors of MaxFuse, the parameter k is typically set to 5% of the dataset size N, and we plot this by running FOSKNN from 1 to 5% of the data.

$$\begin{aligned}
&\mathtt{FOSCTTM}(X, Y, N, (x_{\mathtt{match}}, y_{\mathtt{match}})) = \\
&\quad \frac{1}{2N}\left(\sum_{x\in X}\frac{\sum_{y\in Y}\mathbb{I}(\|x-y\|_2 < \|x-y_{\mathtt{match}}\|_2)}{N} + \right. \\
&\quad \left. \sum_{y\in Y}\frac{\sum_{x\in X}\mathbb{I}(\|y-x\|_2 < \|y-x_{\mathtt{match}}\|_2)}{N}\right)
\end{aligned} \tag{9}$$

$$\begin{aligned}
&\mathtt{FOSKNN}(Y, Z, N, k) = \\
&\quad \frac{1}{2N}\left(\sum_{i=1}^{N}\mathbb{I}(z_i \in \mathrm{kNN}(y_i, Z, k)) + \sum_{i=1}^{N}\mathbb{I}(y_i \in \mathrm{kNN}(z_i, Y, k))\right)
\end{aligned} \tag{10}$$

Secondly, we use label-based metrics to evaluate integration quality based on predefined labels: Accuracy [12], Average Silhouette Width F1 (ASW F1) [74], and Adjusted Rand Index F1 (ARI F1) [74]. The accuracy score is simply the percentage of labels that match after alignment. This explains how effectively labels from different modalities correspond to the same cell types. Furthermore, ASW F1 and ARI F1 are used to measure clustering quality, particularly

Variable	Algorithms	Description and Reasoning
$\widetilde{\mathrm{X}}'$	3, 5, 6,	L2-normalized and fuzzily smoothed feature matrix transmitted to the server for both initial and iterative matching. Although the data is simplified by smoothing toward neighborhood centroids, reducing the risk of direct data leakage, local structural patterns could still be exposed if intercepted. The risk is minimized, however, as only specific forms are shared: the normalized and smoothed shared columns $\widetilde{\mathrm{X}_{\mathrm{i}}^{\circ}}'$ (see Algorithm 3), the normalized and smoothed CCA embeddings $\widetilde{\mathrm{X}_{\mathrm{i}}^{\mathrm{cc}}}'$ (see Algorithm 5), and the normalized and smoothed full datasets for final matching $\widetilde{\mathrm{X}_{\mathrm{i}}}'$ (see Algorithm 6).
Π	3, 5, 6	Matched indices and similarity scores between modalities. These are primarily computed by the server and redistributed to the nodes, except during propagation matching ($\Pi_{\mathrm{i-1}}^{\mathrm{prop}}$), which requires server-side joining and realignment. While index exposure alone does not reveal feature content, it could potentially hint at sample relationships if improperly aggregated or analyzed.
ξ_k, ω_k	8	Latent projections exchanged during iterative CCA fitting via the singular vector power method, a core component of the FL framework. They capture directional trends in local data, but individual cell-level details remain obscured.
δ	8	This denotes convergence metric is a scalar norm difference during CCA convergence checks. Contains no identifiable information about the raw data.

Table 3: Summary of variables exchanged during Federated MaxFuse, associated algorithms, and reasoning about privacy exposure in the honest but curious model.

for assessing batch effect correction using labels. The Average Silhouette Width (ASW) Score, which has a range $[-1,\ 1]$, measures how similar an object is to its cluster with other clusters. On the other hand, scores close to 0 suggest cluster overlap, and high values show substantial intra-cluster and weak inter-cluster similarities. Likewise, the Adjusted Rand Index (ARI) increases the robustness of clustering comparisons by assessing the agreement between two clusterings, corrected for random chance, with a range $[-0.5,\ 1]$, where 0 represents random labeling and 1 represents perfect alignment.

$$\begin{aligned} &\text{Label-F1} = \frac{2 \cdot (1 - B_{\texttt{norm}}) \cdot C_{\texttt{norm}}}{(1 - B_{\texttt{norm}}) + C_{\texttt{norm}}} \\ &\textbf{where}: \\ &B_{\texttt{norm}} \Leftarrow \texttt{NormMedian}(\text{ARI/ASW}(\text{batch labels}, k_{\text{batch}})) \\ &C_{\texttt{norm}} \Leftarrow \texttt{NormMedian}(\text{ARI/ASW}(\text{cell-type labels}, k_{\text{cell-type}})) \end{aligned} \tag{11}$$

These F1 scores are computed using Equation 11, which includes two key components: $B_{\texttt{norm}}$ and $C_{\texttt{norm}}$. The $B_{\texttt{norm}}$ is a result of sub-sampling a batch of data (recommended 80% by [74] and [12]), reducing them to a couple of principal components (recommended 20 [74] and [12]), setting the label as the modality and running either ASW F1 or ARI F1. The function `NormMedian` performs normalization, mapping the values to the $[0, 1]$ range, and calculates the median. Since ARI compares two clusterings, it requires a reference clustering obtained through k-means clustering [55], with k being the number of modalities for $B_{\texttt{norm}}$ score and cardinality of unique cell-type labels for $C_{\texttt{norm}}$ score. The $C_{\texttt{norm}}$ parameter is to run the ASW or ARI score based on the cell-type labels. All these scores are normalized between 0.0 and 1.0, the median value of which is plugged into the equation. The higher F1 values indicate the effective removal of batch effects and accurate cell-type label clustering [74].

2.3.2 Unsupervised. As mentioned earlier, unsupervised evaluation provides a practical methodology for assessing the confidence associated with membership and the stability of clusters across diverse data modalities, all while eliminating the need for ground-truth labels, as highlighted in the work of Chung et al. [13]. These measurements can be categorized into three distinct groups, namely global metrics, cluster metrics, and local metrics, as discussed by Jeon et al. [37]. After dimensionality reduction, 'local' metrics measure how well the regional relationships between data points are maintained. The 'cluster' metrics evaluate whether data belonging to the same cluster in the original space remains in the reduced space. Finally, the overarching retention of data structures and the identification of patterns present within the data are quantitatively measured by employing global metrics.

Firstly, we selected Mean Relative Rank Errors (MRRE) [48] for the 'local' measurement. MRRE is divided into two types of errors: 'false neighbors' evaluates the proportion of incorrectly introduced neighbors (false positives), and 'missing neighbors' evaluates the proportion of actual neighbors lost during the dimensionality reduction (false negatives). These are formally captured in Equation 13 and 14, both of which are directional variants of the general MRRE formulation shown in Equation 12 where 'false neighbors' the 'base ranking' R^{base} is of the original data (denoted as R^{orig}) and 'target ranking' R^{target} being of the embedding (denoted as R^{emb}). For the 'missing neighbors', it is the other way round. This metric provides a rigorous evaluation of the extent to which the reduced embedding preserves the original neighborhood relationships, considering both the introduction of erroneous neighbors and the exclusion of correct ones. This is calculated based on embedding and data with n rows, each with a precomputed set of k NNs per point, represented as N_i^{target} within a target space. Finding the ranks uses $R_i^{\text{base}}(j)$ and $R_i^{\text{target}}(j)$ which represent the position of neighbor $j \in N_i^{\text{target}}$ in the sorted distance list of point i within the base and target spaces, respectively. Then, the absolute difference between these ranks is divided by the target-space rank $R_i^{\text{target}}(j)$, which results in the total neighbor dislocation. Notably, MRRE hinges on comparing a point's global rank among all data points in both the original space and the embedding, since merely knowing the top-k neighbors cannot capture changes in absolute position, for instance going from the 5^{th} nearest neighbor to the 15^{th}. For each point i, the score is normalized by c, defined in Equation 15, and averaged out by n to ensure values in the overall score fall within $[0.0, 1.0]$, where lower scores indicate better preservation. This rank-based formulation makes MRRE especially sensitive to alterations in local structure induced by dimensionality reduction.

$$\text{MRRE} = \frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{\frac{1}{c}\sum_{j\in N_i^{\text{target}}}\frac{|R_i^{\text{base}}(j) - R_i^{\text{target}}(j)|}{R_i^{\text{target}}(j)}}_{\text{Rank error/distortion of neighbors}}\right) \tag{12}$$

$$\text{MRRE}_{\text{false}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{c}\sum_{j\in N_i^{\text{emb}}}\frac{\left|R_i^{\text{orig}}(j) - R_i^{\text{emb}}(j)\right|}{R_i^{\text{emb}}(j)}\right) \tag{13}$$

$$\text{MRRE}_{\text{missing}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{c}\sum_{j\in N_i^{\text{orig}}}\frac{\left|R_i^{\text{emb}}(j) - R_i^{\text{orig}}(j)\right|}{R_i^{\text{orig}}(j)}\right) \tag{14}$$

where:

$$c = \sum_{r=1}^{k}\frac{|n - 2r + 1|}{r} \tag{15}$$

Secondly, we selected Steadiness & Cohesiveness [36] for the' cluster' measurement. Steadiness measures the inter-cluster reliability inside the embedding (i.e., how closely the embedding's clusters reflect the state of clusters in the original data)[36]. At the same time, Cohesiveness measures the inter-cluster reliability in the original data (i.e., how closely the original data's clusters reflect the state of clusters in the embedding)[36]. Reliability is measured through the number of clusters, the number of outliers, the distance between clusters, and the distance between points. The measurement range is $[0.0, 1.0]$, with the optimal value being 1.0, meaning the excellent reliability of the projected clusters or superb reliability of the original data clusters. The clustering algorithm in this measure is based on the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering method, which is ideal for objective evaluation. It can create clusters, handle noise (separating artifacts), produce clusters of varying densities, and its hierarchical structure permits examination at multiple levels [8]. The metrics are placed into a workflow that incorporates randomness to capture complex inter-cluster structures better. Four key functions are required: a pointwise distance function, a cluster distance function, a cluster extraction function, and a clustering function. These are used iteratively to accumulate local distortions and assess space consistency.

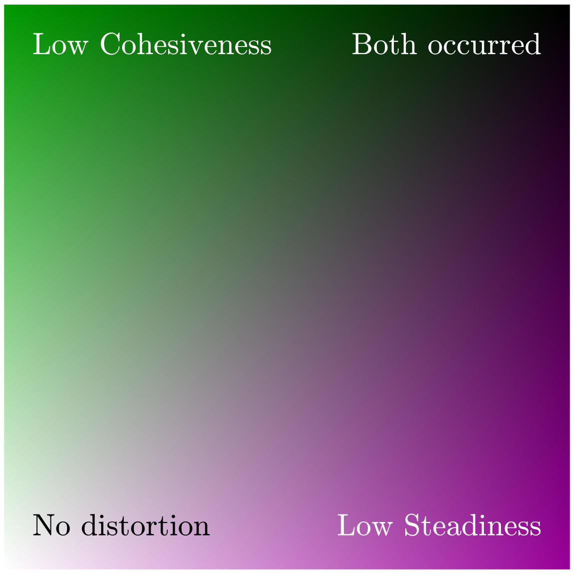

Figure 2: CheckViz and Reliability Map color map interpretation

To help visualize the distortions from the Steadiness & Cohesiveness metric, CheckViz [49] and the Reliability Map [36] were plotted. Both techniques utilize CheckViz to highlight tears and false neighborhoods, while the Reliability Map displays pointwise distortions through edge-based distortion encoding. Additionally, these can be plotted on a 2-dimensional axis chart, which illustrates how to interpret the scores, as shown in Figure 3. Additionally, a color map from Figure 2 guides interpreting CheckViz and Reliability Maps. The top-right corner (high steadiness and cohesiveness) represents high reliability, where the embedding and original spaces preserve each other's cluster structures well. The top-left quadrant (low steadiness and high cohesiveness) indicates the presence of false-positive structures in the embedding, where clusters exist in the embedding that do not exist in the original data. In contrast, the bottom right (high steadiness and low cohesiveness) suggests missing structures where genuine clusters from the original space are not captured in the embedding. Lastly, the projection on the bottom left (both low) suffers from false positives and missing values, reflecting a severe loss of structural fidelity in both directions.

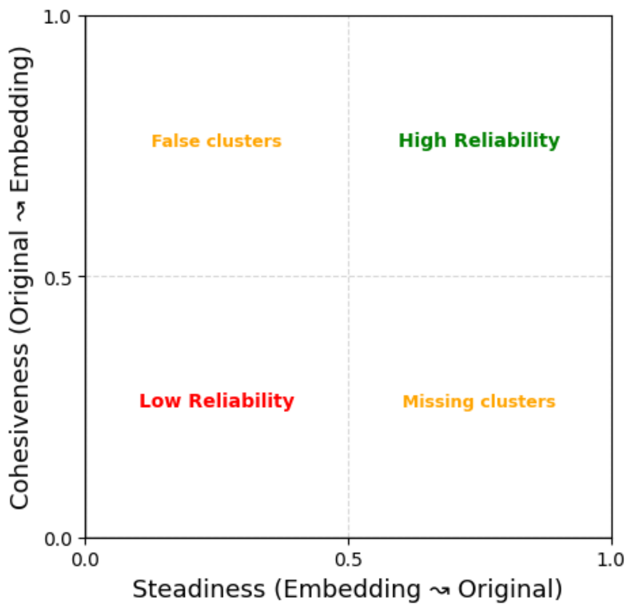

Figure 3: Steadiness & Cohesiveness Interpretation

Finally, we used the scale-normalized version of Kruskal's Stress [45, 46, 60] as our 'global' measure, which is calculated as seen in Equation 16. This metric quantifies the discrepancy between the distances in the low-dimensional d_{ij}^{low} embedding and the original data d_{ij}^{high} over all unique pairs of points $i < j$, to avoid double-counting in symmetric distance matrices. Specifically, it adjusts for scaling effects by introducing an optimal linear scaling factor α, thereby providing a scale-invariant assessment of embedding quality as seen in Equation 17. The values are real positive numbers, and the optimal value is 0.0, representing a perfect embedding in which all pairwise distances are preserved.

$$\text{Stress} = \sqrt{\frac{\sum_{i<j}\left(d_{i,j}^{\text{high}} - \alpha \cdot d_{i,j}^{\text{low}}\right)^2}{\sum_{i<j}\left(d_{i,j}^{\text{high}}\right)^2}} \tag{16}$$

$$\alpha = \frac{\sum_{i<j} d_{i,j}^{\text{high}} \cdot d_{i,j}^{\text{low}}}{\sum_{i<j}\left(d_{i,j}^{\text{low}}\right)^2} \tag{17}$$

2.4 Computational Resources and Software Environment

The first version of the MaxFuse algorithm is publicly available as a complete Python package. Its source code is also available on GitHub[2]. The MaxFuse repository contains a wide range of materials, including Python code, scripts, and Jupyter Notebooks for interactive computing, primarily used for data collection and analysis, thereby enhancing the overall research experience. This federated version of MaxFuse was entirely written in Python, including analysis scripts.

[2]MaxFuse repository: https://github.com/shuxiaoc/maxfuse/

The computational workflow of the federated implementation is showcased through Jupyter Notebooks with in-depth analyses. It significantly depends on a set of widely utilized libraries, such as 'AnnData' for managing annotated data, 'scikit-learn' for machine learning purposes, and 'numpy' for numerical computations, thereby enhancing the implementation's robustness. A particularly significant enhancement to this framework is the incorporation of the ZADU package [37], which provides sophisticated unsupervised evaluation methodologies designed explicitly for assessing embeddings, thereby enriching the algorithm's analytical capabilities.

Additionally, it requires additional resources to run evaluations, which involve calculating distances and NN graphs, as well as NN ranks. The experiments were carried out on the Delft AI Cluster (DAIC)[3] (runs Slurm workload manager) to manage the computational demands associated with this investigation efficiently. Finally, it is essential to mention that the entire codebase for the federated version of the MaxFuse method is now publicly accessible on https://kbaran1998.github.io/fed-maxfuse, enabling wider distribution and potential collaborative improvements among researchers.

3 EXPERIMENTATION

This section presents the details of the experimental setup and the implementation techniques used to run the MaxFuse algorithm in both centralized and federated environments, along with the steps taken to retrieve the output embeddings and the final matching outcomes. The setup details include data distribution practices, a description of the federated architecture used for the experiments, the specific parameter settings employed for each implementation variant, and a detailed description of the various experiments conducted.

3.1 Experimental Setup

We evaluated the MaxFuse algorithm in centralized and federated settings across a series of controlled experiments run as SLURM jobs on the DAIC computational cluster, as indicated in subsection 2.4. Each job produced embeddings and cell-to-cell matchings, which were assessed through separate SLURM jobs using supervised and unsupervised evaluation metrics. Supervised evaluation compares the results against ground-truth labels, while unsupervised evaluation assessed embedding quality relative to the original input data (see subsection 2.3). Metrics from both evaluation types were used to analyze performance across different experimental conditions. Additional implementation details can be found in Appendix A.

The hyperparameters used in our experiments follow the default settings provided in the official MaxFuse example implementations [12]. Complete configuration files, including training parameters, graph construction settings, and evaluation criteria, are available in Appendix C to support reproducibility. The parameter variability depends on the experiment described in the following sections, and each parameter combination was run multiple times to account for algorithmic stochasticity and ensure robust performance estimates. The total number of runs per dataset and experiment is summarized in Table 4, broken down by dataset type and grouped by key experimental conditions: centralized vs. federated setups, CCA initialization randomness, shared parameter tuning, and batching strategies. Each

[3]Delft AI Cluster system specifications: https://doc.daic.tudelft.nl/docs/system/

cell in the table indicates the number of unique configurations multiplied by the number of repetitions, which are further described in subsection 3.2. The final column displays the total number of runs per dataset, totaling 2,600 experiments.

3.2 Experiment Types

This section provides a comprehensive summary of the experiments evaluating the MaxFuse algorithm's performance, resilience, and flexibility in various settings. The purpose of these studies is to investigate how algorithm performance varies across multiple parameter settings and data distribution scenarios, as well as to highlight the variations in results between traditional (centralized) and federated learning configurations. The algorithm's computing efficiency, matching accuracy (if applicable), and embedding quality were evaluated using both supervised and unsupervised assessment measures for each experiment type. Important parameters were adjusted to assess their impact on the results and the feasibility of extending MaxFuse to federated applications, including the number of SVD components, CCA components, and loop iterations. Every experiment type investigates distinct MaxFuse algorithm variations or settings.

Additionally, in these experiments, we refer to datasets as 'meta-' and 'non-meta-' modality datasets, where 'meta' means that we reduce the sample size using Leiden clustering to create meta-cells in the initial matching and refinement loop, and later expand the matching to assign previously unmatched cells. The 'non-meta-' modality dataset uses the original samples during those parts of MaxFuse.

3.2.1 Centralized versus Federated Comparison. To thoroughly evaluate the feasibility of implementing MaxFuse in a federated manner, an in-depth comparison with its centralized counterpart is essential. We utilized the same hyperparameter settings for the federated and centralized configurations within this experimental framework.

We employed the non-parametric Mann-Whitney U examination to evaluate whether the performance distributions of the centralized and federated models exhibit significant divergence, utilizing a significance threshold of 0.05. We selected this threshold to reflect the exploratory nature of our work, the applied context of model comparison, and the intrinsic stochasticity of MaxFuse, where a less stringent threshold is appropriate.

3.2.2 Centralized Data Batching. While batching is commonly used to manage large datasets and reduce variability, our primary objective in this experiment is to investigate whether MaxFuse can be extended beyond a simple two-participant setting in a federated environment, where each node either holds distinct feature sets or applies internal batching. Although the original MaxFuse implementation supports data batching, its impact on integration quality has not been systematically evaluated. We simulate batching within a centralized setup to establish a controlled baseline, approximating a federated scenario with two nodes. Due to the increased complexity of batching implementation in the federated framework, we limited batching experiments to the centralized setting.

In this batched setup, matching and CCA are conducted between each pair of opposite modality batches, and only the best-performing correspondences are preserved after the main loop to construct the final matching and embeddings. Furthermore, this methodology addresses memory limitations and facilitates a more resilient alignment of datasets exhibiting high variability and has previously been executed in the centralized MaxFuse but not evaluated as comprehensively.

Batch size affects not only computational efficiency but also the statistical properties of the resulting embeddings. While large batches can mask fine-grained patterns in the data, smaller batches may result in more variance in estimates, and both compromise the model's convergence [67]. Recent work explores optimal batch configurations to balance these trade-offs, underscoring the importance of experimental design that accounts for sample size and distribution across biological conditions [6].

We aim to evaluate the effectiveness of batching and its impact on final alignment quality. Due to the constraints of the original implementation, batching is governed by a tuple of three key hyperparameters (κ, ρ, μ): the maximum number of cells per batch κ, the average number of matches per cell ρ, and the average number of cells per metacell μ. This works where we have two input datasets $Y \in \mathbb{R}^{N_1 \times d_1}$ and $Z \in \mathbb{R}^{N_2 \times d_2}$ where Y typically represents the 'meta-dataset' and Z the 'non-meta-dataset'. The parameter κ sets an upper bound on how many individual cells or meta-cells from Y can exist within a single batch. The matching density is controlled by ρ, which specifies the average number of candidates in Z to be matched to each unit in Y. Finally, Y may be compressed into meta-cells, with μ defining the average number of raw cells per meta-cell.

Given these parameters, the maximum batch size for Z is computed as $\mathcal{B}_Z = \kappa \cdot \rho$. The number of batches for Z is then $B_Z = \left\lceil \frac{N_2}{\mathcal{B}_Z} \right\rceil$, and each batch has approximate size $b_Z = \left\lfloor \frac{N_2}{B_Z} \right\rfloor$. For Y, the maximum batch size depends on whether meta-cell aggregation is used. If μ is defined, we set $\mathcal{B}_Y = \left\lfloor \frac{b_Z}{\rho/\mu} \right\rfloor$ to balance the number of metacells in Y with the size of the corresponding Z batch. Otherwise, $\mathcal{B}_Y = \kappa \cdot \rho$. The number of batches for Y is then $B_Y = \left\lceil \frac{N_1}{\mathcal{B}_Y} \right\rceil$, and each batch is of size $b_Y = \left\lfloor \frac{N_1}{B_Y} \right\rfloor$. It is also important to note that these batches are randomly assigned and might also be unequally distributed across the dataset, if the total number of cells is not evenly divisible by the batch size (i.e., $b_Y \times B_Y < N_1$ or $b_Z \times B_Z < N_2$), the final batch will be smaller than the others but is nonetheless included in the algorithmic pipeline. The specific configurations used in our experiments are summarized in Table 5.

Finally, this batching experiment is a stepping stone toward more complex scenarios, such as implementing the Generalised Canonical Correlation Analysis (GCCA) approach. By analyzing the effects of batching versus full dataset processing, we can better understand the trade-offs in accuracy, efficiency, and ultimately assess the feasibility of scaling to more nodes in a federated network.

3.2.3 Federated Shared Feature Selection. Identifying shared features across multiple datasets can be challenging, especially when the exact correspondences between features are unknown. MaxFuse relies on these shared features to function correctly, but in practice, such correspondences might not always be available in advance. Before starting MaxFuse, a preprocessing pipeline was run, as described in subsubsection 2.1.3, which is necessary for MaxFuse

Dataset Type	**Number of experiments ran (configurations × repetitions)**					**Total**
	Centralized VS Federated		**CCA**	**Shared Parameters**	**Batching**	
	Centralized	**Federated**				
CITE-seq & PBMC Antibodies	1 × 250	1 × 250	11 × 25	11 × 25	10 × 25	*1300*
scRNA-seq & CODEX Tonsils	1 × 250	1 × 250	11 × 25	11 × 25	10 × 25	*1300*
Total	*500*	*500*	*550*	*550*	*500*	**2600**

Table 4: Distribution of experiment configurations and repetition counts.

CITE-seq *($N = 10,000$)* ***& PBMC*** *($N = 10,000$)* ***Antibodies***					***scRNA-seq*** *($N = 12,977$)* ***& CODEX*** *($N = 178,919$)* ***Tonsils***				
κ	ρ	μ	**Meta batches**	**Non-Meta batches**	κ	ρ	μ	**Meta batches**	**Non-Meta batches**
5,000	3	2	1 × 10,000	1 × 10,000	8,000	4	2	1 × 12,977	5 × 32,000 + 18,919
8,000	4	2	1 × 10,000	1 × 10,000	8,000	4	2	1 × 12,977	5 × 32,000 + 18,919
8,000	4	-	1 × 10,000	1 × 10,000	8,000	4	-	1 × 12,977	5 × 32,000 + 18,919
1,000	3	-	3 × 3,000 + 1,000	3 × 3,000 + 1,000	1,000	3	-	4 × 3,000 + 977	59 × 3,000 + 1,919
2,000	4	2	2 × 4,000 + 2,000	1 × 8,000 + 2,000	2,000	4	2	3 × 4,000 + 977	22 × 8,000 + 2,919
1,000	4	2	5 × 2,000	2 × 4,000 + 2,000	1,000	4	2	6 × 2,000 + 977	44 × 4,000 + 2,919
1,000	4	-	2 × 4,000 + 2,000	2 × 4,000 + 2,000	1000	4	-	3 × 4,000 + 977	44 × 4,000 + 2,919
1,000	4	5	2 × 4,999 + 2	2 × 4,000 + 2,000	1000	4	5	2 × 4,999 + 2,979	44 × 4,000 + 2,919
100	10	5	20 × 500	10 × 1,000	100	10	5	25 × 500 + 477	178 × 1,000 + 919
500	2	10	2 × 4,999 + 2	10 × 1,000	500	2	10	2 × 4,999 + 2,979	178 × 1,000 + 919

Table 5: Batching configuration pairs and corresponding parameter settings for evaluating the effect of batching on performance.

to create two aligned datasets with known feature correspondences. These are used to generate an initial matching, which initiates the primary iterative process (as shown in lines 1–12 of Algorithm 1 for the centralized version, and in Algorithm 3 for the federated version) and affects MaxFuse's convergence to the most optimal matching.

Therefore, this experiment investigates how the number and quality of shared features impact MaxFuse's performance. Specifically, we test two conditions: (1) how well MaxFuse performs when using only a random subset of the actual shared feature correspondences, and (2) whether randomly selected, non-corresponding features can be treated as shared and still lead to stable results. To evaluate this, we vary the randomly selected variables (2, 5, 20, 50, 100, and 150) and the proportion of true shared features used (10%, 20%, 50%, 80%, and 100%).

3.2.4 *Random Initialization in Federated CCA.*

In the federated version, as described in subsubsection 2.2.2, federated CCA, specifically the Singular Vector Power Method as delineated in Algorithm 8 (in lines 1 and 4), cannot simply distribute a stochastic vector from actual data to latent scores $\xi_1 \in \mathbb{R}^n$ and $\omega_1 \in \mathbb{R}^m$. We incorporate stochastic normal noise $\mathcal{N}(0,1)^n$ or $\mathcal{N}(0,1)^m$ and apply linear interpolation-based smoothing with $\alpha \in [0,1]$ interpolation ratio governing the extent of noise as seen in Equation 8. This facilitates the assessment of its impact on matching consistency and determines the degree of noise that can be initially introduced without jeopardizing the integration output. Therefore, we chose 10%, '20%, 50%, 70%, 90%, and 100% as interpolation values.

4 RESULTS

This section presents the evaluative insights regarding our federated and centralized implementations of the MaxFuse algorithm. The experimental design evaluates integration quality across multiple metrics, including both supervised and unsupervised ones. We focus on whether the federated MaxFuse preserves alignment and structural integrity compared to its centralized counterpart, without degrading embedding quality or matching performance.

To evaluate MaxFuse across different experiments, we deployed metrics that capture various integration performance dimensions, including alignment precision, cluster unity, and the maintenance of neighborhood ties. Supervised metrics, such as accuracy, ASW F1 and ARI F1, FOSCTTM, and FOSKNN, measure alignment quality by leveraging ground-truth labels and established cell correspondences, thus delivering powerful insights into the efficacy of direct match integration. Conversely, unsupervised metrics such as MRRE, Steadiness, and Cohesiveness evaluate structural preservation independently of labels, thus enabling us to scrutinize intrinsic data alignment quality within the federated framework.

Our findings highlight the potential trade-offs between performance and structure in the federated approach.

4.1 Centralized versus Federated Comparison

We perform an in-depth comparison with its centralized counterpart to evaluate the feasibility of implementing MaxFuse in a federated setting. We utilized the same hyperparameter settings for the federated and centralized configurations. Here, we employ a comprehensive set of evaluation metrics to analyze performance thoroughly. Additionally, we performed statistical tests to determine whether the differences between centralized and federated metrics were significant using the Mann-Whitney U test at a significance level of $p < 0.05$.

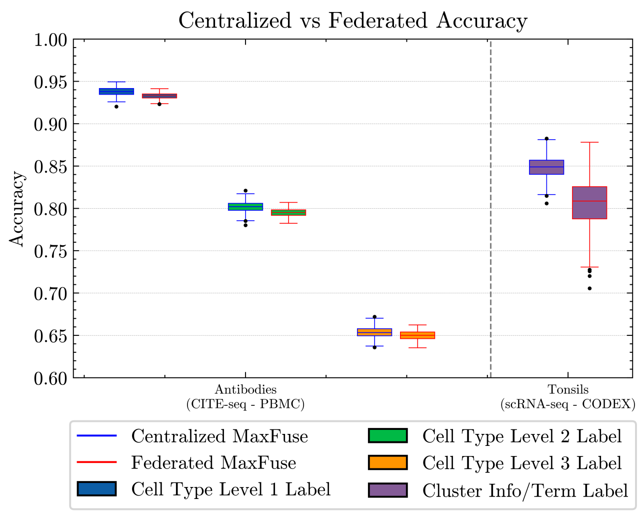

Figure 4: Summary of Accuracy per cell label on Federated versus Centralized MaxFuse

Firstly, comparison of classification performance between centralized and federated, as shown in the Figure 4, MaxFuse reveals that matching towards labels, whether at Level 1, 2, or 3 for the Antibodies dataset (CITE-seq and PBMC) and Cluster Info/Term for the Tonsils dataset (scRNA-seq and CODEX), generally seems to perform very similarly. However, we can observe that the federated setting closely aligns with the centralized version across various labels. However, statistically significant differences indicate that centralized integration slightly outperforms federated approaches for certain label types (see Table 6). The underlying cause for this discrepancy in the Antibodies dataset is likely implementation-related. Notably, these differences become less pronounced at finer label levels, where there is a higher number of unique labels (see Table 1), suggesting that federated learning retains most of its classification capability at detailed annotation levels, as well as at the centralized one. However, on the more heterogeneous Tonsils dataset, the federated setting exhibits greater variability and a noticeable drop in accuracy, with statistically significant differences that consistently favor the centralized approach. We believe this is caused by implementation issues and the fact that, by default, the centralized algorithm for the Tonsils dataset performs batching, which facilitates a greater number of possible cell-to-cell matches. At the same time, the federated version does not implement it owing to implementation complexity.

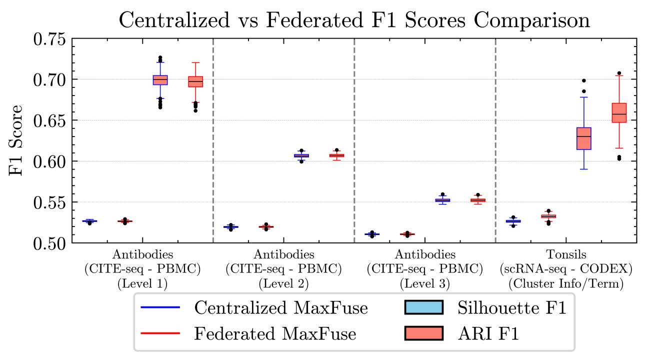

Figure 5: Summary of Average Silhouette Width F1 and Adjusted Rand Index F1 per cell label on Federated versus Centralized MaxFuse

Secondly, metrics ASW F1 and ARI F1 metrics show that federated MaxFuse achieves clustering quality comparable to that of its centralized counterpart(see Figure 5). Both methods perform similarly across all label levels in the Antibodies dataset, with minor, statistically insignificant differences at finer resolutions. For the Tonsils dataset, federated integration shows slightly greater variability but maintains strong overall performance. These results indicate that federated MaxFuse effectively preserves cell-type clustering and corrects for batch effects, validating its applicability for decentralized multi-omic data integration. However, it does come with a trade-off: the federated version appears to be much less stable, as its metrics seem rather varied.

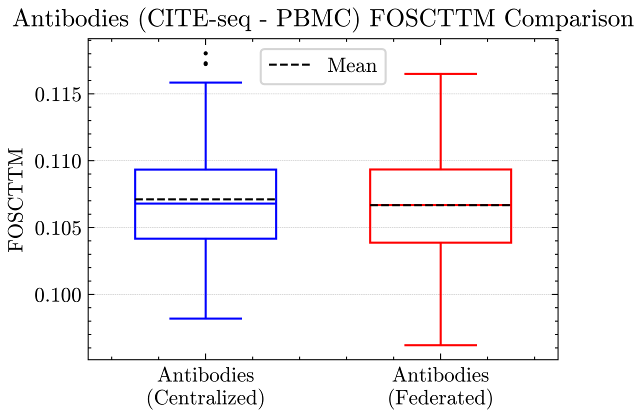

Figure 6: FOSCTTM on Federated versus Centralized MaxFuse.

Thirdly, further analysis of embedding quality, which we have only performed for the Antibodies dataset, through supervised metrics such as FOSCTTM (see Figure 6) and FOSKNN (see Figure 7) shows that federated MaxFuse achieves comparable performance to centralized MaxFuse. The near-identical curves in FOSKNN and the non-significant statistical differences indicate that local neighborhood structures are well-preserved in the federated setting (see Table 6). Similarly, unsupervised metric MRRE plots (Antibodies dataset) demonstrate largely overlapping performance between federated and centralized embeddings, and results lie within the green quadrants on the plot, which specifies that there were not so many missing or false neighbors (see Figure 8). This is noteworthy, given that MRRE evaluates based on the ranking of k-NN. This means that MaxFuse can place close neighbors in the almost correct order of proximity with very few wrongly assigned neighbors inside the embedding. However, a minor statistically significant difference is noted in the missing neighbors component for the meta-modality (see Table 6). The meta-modality also performs worse than the non-meta-modality due to the cells-as-cluster summarization as an output of Leiden's Algorithm [75], which could remove some crucial structural information.

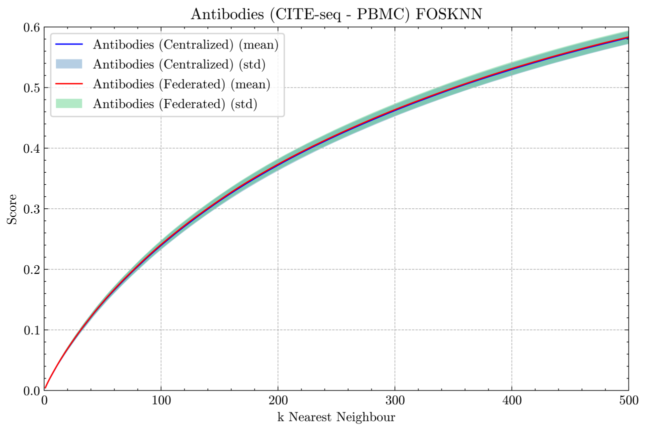

Figure 7: FOSKNN on Centralized versus Federated MaxFuse.

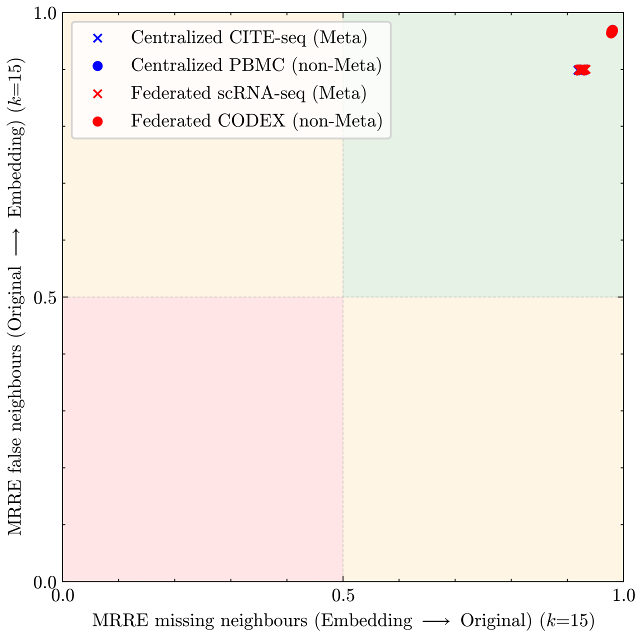

Figure 8: MRRE on Centralized versus Federated MaxFuse.

Furthermore, as illustrated in Figure 9, the analysis of Steadiness and Cohesiveness metrics (for Antibodies dataset) indicates that federated embeddings create almost the same quality of cluster structures as centralized ones. Both settings show tight clustering in the favorable upper-right region of the plots, with no significant differences in most statistical tests (see Table 6). These results provide strong evidence that federated MaxFuse can preserve cluster structural features during integration, an essential downstream analysis consideration for non-meta data modality as they are within the green quadrant, which signifies reliability of MaxFuse creating cluster structures within the embedding as in the original data (see Figure 3). However, performance is significantly worse for the meta-data modality, with very low cohesiveness. This suggests that MaxFuse introduced artificial cluster structures within the embedding that are not present in the original dataset. One possible reason is the summarization effect of the Leiden algorithm, which also explains the behavior observed in MRRE metric.

Additionally, we can visualize Steadiness and Cohesiveness using CheckViz and Reliability Maps. These provide further intuitive validation of the embedding quality findings as seen in Figure 11. Both centralized and federated settings display largely consistent structures, with reliable regions overlapping substantially. However, federated embeddings exhibit slightly larger distorted areas, which are visible through darker and expanded patches in the CheckViz overlays. This aligns with the slight deterioration observed in stress and MRRE metrics for the federated setting. While these distortions are minor for the Antibodies dataset, they become more prominent in the Tonsils dataset. This confirms that federated diagonal integration maintains competitive quality under simpler conditions but faces increased data heterogeneity and complexity challenges.

Lastly, some significant differences emerge when analyzing stress scores, a metric that captures the preservation of distances. While stress scores between centralized and federated embeddings are similar for the Antibodies dataset, federated embeddings exhibit significantly higher stress in the Tonsils dataset, most likely caused by the MaxFuse using data batching by default for the Tonsils dataset.

Feature	Dataset	Type	U1	U2	p-value	Reject H_0 (p = 0.05)
L1 Accuracy	Antibodies	Supervised	50198.0	12302.0	8.94e-32	True
L2 Accuracy	Antibodies	Supervised	50325.5	12174.5	3.52e-32	True
L3 Accuracy	Antibodies	Supervised	40879.0	21621.0	2.51e-09	True
Cluster-Term Accuracy	Tonsils	Supervised	56214.0	4795.0	4.57e-59	True
L1 Silhouette F1 Score	Antibodies	Supervised	35538.0	26962.0	7.95e-03	True
L2 Silhouette F1 Score	Antibodies	Supervised	30305.0	32195.0	5.59e-01	False
L3 Silhouette F1 Score	Antibodies	Supervised	29794.0	32706.0	3.68e-01	False
L1 ARI F1 Score	Antibodies	Supervised	35300.0	27200.0	1.22e-02	True
L2 ARI F1 Score	Antibodies	Supervised	28978.0	33522.0	1.60e-01	False
L3 ARI F1 Score	Antibodies	Supervised	30648.0	31852.0	7.10e-01	False
Cluster-Term Silhouette F1 Score	Tonsils	Supervised	2281.0	58728.0	8.34e-71	True
Cluster-Term ARI F1 Score	Tonsils	Supervised	8049.0	52960.0	1.74e-45	True
FOSCTTM	Antibodies	Supervised	32752.0	29748.0	3.53e-01	False
FOSKNN	Antibodies	Supervised	124288.0	125712.0	8.76e-01	False
Steadiness (15-NN, Non-Meta Modality)	Antibodies	Unsupervised	32259.0	30241.0	5.32e-01	False
Steadiness (15-NN, Meta Modality)	Antibodies	Unsupervised	32904.0	29596.0	3.06e-01	False
Cohesiveness (15-NN, Non-Meta Modality)	Antibodies	Unsupervised	32427.0	30073.0	4.66e-01	False
Cohesiveness (15-NN, Meta Modality)	Antibodies	Unsupervised	30949.0	31551.0	8.52e-01	False
MRRE (15-NN, Meta Modality, Missing)	Antibodies	Unsupervised	26708.0	35792.0	4.93e-03	True
MRRE (15-NN, Non-Meta Modality, Missing)	Antibodies	Unsupervised	33906.0	28594.0	1.00e-01	False
MRRE (15-NN, Meta Modality, False)	Antibodies	Unsupervised	28132.0	34368.0	5.36e-02	False
MRRE (15-NN, Non-Meta Modality, False)	Antibodies	Unsupervised	32365.0	30135.0	4.90e-01	False
Stress (Meta Modality)	Antibodies	Unsupervised	29290.0	33210.0	2.25e-01	False
Stress (Non-Meta Modality)	Antibodies	Unsupervised	26667.0	35833.0	4.56e-03	True
Stress (Meta Modality)	Tonsils	Unsupervised	13848.0	47161.0	8.70e-26	True
Stress (Non-Meta Modality)	Tonsils	Unsupervised	49799.0	11210.0	4.94e-34	True

Table 6: Statistical comparison of metrics for the Centralized versus Federated MaxFuse comparison with Hypothesis Testing through Mann–Whitney U test

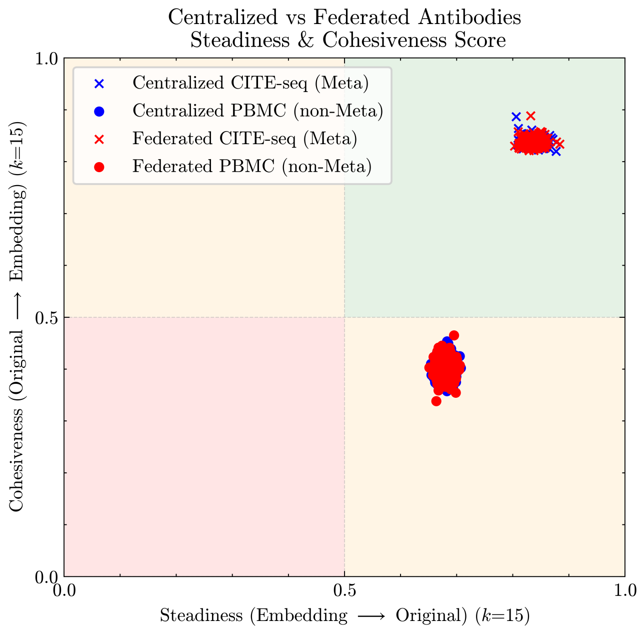

Figure 9: Steadiness and Cohesiveness on Centralized versus Federated MaxFuse.

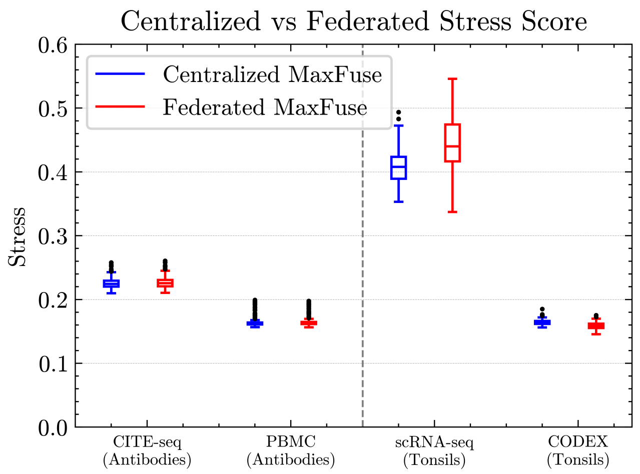

Figure 10: Stress on Centralized versus Federated MaxFuse.

4.2 Centralized Data Batching

We examine whether MaxFuse can be extended beyond two participants in a federated environment, where two participants utilize

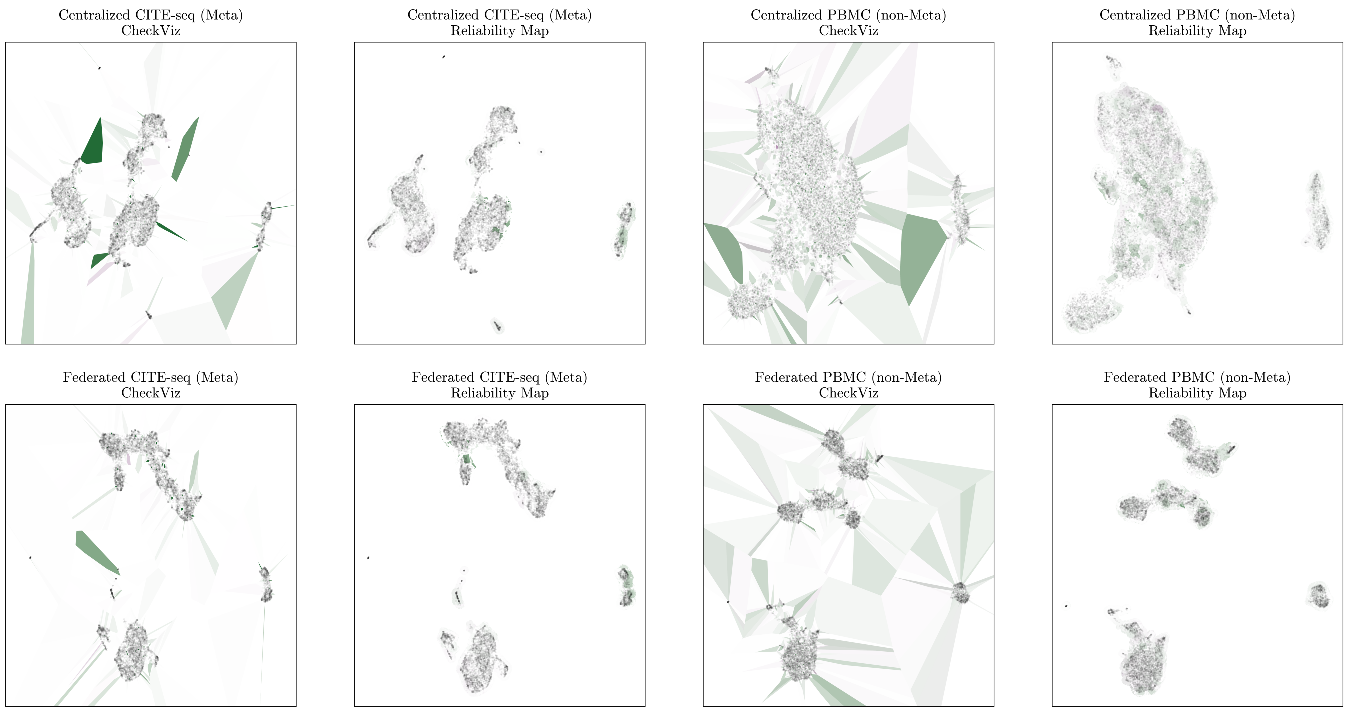

Figure 11: CheckViz and Reliability Map of the Steadiness and Cohesiveness for the Antibodies dataset

batching. Due to significant implementation challenges and complexity associated with federated batching support, we only examine centralized batching as a baseline for assessing its effects on integration quality with two clients.

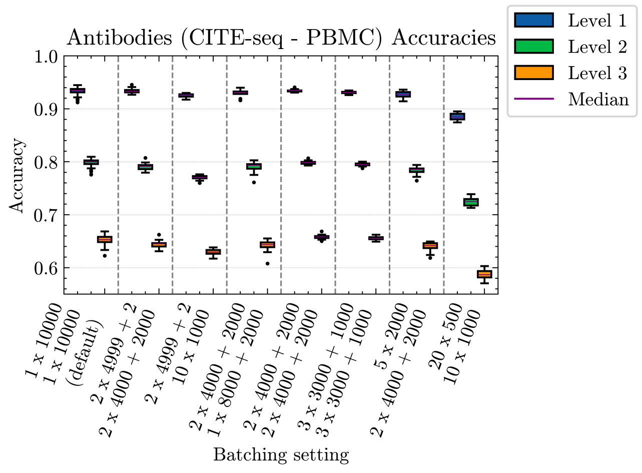

Figure 12: Summary of Accuracy per cell label on Batched MaxFuse for Antibodies Dataset

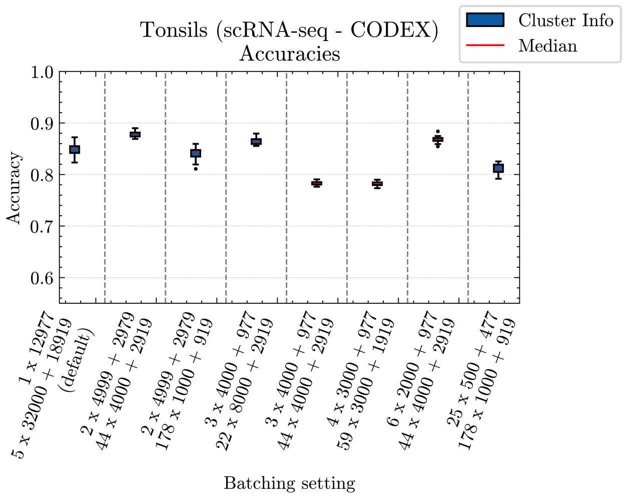

Figure 13: Summary of Accuracy per cell label on Batched MaxFuse for Tonsils Dataset

As shown in Figure 12 and Figure 13, batching only marginally impacts cell label prediction accuracy, provided that the number of batches remains moderate and sufficiently large batch sizes are maintained. For both the Antibodies and Tonsils datasets, excessive fragmentation (i.e., tiny batches) leads to noticeable drops in accuracy. Interestingly, the non-batched (default) setting generally performs among the best, indicating that while batching is feasible, there is no inherent advantage for simple supervised accuracy when datasets are manageable in size.

To complement these findings, we further examined the behavior of F1 scores, which capture a trade-off between batch mixing and cell-type clustering, as shown in Figure 14 and Figure 15.

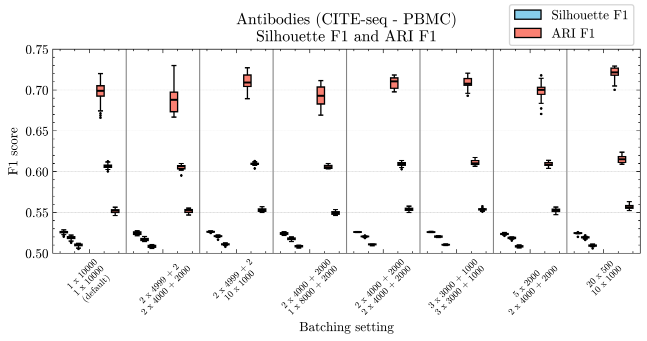

Figure 14: Summary of ASW F1 (Level 1,2,3) and ARI F1 (Level 1,2,3) scores on Batched MaxFuse for Antibodies Dataset

These results reveal that a moderate degree of batching can sometimes improve F1 scores, particularly when the batch sizes are balanced and not too small. However, aggressive batching, for instance, involving many small or large batches, significantly degrades the quality of clustering metrics. This suggests that minor batching, when carefully controlled, can facilitate better preservation of structure during integration without overly distorting batch correction or cell-type resolution.

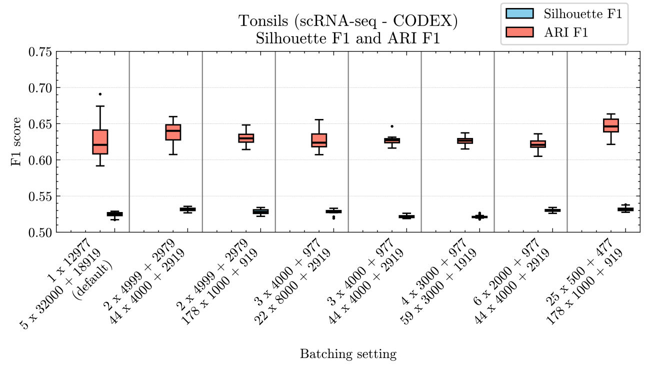

Figure 15: Summary of ASW F1 (Cluster Info/Term) and ARI F1 (Cluster Info/Term) score on Batched MaxFuse for Tonsils Dataset

We analyze the FOSKNN and FOSCTTM metrics to gain further insight into the preservation of embedding structure.

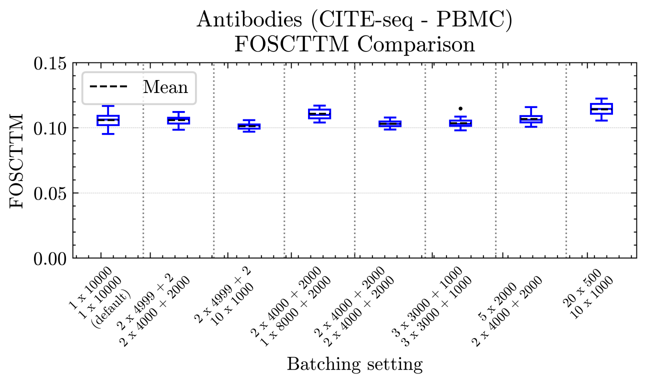

Figure 16: Centralized MaxFuse FOSKNN metric on Batched Antibodies Dataset

In Figure 16 and Figure 17, we observe that carefully applied 2-3 medium-sized batches can yield marginal improvements in local neighborhood preservation. Specifically, lower FOSCTTM values and slightly higher FOSKNN curves suggest that moderate batching enables better proximity matching, likely due to reduced cross-batch embedding distortions. However, these improvements quickly vanish if batch sizes become too small, emphasizing that while batching can be beneficial, excessive fragmentation must be avoided.

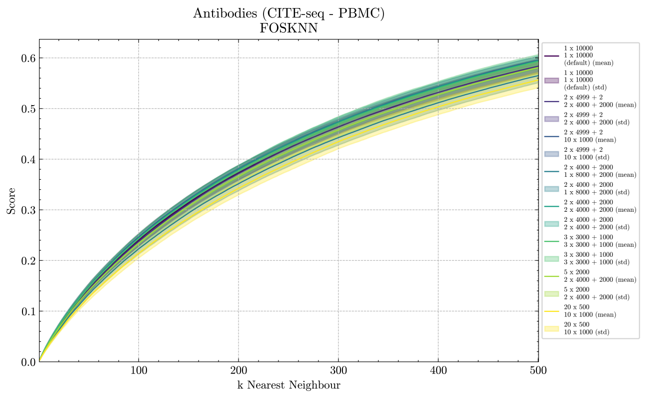

Figure 17: Centralized FOSKNN FOSKNN metric on Batched Antibodies Dataset

4.3 Federated Shared Feature Selection

MaxFuse relies on shared features to function correctly, but such correspondences might not always be available in advance. We explore how the number and quality of shared features affect MaxFuse's performance. Specifically, we test two conditions: (1) using only a random subset of shared features, and (2) selecting features randomly, to see whether non-corresponding features can be treated as shared and still lead to stable results. We vary the proportion of true shared features used (10%, 20%, 50%, 80%, and 100%), and the randomly selected variables (2, 5, 20, 50, 100, and 150) in their respective experiments. Since we alter only the shared-feature selection parameter, we focus on FOSCTTM and FOSKNN, as these metrics are particularly sensitive to embedding alignment quality and nearest-neighbor matching stability, and thus effectively reflect broader performance trends seen across various supervised and unsupervised metrics.

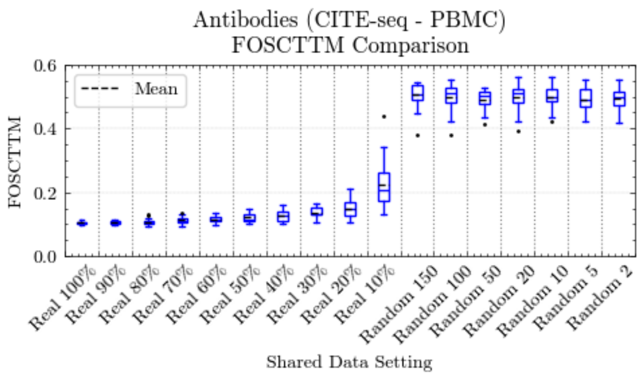

Figure 18: FOSCTTM on Federated Shared Feature Selection

Figure 18 and Figure 19 clearly illustrate a stark contrast between using actual shared and randomly chosen features. Specifically, random features severely impair the integration quality, indicated by significantly poorer FOSKNN and higher FOSCTTM scores. This decline in performance can be attributed to an inability to establish meaningful initial correspondences, leading to suboptimal canonical correlations and inaccurate embeddings. Conversely, performance improves when leveraging substantial proportions of about 60-80% of actual shared features, nearing optimal results obtained using the complete shared dataset. These findings underscore the necessity of careful and meaningful feature selection for achieving reliable federated diagonal integration, and that comparable results can still be achieved even if not all information feature correspondences are available.

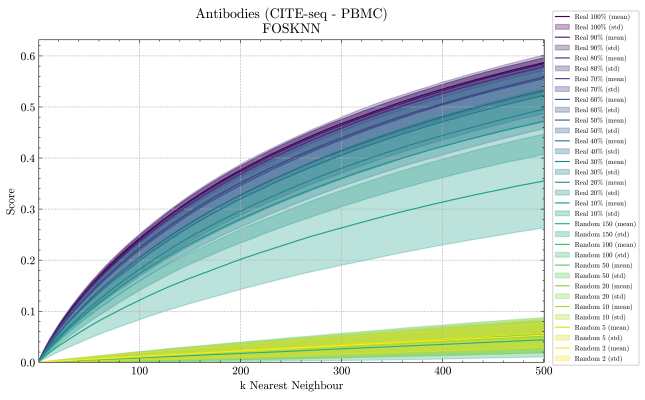

Figure 19: FOSKNN Federated Shared Feature Selection

4.4 Random Initialization in Federated CCA

As described in subsubsection 2.2.2, federated CCA, specifically the Singular Vector Power Method as delineated in Algorithm 8, requires the addition of noise as seen in Equation 8. To investigate the sensitivity of the method to random initialization, we systematically varied the degree of added randomness in the range $w \in [0.0, 1.0]$ (with a step size of 0.1) and assessed its impact on integration performance. It is essential to note that, as stated in the equation, 0.0 is undesirable, as it implies that a column of data must be shared during the initialization step.

Given that we are only perturbing the initialization step, we focused on evaluating two core metrics, FOSCTTM and FOSKNN, which effectively capture fine-grained matching fidelity and neighbor structure preservation, and are highly representative of broader trends across supervised and unsupervised metrics as mentioned in subsection 4.3.

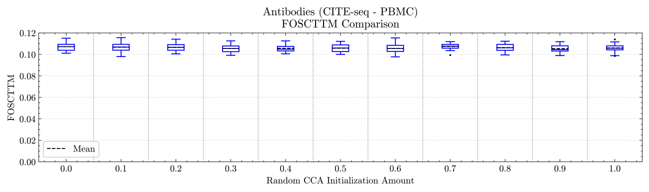

Figure 20: FOSCTTM Federated Random CCA Initialization Selection

The results in Figure 20 show that varying the level of random initialization has only a minor effect on integration quality, as reflected by relatively stable FOSCTTM scores. Interestingly, introducing approximately 60% randomness achieves the lowest median FOSCTTM value, suggesting slightly improved matching fidelity, where matched cell pairs remain closer than mismatched ones. This observation implies that moderate stochasticity may help avoid suboptimal convergence during the singular vector updates, although the overall effect size remains small.

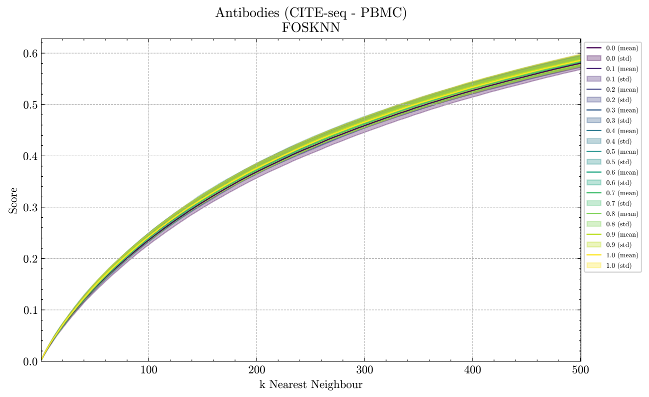

Figure 21: FOSKNN Federated Random CCA Initialization Selection

Similarly, Figure 21 shows that the FOSKNN scores remain stable across different randomness levels, with only minor variations. This further supports the conclusion that introducing reasonable amounts of random noise during initialization does not significantly impact neighbor preservation or overall integration quality. These findings validate the robustness of Federated CCA under various noise settings, providing flexibility in implementation for enhancing privacy or improving convergence stability.

5 DISCUSSION

This study investigates whether federated diagonal integration of two modalities, specifically through the Federated MaxFuse algorithm, can match the performance of centralized approaches while addressing pressing privacy concerns inherent in multi-omic data analysis. Integration enables the combined exploitation of individual dataset strengths, potentially facilitating a more comprehensive understanding of cellular interactions and immune responses within the tonsil microenvironment by subjecting it to downstream analysis or other algorithmic methods. Our experiments examined different dimensions of the problem, evaluating integration quality, matching accuracy, stability, and sensitivity to architectural and methodological choices. This section presents the key findings, limitations, and implications of each experimental setup and identifies future work directions.

5.1 Centralized versus Federated Comparison

Our analysis shows that, with respect to cell annotation accuracy and embedding quality, the federated version of MaxFuse performs as accurately as its centralized equivalent. Although the integration is generally effective, it struggles to delineate cluster boundaries and preserve fine-grained biological structures clearly. Future enhancements may investigate more complex optimization techniques, such as adaptive learning rates or the incorporation of additional regularization approaches, to further improve stability and accuracy.

Notably, non-meta datasets consistently outperform meta-aggregated counterparts. This difference likely arises due to the information loss inherent in meta summarization, specifically from using Leiden cluster centroids, which suggests a trade-off between computational efficiency and relative improvement in generalization. Nevertheless, MaxFuse effectively maintains strong local proximity preservation with minimal neighbor misalignments, accurately reflecting biological relationships within embeddings.

In general, the centralized approach yields slightly higher scores in metrics that measure cluster reliability and boundary clarity, specifically in terms of steadiness and cohesiveness. These results suggest that centralized integration provides slightly more robust embeddings, closely reflecting actual biological structures.

Despite these modest advantages of centralization, the federated version of MaxFuse demonstrates significant promise, particularly in scenarios with lower complexity and heterogeneity. Its performance gap in more intricate datasets underscores the necessity for targeted improvements, particularly in implementing federated batching, enhancing embedding stability, and strengthening local structure preservation.

Our findings highlight the substantial potential of federated diagonal integration methods. They outline specific areas for future research to bridge current performance gaps and expand the applicability of their approach across diverse biological datasets.

5.2 Centralized Data Batching

The exploration of centralized data batching within the MaxFuse algorithm highlights batching as a practical method for simulating federated environments involving multiple clients or datasets that share a common feature space. The findings suggest that, under controlled conditions, batching maintains high-level integration performance. Specifically, when batch sizes are moderately large and data fragmentation is minimized (small, low-volume batches), the overall matching accuracy and clustering quality are largely preserved. This suggests that federated scenarios, where clients may possess subsets of a shared feature space, can achieve reliable integration without substantial loss of alignment fidelity if carefully managed batching is employed.

Nevertheless, the findings also show that overly granular batching causes a noticeable decline in label accuracy and embedding quality, primarily when the data is split into multiple tiny batches (fragmented). The matching process may be compromised, and the resulting embeddings may become unstable due to the tendency of smaller batches to enhance noise and fragment local structures. This underscores the importance of optimizing batch size in any federated adaptation of MaxFuse or similar frameworks: batching should strike a balance between representing distributed data and preserving sufficient local structure for robust integration.

These results suggest that centralized batching can effectively mimic federated integration; however, careful calibration is necessary to achieve this. They encourage more research into multi-lab federated networks, especially for multi-modality applications like federated GCCA. Future research should investigate intrinsic data structure-based adaptive batching algorithms and address the real-world challenges of deploying batching in federated environments with stricter privacy and communication regulations. These problems must be resolved to scale federated diagonal integration to large multi-institutional partnerships.

5.3 Federated Shared Feature Selection

The shared feature selection experiments highlight the critical role of meaningful correspondences in achieving robust federated diagonal integration. When true shared features are available, even when only 60–80% of the original correspondences are retained, they are sufficient to maintain strong matching and embedding performance. However, incorporating features selected randomly results in a pronounced deterioration of integration quality, likely attributable to the absence of coherent alignment among modalities. These results indicate that federated integration is resilient to moderate reductions in shared information. However, it remains vulnerable to the quality and relevance of the features selected for integration. Consequently, the meticulous curation of shared features is paramount for dependable integration, particularly in contexts where prior knowledge regarding correspondences may be lacking.

5.4 Random Initialization in Federated CCA

The primary reason for introducing randomness is to maintain data privacy. Additionally, randomness can facilitate faster convergence or yield better canonical correlations, which are more suitable for matching. Maximizing the stochasticity of initialization may improve convergence properties and correlation outcomes. Our analysis shows that the integration process in Federated CCA is remarkably robust to variations in the degree of random initialization. Integration quality remains stable across various randomization levels, with only minor improvements observed at moderate noise levels. These improvements suggest that introducing stochasticity during the initialization phase can marginally enhance matching fidelity, likely by helping the algorithm escape poor local optima during singular vector updates. However, the aggregate effect size persists as minimal, indicating that Federated CCA remains robust even with adjustments to the randomness parameter. This adaptability enhances its pragmatic utility by reconciling the safeguarding of privacy with superior integration results.

5.5 Limitations

While this investigation reveals that Federated MaxFuse can achieve robust matching and embedding efficacy without centralized data access, several notable limitations remain. Primarily, the methodology presupposes the presence of honest-but-curious participants and fails to furnish formal privacy assurances against active adversaries. Secondly, the dependence on a centralized server architecture may introduce communication bottlenecks and algorithmic intricacies, particularly during federated CCA, which may hinder scalability to larger datasets or networks comprising numerous nodes. Thirdly, assessment in truly federated environments remains problematic, as most validation metrics necessitate centralized access to comprehensive embeddings or ground-truth correspondences. Fourthly, due to the substantial memory demands of implementing existing unsupervised metrics, we were unable to comprehensively evaluate performance on extensive datasets, mainly because of these substantial memory demands. Lastly, although this study focused on integrating two modalities, extending the methodology to encompass additional modalities or highly heterogeneous multi-omic datasets presents an unresolved and challenging avenue for future work.

Firstly, although federated learning inherently provides some level of data minimization by sending only partial, normalized data representations with optional noise interpolation, formal guarantees such as resilience to poisoning, robustness against network attacks, or robustness against message dropping have not been established. This was not the primary focus of this work, and our findings indicate that the federated setting can partially preserve privacy by design since raw data remains local without apparent degradation in matching and embedding quality. Additionally, we relied on the assumption that participating nodes and the server do not act maliciously. This simplifies the security landscape but limits real-world applicability, as adversarial behavior is possible and could attempt to reconstruct private data from aggregated model parameters. Therefore, it is recommended to explore the possibility of rebuilding it, if feasible, and to determine the extent of the reconstruction. Additionally, it would be beneficial to investigate alternative methods for communicating and securing the aggregation of calculations, which should be considered.

Secondly, the architecture of having a centralized server that handles all communication might not be the best, as it requires more message complexity than necessary and, in real-world settings, can cause latency and a single point of failure, where if the server goes down, all communication will be disrupted. These complexities especially add up when calculating CCA with the PLS model B, where there would be T loops for C shared components, where each node needs to find singular vectors K times in the worst case scenario. Therefore, exploring more efficient communication methods or alternative decentralized approaches that eliminate reliance on a central server to hold and manage data is essential to overcoming these bottlenecks.

Furthermore, evaluating federated MaxFuse performance introduces significant challenges, particularly due to limitations of existing supervised metrics, which typically require complete embedding data and explicit cell-level correspondences. In federated scenarios, these correspondences are often unavailable or impractical to establish, and transmitting full embeddings to a central node may compromise data privacy. Although benchmark datasets like CITE-seq and PBMC [26] can facilitate supervised evaluation when explicit correspondences exist, practical federated settings demand new evaluation strategies. Therefore, developing novel metrics or adapting existing methods capable of assessing alignment quality across silos—without requiring complete data sharing—is critical for accurately evaluating federated integration algorithms.

Additionally, a significant limitation of several unsupervised evaluation metrics, such as Steadiness, Cohesiveness, and Mean Relative Rank Errors, is their primary implementation in the ZADU package [37]. These metrics rely on calculations using full distance matrices, distance rank matrices, or sparse matrices that are dominated by zeros, all of which require significant memory allocation. In practice, we encountered substantial issues when working with datasets that exceeded 100,000 cells and used 32-bit floating-point numbers. These cases needed at least 37GB of memory, demonstrating limited scalability since memory demands increase by $O(n^2)$ with the dataset size. Although addressing these memory limitations would be valuable, it was beyond the scope of this paper, since metric optimization was beyond the scope of this study. Nevertheless, we recommend future work on developing more memory-efficient implementations of these metrics.

Lastly, our study experiments focused on two-modality integration, which we considered a case where batching is not used within the calculations. However, there could be more complex cases involving over 500,000 cells or even many cells with the same modality spread across multiple devices. This data size problem would severely degrade the performance, and as the centralized batching experiments showed, they can help improve specific metrics. However, the current batching method should be evaluated to determine the best approach. Additionally, we must consider how to scale to three or more modalities in a federated fashion, which remains an unresolved challenge, particularly in achieving both consistency and scalability. Tri-omic analysis has been performed in a centralized manner by the authors of MaxFuse [12], who applied a 'merged' Generalised Canonical Correlation Analysis (GCCA) to fitted and filtered pivots. It remains to be analyzed whether this approach is the best. Additionally, it would be necessary to explore whether complex implementation variants of distributed CCA/GCCA, such as communication-efficient [71, 22] or MAX-VAR [32, 23], are suitable. Additionally, it might be valuable to look into different ways of calculating CCA, whether that would be through kernels, probabilistic models, or even deep models [11].

6 CONCLUSION

This empirical study demonstrates that applying MaxFuse to two-modal data integration in a federated setting provides a promising foundation for extending the method to multi-modal, decentralized applications. Despite practical challenges and inherent limitations, the findings establish a groundwork for scalable, flexible, and privacy-conscious cross-modality correspondence in federated environments. Future research should focus on enhancing formal privacy guarantees, improving communication efficiency, optimizing batching strategies, and broadening support for more diverse and complex modality configurations.

REFERENCES

[1] Amir "Jaberzadeh et al. “"Blockchain-Based Federated Learning: Incentivizing Data Sharing and Penalizing Dishonest Behavior"”. In: *"Blockchain and Applications, 5th International Congress"*. Ed. by José Manuel "Machado et al. "Cham": "Springer Nature Switzerland", 2023, "186–195".

[2] Nigatu Adossa et al. “Computational strategies for single-cell multi-omics integration”. In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 2588–2596. ISSN: 20010370. DOI: 10.1016/j.csbj.2021.04.060. URL: https://doi.org/10.1016/j.csbj.2021.04.060.

[3] Ricard Argelaguet et al. “Computational principles and challenges in single-cell data integration”. In: *Nature Biotechnology* 39.10 (2021), pp. 1202–1215. ISSN: 15461696. DOI: 10.1038/s41587-021-00895-7. URL: http://dx.doi.org/10.1038/s41587-021-00895-7.

[4] Tasbiraha Athaya et al. “Multimodal deep learning approaches for single-cell multi-omics data integration”. In: *Briefings in Bioinformatics* 24.5 (2023), pp. 1–15. ISSN: 14774054. DOI: 10.1093/bib/bbad313.

[5] Franziska Boenisch et al. “When the Curious Abandon Honesty: Federated Learning Is Not Private”. In: *Proceedings - 8th IEEE European Symposium on Security and Privacy, Euro S and P 2023* (2023), pp. 175–199. DOI: 10.1109/EuroSP57164.2023.00020.

[6] Bram Burger, Marc Vaudel, and Harald Barsnes. “Automated splitting into batches for observational biomedical studies with sequential processing”. In: *Biostatistics* 24.4 (2023), pp. 1031–1044. ISSN: 14684357. DOI: 10.1093/biostatistics/kxac014.

[7] Rainer Burkard, Mauro Dell’Amico, and Silvano Martello. *Assignment Problems. Revised reprint.* English. 393 Seiten. SIAM - Society of Industrial and Applied Mathematics, 2012. ISBN: 978-1-611972-22-1.

[8] Ricardo J.G.B. Campello, Davoud Moulavi, and Joerg Sander. “Density-based clustering based on hierarchical density estimates”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7819 LNAI.PART 2 (2013), pp. 160–172. ISSN: 03029743. DOI: 10.1007/978-3-642-37456-2{_}14.

[9] Kai Cao et al. “A unified computational framework for single-cell data integration with optimal transport”. In: *Nature Communications* 13.1 (2022), pp. 1–15. ISSN: 20411723. DOI: 10.1038/s41467-022-35094-8.

[10] Tien Dung Cao et al. “A federated deep learning framework for privacy preservation and communication efficiency”. In: *Journal of Systems Architecture* 124.January (2022), p. 102413. ISSN: 13837621. DOI: 10.1016/j.sysarc.2022.102413. URL: https://doi.org/10.1016/j.sysarc.2022.102413.

[11] James Chapman and Hao-Ting Wang. “CCA-Zoo: A collection of Regularized, Deep Learning based, Kernel, and Probabilistic CCA methods in a scikit-learn style framework”. In: *Journal of Open Source Software* 6.68 (2021), p. 3823. DOI: 10.21105/joss.03823.

[12] Shuxiao Chen et al. *Integration of spatial and single-cell data across modalities with weakly linked features*. Springer US, 2023. DOI: 10.1038/s41587-023-01935-0.

[13] Neo Christopher Chung. “Statistical significance of cluster membership for unsupervised evaluation of cell identities”. In: *Bioinformatics* 36.10 (2020), pp. 3107–3114. ISSN: 14602059. DOI: 10.1093/bioinformatics/btaa087.

[14] A.G. Cioletti et al. “Deep Learning in Bioinformatics”. In: *Advances in bioinformatics and biomedical engineering book series* (2024), pp. 137–168. DOI: 10.4018/979-8-3693-3192-7.ch005.

[15] David F. Crouse. “On implementing 2D rectangular assignment algorithms”. In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696. ISSN: 00189251. DOI: 10.1109/TAES.2016.140952.

[16] Varsha Dani et al. “Secure multi-party computation in large networks”. In: *Distributed Computing* 30.3 (2017), pp. 193–229. ISSN: 01782770. DOI: 10.1007/s00446-016-0284-9.

[17] Ozgur Demir-Kavuk et al. “Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features”. In: *BMC Bioinformatics* 12 (2011), pp. 1–10. ISSN: 14712105. DOI: 10.1186/1471-2105-12-412.

[18] Lei Ding, Gabriel E. Zentner, and Daniel J. McDonald. “Sufficient principal component regression for pattern discovery in transcriptomic data”. In: *Bioinformatics Advances* 2.1 (2022), pp. 1–8. ISSN: 26350041. DOI: 10.1093/bioadv/vbac033.

[19] Arnold Dresden. “The fourteenth western meeting of the american mathematical society”. In: *Bulletin of the American Mathematical Society* 26.9 (1920), pp. 385–396. ISSN: 02730979. DOI: 10.1090/S0002-9904-1920-03322-7.

[20] Paolo Fardin et al. “The l1-l2regularization framework unmasks the hypoxia signature hidden in the transcriptome of a set of heterogeneous neuroblastoma cell lines”. In: *BMC Genomics* 10 (2009), p. 474. ISSN: 14712164. DOI: 10.1186/1471-2164-10-474.

[21] Mattia Forcato, Oriana Romano, and Silvio Bicciato. “Computational methods for the integrative analysis of single-cell data”. In: *Briefings in Bioinformatics* 22.1 (2021), pp. 20–29. ISSN: 14774054. DOI: 10.1093/bib/bbaa042.

[22] Xiao Fu et al. “Efficient and Distributed Generalized Canonical Correlation Analysis for Big Multiview Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 31.12 (2019), pp. 2304–2318. ISSN: 15582191. DOI: 10.1109/TKDE.2018.2875908.

[23] Xiao Fu et al. “Scalable and Flexible Multiview MAX-VAR Canonical Correlation Analysis”. In: *IEEE Transactions on Signal Processing* 65.16 (2017), pp. 4150–4165. ISSN: 1053587X. DOI: 10.1109/TSP.2017.2698365.

[24] Paul Geladi. “Notes on the history and nature of partial least squares (PLS) modelling”. In: *Journal of Chemometrics* 2.4

(1988), pp. 231–246. ISSN: 0886-9383. DOI: 10.1002/cem.1180020403.

[25] Yury Goltsev et al. “Deep Profiling of Mouse Splenic Architecture with CODEX Multiplexed Imaging”. In: *Cell* 174.4 (2018), pp. 968–981. ISSN: 10974172. DOI: 10.1016/j.cell.2018.07.010. URL: https://doi.org/10.1016/j.cell.2018.07.010.

[26] Yuhan Hao et al. “Integrated analysis of multimodal single-cell data”. In: *Cell* 184.13 (2021), pp. 3573–3587. ISSN: 10974172. DOI: 10.1016/j.cell.2021.04.048. URL: https://doi.org/10.1016/j.cell.2021.04.048.

[27] Markus Helmer et al. “On the stability of canonical correlation analysis and partial least squares with application to brain-behavior associations”. In: *Communications Biology* (2024). ISSN: 2399-3642. DOI: 10.1038/s42003-024-05869-4. URL: http://dx.doi.org/10.1038/s42003-024-05869-4.

[28] Jörg Henseler et al. “Common Beliefs and Reality About PLS: Comments on Rönkkö and Evermann (2013)”. In: *Organizational Research Methods* 17.2 (2014), pp. 182–209. ISSN: 15527425. DOI: 10.1177/1094428114526928.

[29] Lukas Heumos et al. “Best practices for single-cell analysis across modalities”. In: *Nature Reviews Genetics* 24.8 (2023), pp. 550–572. ISSN: 14710064. DOI: 10.1038/s41576-023-00586-w.

[30] John W. Hickey et al. “Organization of the human intestine at single-cell resolution”. In: *Nature* 619.7970 (2023), pp. 572–584. ISSN: 14764687. DOI: 10.1038/s41586-023-05915-x.

[31] Harold Hotelling. “Relations Between Two Sets of Variates”. In: *Biometrika* 28.3/4 (1936), p. 321. ISSN: 00063444. DOI: 10.2307/2333955.

[32] Charles Hovine and Alexander Bertrand. “Distributed MAX-VAR: Identifying Common Signal Components across the Nodes of a Sensor Network”. In: *European Signal Processing Conference* 2021-Augus (2021), pp. 2159–2163. ISSN: 22195491. DOI: 10.23919/EUSIPCO54536.2021.9615952.

[33] Lauren L. Hsu and Aedin C. Culhane. “Impact of Data Preprocessing on Integrative Matrix Factorization of Single Cell Data”. In: *Frontiers in Oncology* 10.June (2020). ISSN: 2234943X. DOI: 10.3389/fonc.2020.00973.

[34] Christopher A. Jackson and Christine Vogel. “New horizons in the stormy sea of multimodal single-cell data integration”. In: *Molecular Cell* 82.2 (2022), pp. 248–259. ISSN: 10974164. DOI: 10.1016/j.molcel.2021.12.012. URL: https://doi.org/10.1016/j.molcel.2021.12.012.

[35] Derry Jatnika, Moch Arif Bijaksana, and Arie Ardiyanti Suryani. “Word2Vec Model Analysis for Semantic Similarities in English Words”. In: *Procedia Computer Science* 157 (2019). The 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society, pp. 160–167. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.08.153. URL: https://www.sciencedirect.com/science/article/pii/S1877050919310713.

[36] Hyeon Jeon et al. “Measuring and Explaining the Inter-Cluster Reliability of Multidimensional Projections”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), pp. 551–561. ISSN: 19410506. DOI: 10.1109/TVCG.2021.3114833.

[37] Hyeon Jeon et al. “ZADU: A Python Library for Evaluating the Reliability of Dimensionality Reduction Embeddings”. In: *Proceedings - 2023 IEEE Visualization Conference - Short Papers, VIS 2023* (2023), pp. 196–200. DOI: 10.1109/VIS54172.2023.00048.

[38] Iain M. Johnstone and D. Michael Titterington. “Statistical challenges of high-dimensional data”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1906 (2009), pp. 4237–4253. ISSN: 1364503X. DOI: 10.1098/rsta.2009.0159.

[39] Adam Jóźwik. “A learning scheme for a fuzzy k-NN rule”. In: *Pattern Recognition Letters* 1.5-6 (1983), pp. 287–289. ISSN: 01678655. DOI: 10.1016/0167-8655(83)90064-8.

[40] James M. Keller and Michael R. Gray. “A Fuzzy K-Nearest Neighbor Algorithm”. In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-15.4 (1985), pp. 580–585. ISSN: 21682909. DOI: 10.1109/TSMC.1985.6313426.

[41] Julia Kennedy-Darling et al. “Highly multiplexed tissue imaging using repeated oligonucleotide exchange reaction”. In: *European Journal of Immunology* 51.5 (2021), pp. 1262–1277. ISSN: 15214141. DOI: 10.1002/eji.202048891.

[42] Hamish W. King et al. “Integrated single-cell transcriptomics and epigenomics reveals strong germinal center–associated etiology of autoimmune risk loci”. In: *Science Immunology* 6.64 (2021). ISSN: 24709468. DOI: 10.1126/sciimmunol.abh3768.

[43] Jip W.T.M. de Kok et al. “A guide to sharing open healthcare data under the General Data Protection Regulation”. In: *Scientific Data* 10.1 (2023), pp. 1–10. ISSN: 20524463. DOI: 10.1038/s41597-023-02256-2.

[44] Ilya Korsunsky et al. “Fast, sensitive and accurate integration of single-cell data with Harmony”. In: *Nature Methods* 16.12 (2019), pp. 1289–1296. ISSN: 15487105. DOI: 10.1038/s41592-019-0619-0. URL: http://dx.doi.org/10.1038/s41592-019-0619-0.

[45] J. B. Kruskal. “Nonmetric multidimensional scaling: A numerical method”. In: *Psychometrika* 29 (1964), pp. 115–129. ISSN: 00333123. DOI: 10.1007/BF02289694.

[46] J. B. KRUSKAL and BELL. “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”. In: *BELL TELEPHONE LABORATORIES MURRAY HILL, N. J.* 29 (1964), pp. 1–27. ISSN: 26515032. DOI: 10.5137/1019-5149.JTN.14036-15.1.

[47] Tomasz Kujawa, Michał Marczyk, and Joanna Polanska. “Influence of single-cell RNA sequencing data integration on the performance of differential gene expression analysis”. In: *Frontiers in Genetics* 13.November (2022), pp. 1–13. ISSN: 16648021. DOI: 10.3389/fgene.2022.1009316.

[48] John A. Lee and Michel Verleysen. “Quality assessment of dimensionality reduction: Rank-based criteria”. In: *Neurocomputing* 72.7-9 (2009), pp. 1431–1443. ISSN: 09252312. DOI: 10.1016/j.neucom.2008.12.017.

[49] Sylvain Lespinats and Michaël Aupetit. “CheckViz: Sanity check and topological clues for linear and non-linear mappings”. In: *Computer Graphics Forum* 30.1 (2011), pp. 113–125. ISSN: 14678659. DOI: 10.1111/j.1467-8659.2010.01835.x.

[50] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60. ISSN: 15580792. DOI: 10.1109/MSP.2020.2975749.

[51] Wenqi Li et al. "Privacy-preserving Federated Brain Tumour Segmentation". In: *Machine Learning in Medical Imaging*. Vol. 1. October. Shenzhen, China: Springer US, 2019, pp. 673–680. ISBN: 978-3-030-32691-3. DOI: 10.1007/978-3-030-32692-0{_}16. URL: https://doi.org/10.1007/978-3-030-32692-0_16.

[52] Jie Liu et al. "Jointly embedding multiple single-cell omics measurements". In: *Leibniz International Proceedings in Informatics, LIPIcs* 143.10 (2019), pp. 1–10. ISSN: 18688969. DOI: 10.4230/LIPIcs.WABI.2019.10.

[53] Malte D. Luecken et al. "Benchmarking atlas-level data integration in single-cell genomics". In: *Nature Methods* 19.1 (2022), pp. 41–50. ISSN: 15487105. DOI: 10.1038/s41592-021-01336-8.

[54] Emma Lundberg and Georg H.H. Borner. "Spatial proteomics: a powerful discovery tool for cell biology". In: *Nature Reviews Molecular Cell Biology* 20.5 (2019), pp. 285–302. ISSN: 14710080. DOI: 10.1038/s41580-018-0094-y. URL: http://dx.doi.org/10.1038/s41580-018-0094-y.

[55] MacQueen, James and others. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* 1.14 (1967), pp. 281–297. URL: http://books.google.de/books?hl=de&lr=&id=IC4Ku_7dBFUC&oi=fnd&pg=PA281&dq=MacQueen+some+methods+for+classification&ots=nNTcK1IdoQ&sig=fHzdVcbvmYJ-lTNHu1HncmOFOkM#v=onepage&q=MacQueen%20some%20methods%20for%20classification&f=false.

[56] Julian Matschinske et al. "The FeatureCloud Platform for Federated Learning in Biomedicine: Unified Approach". In: *Journal of Medical Internet Research* 25 (2023). ISSN: 14388871. DOI: 10.2196/42621.

[57] Agoston Mihalik et al. "Canonical Correlation Analysis and Partial Least Squares for Identifying Brain – Behavior Associations : A Tutorial and a Comparative Study". In: *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging* 7.11 (2022), pp. 1055–1067. ISSN: 2451-9022. DOI: 10.1016/j.bpsc.2022.07.012. URL: https://doi.org/10.1016/j.bpsc.2022.07.012.

[58] Hai C.T. Nguyen et al. "Benchmarking integration of single-cell differential expression". In: *Nature Communications* 14.1 (2023). ISSN: 20411723. DOI: 10.1038/s41467-023-37126-3.

[59] Giuseppe Palermo, Paolo Piraino, and Hans Dieter Zucht. "Performance of PLS regression coefficients in selecting variables for each response of a multivariate PLS for omics-type data". In: *Advances and Applications in Bioinformatics and Chemistry* 2.1 (2009), pp. 57–70. ISSN: 11786949. DOI: 10.2147/aabc.s3619.

[60] Fernando V. Paulovich et al. "Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping". In: *IEEE Transactions on Visualization and Computer Graphics* 14.3 (2008), pp. 564–575. ISSN: 10772626. DOI: 10.1109/TVCG.2007.70443.

[61] Liangrui Ren et al. "Single-cell RNA-seq data clustering by deep information fusion". In: *Briefings in functional genomics* 23 (May 2023), p. 1570. DOI: 10.1093/bfgp/elad017.

[62] Nicola Rieke et al. "The future of digital health with federated learning". In: *npj Digital Medicine* 3.1 (2020), pp. 1–7. ISSN: 23986352. DOI: 10.1038/s41746-020-00323-1. URL: http://dx.doi.org/10.1038/s41746-020-00323-1.

[63] Luc Rocher, Julien M. Hendrickx, and Yves Alexandre de Montjoye. "Estimating the success of re-identifications in incomplete datasets using generative models". In: *Nature Communications* 10.1 (2019). ISSN: 20411723. DOI: 10.1038/s41467-019-10933-3. URL: http://dx.doi.org/10.1038/s41467-019-10933-3.

[64] Yeonjae Ryu et al. "Integration of Single-Cell RNA-Seq Datasets: A Review of Computational Methods". In: *Molecules and Cells* 46.2 (2023), pp. 106–119. ISSN: 02191032. DOI: 10.14348/molcells.2023.0009.

[65] Rahul Satija et al. "Spatial reconstruction of single-cell gene expression data". In: *Nature Biotechnology* 33.5 (2015), pp. 495–502. ISSN: 15461696. DOI: 10.1038/nbt.3192.

[66] Eric E. Schadt et al. "Genetics of gene expression surveyed in maize, mouse and man". In: *Nature* 422.6929 (2003), pp. 297–302. ISSN: 00280836. DOI: 10.1038/nature01434.

[67] Bruce Schmeiser. "Batch Size Effects in the Analysis of Simulation Output". In: *INFORMS* 30.3 (1982), pp. 556–568.

[68] Jennifer A. Scott. "An Arnoldi Code for Computing Selected Eigenvalues of Sparse, Real, Unsymmetric Matrices". In: *ACM Transactions on Mathematical Software (TOMS)* 21.4 (1995), pp. 432–475. ISSN: 15577295. DOI: 10.1145/212066.212091.

[69] Angela Serra et al. "Data integration in genomics and systems biology". In: *2016 IEEE Congress on Evolutionary Computation, CEC 2016* (2016), pp. 1272–1279. DOI: 10.1109/CEC.2016.7743934.

[70] Wharton Research Data Services. *Wharton Research Data Services — wrds-www.wharton.upenn.edu*. https://wrds-www.wharton.upenn.edu/. [Accessed 17-09-2024]. 1993.

[71] Sagar Shrestha and Xiao Fu. "Communication-Efficient Federated Linear and Deep Generalized Canonical Correlation Analysis". In: *IEEE Transactions on Signal Processing* 71 (2023), pp. 1379–1394. ISSN: 19410476. DOI: 10.1109/TSP.2023.3265886.

[72] Djura Smits et al. "An Improved Infrastructure for Privacy-Preserving Analysis of Patient Data". In: *Studies in Health Technology and Informatics* 295 (2022), pp. 144–147. ISSN: 18798365. DOI: 10.3233/SHTI220682.

[73] Tim Stuart et al. "Comprehensive Integration of Single-Cell Data". In: *Cell* 177.7 (2019), pp. 1888–1902. ISSN: 10974172. DOI: 10.1016/j.cell.2019.05.031.

[74] Hoa Thi et al. "A benchmark of batch-effect correction methods for single-cell RNA sequencing data". In: *Genome Biology* (2020), pp. 1–32.

[75] V. A. Traag, L. Waltman, and N. J. van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific Reports* 9.1 (2019), pp. 1–12. ISSN: 20452322. DOI: 10.1038/s41598-019-41695-z.

[76] Chenfei Wang et al. "Integrative analyses of single-cell transcriptome and regulome using MAESTRO". In: *Genome Biology* 21.1 (2020), pp. 1–28. ISSN: 1474760X. DOI: 10.1186/s13059-020-02116-x.

[77] Xuesong Wang et al. "Con-AAE: contrastive cycle adversarial autoencoders for single-cell multi-omics alignment and integration". In: *Bioinformatics* 39.4 (2023), pp. 1–7. ISSN: 13674811. DOI: 10.1093/bioinformatics/btad162.

[78] J.A. Wegelin. *A survey of Partial Least Squares (PLS) methods, with emphasis on the two-block case*. Seattle, 2000. URL: https://stat.uw.edu/sites/default/files/files/reports/2000/tr371.pdf.

[79] Joshua D. Welch, Alexander J. Hartemink, and Jan F. Prins. "MATCHER: Manifold alignment reveals correspondence between single cell transcriptome and epigenome dynamics". In: *Genome Biology* 18.1 (2017), pp. 1–19. ISSN: 1474760X. DOI: 10.1186/s13059-017-1269-0.

[80] Joshua D. Welch et al. "Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity". In: *Cell* 177.7 (2019), pp. 1873–1887. ISSN: 10974172. DOI: 10.1016/j.cell.2019.05.006.

[81] HERMAN WOLD. "11 - Path Models with Latent Variables: The NIPALS Approach**NIPALS = Nonlinear Iterative PArtial Least Squares." In: *Quantitative Sociology*. Ed. by H.M. Blalock et al. International Perspectives on Mathematical and Statistical Modeling. Academic Press, 1975, pp. 307–357. ISBN: 978-0-12-103950-9. DOI: https://doi.org/10.1016/B978-0-12-103950-9.50017-4. URL: https://www.sciencedirect.com/science/article/pii/B9780121039509500174.

[82] Svante Wold and Michael Sjostrom. "PLS-regression : a basic tool of chemometrics". In: (2001), pp. 109–130.

[83] Lang Wu et al. "A Survey on Blockchain-Based Federated Learning". In: *Future Internet* 15.12 (2023), pp. 1–22. ISSN: 19995903. DOI: 10.3390/fi15120400.

[84] Mengyun Wu, Huangdi Yi, and Shuangge Ma. "Vertical integration methods for gene expression data analysis". In: *Briefings in Bioinformatics* 22.3 (2021), pp. 1–14. ISSN: 14774054. DOI: 10.1093/bib/bbaa169.

[85] Yang Xu and Rachel Patton McCord. "Diagonal integration of multimodal single-cell data: potential pitfalls and paths forward". In: *Nature Communications* 13.1 (2022), pp. 1–4. ISSN: 20411723. DOI: 10.1038/s41467-022-31104-x.

[86] Qiang Yang et al. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (2019), pp. 1–19. ISSN: 21576912. DOI: 10.1145/3298981.

[87] Zi Yang and George Michailidis. "A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data". In: *Bioinformatics* 32.1 (2016), pp. 1–8. ISSN: 14602059. DOI: 10.1093/bioinformatics/btv544.

[88] Chun-Han Yao et al. "Federated Multi-Target Domain Adaptation". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2022, pp. 1081–1090. ISBN: 978-1-6654-0915-5. DOI: 10.1109/WACV51458.2022.00115. URL: https://ieeexplore.ieee.org/document/9706703/.

[89] Ziqi Zhang et al. "scMoMaT jointly performs single cell mosaic integration and multi-modal bio-marker detection". In: *Nature Communications* 14.1 (2023). ISSN: 20411723. DOI: 10.1038/s41467-023-36066-2.

[90] Grace X.Y. Zheng et al. "Massively parallel digital transcriptional profiling of single cells". In: *Nature Communications* 8 (2017). ISSN: 20411723. DOI: 10.1038/ncomms14049.

ACRONYMS

ARI Adjusted Rand Index 13
ARI F1 Adjusted Rand Index F1 12, 13, 17, 18, 22
ASW Average Silhouette Width 13
ASW F1 Average Silhouette Width F1 12, 13, 17, 18, 22

CCA Canonical Correlation Analysis 3, 4, 6, 7, 9, 10, 11, 13, 15, 16, 17, 23, 24, 25, 30
CITE-seq Cellular Indexing of Transcriptomes and Epitopes by Sequencing 5, 6, 17, 18, 25
CODEX CO-Detection by indEXing data 3, 5, 6, 17, 18

DAIC Delft AI Cluster 15, 30
DNN Deep Neural Network 3

FL Federated Learning 3, 4, 11, 13
FOSCTTM Fraction of Samples Closer Than True Match 12, 17, 18, 22, 23
FOSKNN Fraction Of Samples whose true matches are among their K-Nearest Neighbors 12, 17, 18, 19, 22, 23

GCCA Generalised Canonical Correlation Analysis 16, 24, 25
GDPR General Data Protection Regulation 4

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise 14
HIPAA Health Insurance Portability and Accountability Act 4
HVGs Highly Variable Genes 5, 6

iNMF integrative Non-negative Matrix Factorization 4

L1 Least absolute shrinkage and selection operator 3
L2 Ridge regression 3
LIGER Linked Inference of Genomic Experimental Relationships 4, 5

MATCHER **M**anifold **A**lignment to **CH**aracterize **E**xperimental **R**elationships 4
MaxFuse **Ma**tching **x**cross modalities via **Fu**zzy **s**moothed **e**mbeddings 3, 4, 5, 6, 8, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 30, 31
MPC Multi-Party Computation 4
MRRE Mean Relative Rank Errors 13, 14, 17, 18, 19

NN Nearest Neighbour 4, 5, 6, 7, 8, 9, 12, 14, 15, 19

PBMC Peripheral Blood Mononuclear Cell 5, 6, 17, 18, 25
PCA Principal Component Analysis 3
PII Personally Identifiable Information 4
PLS Partial Least Squares 10, 11, 25

scATAC-seq single-cell Assay for Transposase-Accessible Chromatin sequencing 3
scRNA-seq single-cell RNA sequencing 3, 5, 6, 17, 18
SVD Singular Value Decomposition 4, 7, 16
SVM Support Vector Machines 3

UUID Universal Unique Identifier 30

WRDS Wharton Research Data Services 5

A TECHNICAL IMPLEMENTATION AND REPRODUCIBILITY

All experiments were executed on the Delft AI Cluster (DAIC), where each was submitted as a job through the SLURM queuing system. Each job was initialized from a JSON configuration file, which included paths to data and labels, training hyperparameters, and a specification of the experiment type (e.g., standard vs. federated MaxFuse, CCA randomization) as seen in the example in Appendix C. These per-run configuration files were derived from a master JSON file containing parameter lists, from which combinations were generated automatically.

All software dependencies and binaries were containerized to ensure reproducibility and consistent environments across machines. Docker images were built locally and pushed to the GitHub Container Registry, then pulled on DAIC and executed with Apptainer (formerly Singularity), as root access is restricted on the cluster.

Each run was tagged with a unique identifier composed of a UUID and a SLURM-generated ID. Output from each run included:

- **Embedding files**: *'meta_embedding.npy'* and *'non_meta_embedding.npy'*
- **Matching results**: *'matching.csv'*
- **Evaluation outputs**: *'supervised_evaluation_metrics.json'*, *'unsupervised_evaluation_metrics.json'* and *'local_evaluation_metrics.json'*

As the names suggest, 'supervised_evaluation_metrics.json' contains supervised and 'unsupervised_evaluation_metrics.json' unsupervised scores, while 'local_evaluation_metrics.json' provides localized unsupervised metrics, useful for distortion visualization via CheckViz and Reliability Map. The directory structure of outputs is shown in Figure 22. This fine-grained implementation was used to ensure that the job would complete and save the result to disk in the event of a failure.

```
out
└── Centralized_vs_federated
│   └── regular_cite_seq_pbmc_params_{ID}
│   │   └── supervised_evaluation_metrics.json
│   │   └── unsupervised_evaluation_metrics.json
│   │   └── local_evaluation_metrics.json
│   │   └── meta_embedding.npy
│   │   └── non_meta_embedding.npy
│   │   └── matching.csv
│   └── regular_tonsils_params_{ID}
│   └── federated_cite_seq_pbmc_params_{ID}
│   └── federated_tonsils_params_{ID}
└── shared_params
│   └── federated_cite_seq_pbmc_params_{ID}
│   └── federated_tonsils_params_{ID}
└── cca
│   └── federated_cite_seq_pbmc_params_{ID}
│   └── federated_tonsils_params_{ID}
└── batching
    └── regular_cite_seq_pbmc_params_{ID}
    └── regular_tonsils_params_{ID}
```

Figure 22: Directory structure of the experimental output

B TRUNCATED SVD

Truncated SVD (leveraging Arnoldi/Lanczos methods) can be utilized either for dataset denoising, computing an approximation $\tilde{X} = U_k \cdot \mathrm{diag}(\Sigma_k) \cdot V_k^\top$, where $X \approx \tilde{X}$, or for dimensionality reduction by retaining only the k most significant singular values and vectors, represented by $U_k \cdot \mathrm{diag}(\Sigma_k)$. Here, $U_k \in \mathbb{R}^{m\times k}$, $V_k \in \mathbb{R}^{n\times k}$, and $\Sigma_k \in \mathbb{R}^{k\times k}$. Although optional, this step is highly recommended to enhance computational efficiency and overall performance.

C CONFIGURATION EXAMPLE

This section provides a concrete example of a configuration used in our experiments. It is intended to help readers reproduce our setup and gain a better understanding of the parameters and structure involved. The example includes representative values and settings that align with the methodology described in the main text.

Each listing corresponds to a distinct use case for a regular or federated MaxFuse run on different datasets, namely CITE-seq PBMC and RNA-seq CODEX tonsils. The structure is divided into high-level categories such as `training` and `evaluation`, with further subdivisions that distinguish between `meta_data_parameters` and `non_meta_data_parameters`, reflecting MaxFuse's dual-modality optimization.

Key configurable elements include:

- **Dimensionality reduction settings** (e.g., `svd_components`, `is_randomized`)
- **Batching and matching strategy** (e.g., `matching_ratio`, `meta_cell_size`)
- **Graph construction and clustering** (e.g., Leiden resolution parameters)
- **CCA refinement loop** configurations
- **Propagation and filtering weights**
- **Evaluation metrics** used for performance benchmarking

These JSON files serve as templates for users who wish to adapt MaxFuse to their datasets.

```
{
  "dataset_id": "cite_seq_and_pbmc",
  "training": {
    "global_parameters": {
      "svd": {
        "is_randomized": false,
        "runs": 1
      },
      "split_into_batches": {
        "assignment_method": "random",
        "batching_scheme": "pairwise",
        "grouping": {
          "max_outward_size": 5000,
          "matching_ratio": 3,
          "meta_cell_size": 2
        }
      },
      "graph_construction": {
        "leiden_algorithm": {
          "resolution": 2.0,
          "resolution_tol": 0.1,
          "runs": 1
        },
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        },
        "fuzzy_smoothing": "centroid_shrinkage"
      },
      "refinement_loop": {
        "cca": {
          "randomness": 0.0,
          "components": 20,
          "max_loop_iterations": 2000,
          "bad_filter_wt": 0.0
        },
        "loop_iterations": 3,
        "previous_cell_matching_distance_wt": 0.0
      },
      "filtering": {
        "pivot_wt": 0.3,
        "propagation_wt": 0.0
      }
    },
    "meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 30,
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        }
      },
      "initial_correlation": {
        "svd_components": 25,
        "shrink_wt": 0.7
      },
      "refinement_loop": {
        "svd_components": 30,
        "smoothing_weight": 0.7
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": 30
      }
    },
    "non_meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 30,
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        }
      },
      "initial_correlation": {
        "svd_components": 20,
        "shrink_wt": 0.7
      },
      "refinement_loop": {
        "svd_components": 30,
        "smoothing_weight": 0.7
      },
      "propagation": {
        "smoothing_wt": 0.7,

        "svd_components": 30
      }
    }
  },
  "evaluation": {
    "knn_alignment_proportion": 0.05,
    "repetitions": 20,
    "principle_components": 20,
    "sub_sampling_ratio": 0.8,
    "neighborhood_k": 15
  }
}
```

Listing 1: Regular MaxFuse Run configuration for CITEseq-PBMC

```
{
  "dataset_id": "tonsils",
  "training": {
    "global_parameters": {
      "svd": {
        "is_randomized": false,
        "runs": 1
      },
      "split_into_batches": {
        "assignment_method": "random",
        "batching_scheme": "pairwise",
        "grouping": {
          "max_outward_size": 8000,
          "matching_ratio": 4,
          "meta_cell_size": 2
        }
      },
      "graph_construction": {
        "leiden_algorithm": {
          "resolution": 2.0,
          "resolution_tol": 0.1,
          "runs": 1
        },
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        },
        "fuzzy_smoothing": "centroid_shrinkage"
      },
      "refinement_loop": {
        "cca": {
          "randomness": 0.0,
          "components": 25,
          "max_loop_iterations": 2000,
          "bad_filter_wt": 0.0
        },
        "loop_iterations": 1,
        "previous_cell_matching_distance_wt": 0.0
      },
      "filtering": {
        "pivot_wt": 0.5,
        "propagation_wt": 0.3
      }
    },
    "meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 40
      },
      "initial_correlation": {
        "svd_components": 25,
        "shrink_wt": 0.3
      },
      "refinement_loop": {
        "svd_components": 40,
        "smoothing_weight": 0.3
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": 40
      }
    },
    "non_meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 15
      },
      "initial_correlation": {
        "svd_components": 20,
        "shrink_wt": 0.3
      },
      "refinement_loop": {
        "svd_components": null,
        "smoothing_weight": 0.3
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": null
      }
    }
  },
  "evaluation": {
    "knn_alignment_proportion": 0.05,
    "repetitions": 20,
    "principle_components": 20,
    "sub_sampling_ratio": 0.8,
    "neighborhood_k": 15
  }
}
```

Listing 2: Regular MaxFuse Run configuration for RNAseq-CODEX Tonsils

```
{
  "dataset_id": "cite_seq_and_pbmc",
  "training": {
    "global_parameters": {
      "svd": {
        "is_randomized": false,
        "runs": 1
      },
      "graph_construction": {
        "leiden_algorithm": {
          "resolution": 2.0,
          "resolution_tol": 0.1,
          "runs": 1
        },
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        },
        "fuzzy_smoothing": "centroid_shrinkage"
      },
      "refinement_loop": {
        "cca": {
          "randomness": 0.0,
          "components": 20,
          "max_loop_iterations": 2000,
          "bad_filter_wt": 0.0
        },
        "loop_iterations": 3,
        "previous_cell_matching_distance_wt": 0.0
      },
      "filtering": {
        "pivot_wt": 0.3,
        "propagation_wt": 0.0
      }
    },
    "meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 30,
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        }
      },
      "initial_correlation": {
        "svd_components": 25,
        "shrink_wt": 0.7
      },
      "refinement_loop": {
        "svd_components": 30,
        "smoothing_weight": 0.7
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": 30
      }
    },
    "non_meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 30,
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        }
      },
      "initial_correlation": {
        "svd_components": 20,
        "shrink_wt": 0.7
      },
      "refinement_loop": {
        "svd_components": 30,
        "smoothing_weight": 0.7
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": 30
      }
    }
  },
  "evaluation": {
    "knn_alignment_proportion": 0.05,
    "repetitions": 20,
    "principle_components": 20,
    "sub_sampling_ratio": 0.8,
    "neighborhood_k": 15
  }
}
```

Listing 3: Federated MaxFuse Run configuration for CITEseq-PBMC Antibodies

```
{
  "dataset_id": "tonsils",
  "training": {
    "global_parameters": {
      "svd": {
        "is_randomized": false,
        "runs": 1
      },
      "graph_construction": {
        "leiden_algorithm": {
          "resolution": 2.0,
          "resolution_tol": 0.1,
          "runs": 1
        },
        "randomness_seeds": {
          "leiden_seed": null,
          "nn_graph_seed": null
        },
        "fuzzy_smoothing": "centroid_shrinkage"
      },
      "refinement_loop": {
        "cca": {
          "randomness": 0.0,
          "components": 25,
          "max_loop_iterations": 2000,
          "bad_filter_wt": 0.0
        },
        "loop_iterations": 1,
        "previous_cell_matching_distance_wt": 0.0
      },
      "filtering": {
        "pivot_wt": 0.5,
        "propagation_wt": 0.3
      }
    },
    "meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 40
      },
      "initial_correlation": {
        "svd_components": 25,
        "shrink_wt": 0.3
      },
      "refinement_loop": {
        "svd_components": 40,
        "smoothing_weight": 0.3
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": 40
      }
    },
    "non_meta_data_parameters": {
      "graph_construction": {
        "nearest_neighbors": 15,
        "svd_components": 15
      },
      "initial_correlation": {
        "svd_components": 20,
        "shrink_wt": 0.3
      },
      "refinement_loop": {
        "svd_components": null,
        "smoothing_weight": 0.3
      },
      "propagation": {
        "smoothing_wt": 0.7,
        "svd_components": null
      }
    }
  },
  "evaluation": {
    "knn_alignment_proportion": 0.05,
    "repetitions": 20,
    "principle_components": 20,
    "sub_sampling_ratio": 0.8,
    "neighborhood_k": 15
  }
}
```

Listing 4: Federated MaxFuse Run configuration for RNAseq-CODEX Tonsils

D RNA TO PROTEIN NAME MAPPING

Table 7: RNA to Protein Name Correspondences

RNA name	Protein name
CD80	CD80
CD86	CD86
CD274	CD274
PDCD1LG2	CD273
TNFRSF14	CD270
TNFRSF14-AS1	CD270
TNFRSF4	CD252
PVR	CD155
NECTIN2	CD112
CD47	CD47
CD70	CD70
TNFRSF8	CD30
CD48	CD48
CD40	CD40
CD40LG	CD154
CD52	CD52
CD8A	CD8a
CD8B	CD8a
CD8A	CD8
CD8B	CD8
CD19	CD19
ITGAX	CD11c
CD34	CD34
TNFRSF17	CD269
KIT	CD117
PTPRC	CD45RA
PTPRCAP	CD45RA
IL3RA	CD123
ENG	CD105
PROCR	CD201
CCR4	CD194
CD14	CD14
IL2RA	CD25
PTPRC	CD45RO
PTPRCAP	CD45RO
PDCD1	CD279
TIGIT	TIGIT

Continued on next page

Table 7: RNA to Protein Name Correspondences (Continued)

RNA name	Protein name
MS4A1	CD20
NCR1	CD335
PTGDR2	CD294
PECAM1	CD31
PDGFRA	CD140a
PDGFRB	CD140b
ERBB2	CD340
MCAM	CD146
CDH1	CD324
CD40LG	IgM
CCR5	CD195
CCR6	CD196
CXCR5	CD185
ITGAE	CD103
CD69	CD69
CTLA4	CD152
LAG3	CD223
CD27	CD27
LAMP1	CD107a
FAS	CD95
TNFRSF4	CD134
CD1C	CD1c
FCGR1A	CD64
THBD	CD141
CD1D	CD1d
KLRK1	CD314
CR1	CD35
B3GAT1	CD57
HAVCR2	CD366
BTLA	CD272
ICOS	CD278
CD96	CD96
ENTPD1	CD39
FASLG	CD178
CX3CR1	CX3CR1
CD24	CD24
CR2	CD21

Continued on next page

Table 7: RNA to Protein Name Correspondences (Continued)

RNA name	Protein name
CD79B	CD79b
CD244	CD244
GYPA	CD235ab
GYPB	CD235ab
MRC1	CD206
SIGLEC1	CD169
CLEC9A	CD370
XCR1	XCR1
TNFRSF13C	CD268
GP1BA	CD42b
ICAM1	CD54
SELP	CD62P
IFNGR1	CD119
CD68	CD68
CCR2	CD192
ICAM2	CD102
VCAM1	CD106
IL2RB	CD122
TNFRSF13B	CD267
FLT3	CD135
ITGA2B	CD41
TNFRSF9	CD137
SPN	CD43
CD163	CD163
CD83	CD83
TNFRSF18	CD357
CD59	CD59
IL4R	CD124
ANPEP	CD13
CXCR4	CD184
CD2	CD2
ITGB1	CD29
CLEC4C	CD303
ITGA2	CD49b
ITGB3	CD61
CD81	CD81
SLC7A5	CD98

Continued on next page

Table 7: RNA to Protein Name Correspondences (Continued)

RNA name	Protein name
CD55	CD55
ITGB2	CD18
CD28	CD28
CRLF2	TSLPR
IL7R	CD127
FUT4	CD15
CD22	CD22
TFRC	CD71
CCR3	CD193
CSF1R	CD115
MSR1	CD204
CDH5	CD144
CLEC10A	CD301
CD1A	CD1a
CD207	CD207
CD63	CD63
TLR4	CD284
NRP1	CD304
CD36	CD36
SIRPA	CD172a
LILRA4	CD85g
ABCB1	CD243
CD72	CD72
MERTK	MERTK
L1CAM	CD171
CDH2	CD325
CD93	CD93
CD200	CD200
ABCG2	CD338
C5AR2	C5L2
GYPA	CD235a
ITGA1	CD49a
ITGA4	CD49d
NT5E	CD73
CD79A	CD79a
CD9	CD9
TREM1	CD354

Continued on next page

Table 7: RNA to Protein Name Correspondences (Continued)

RNA name	**Protein name**
LAIR1	CD305
ENPP3	CD203c
CD209	CD209
MPL	CD110
NCR3	CD337
TNFSF10	CD253
CXCR6	CD186
CD226	CD226
LY75	CD205
CD109	CD109
ITGB2	CD18
IL6R	CD126
CD164	CD164
F3	CD142
FCRL4	CD307d
FCRL5	CD307e
SLAMF7	CD319
CD99	CD99
CLEC12A	CLEC12A
FCGR3A	CD16
KLRB1	CD161
CCR10	CCR10
NGFR	CD271
IL6ST	GP130
CCR9	CD199
CD46	CD46
CLEC1B	CLEC2
IGHD	IgD
CRLF2	TSLPR
CD99	CD99
HLA-DRA	HLA.DR
HLA-DRB1	HLA.DR