INCREMENTAL VERTICAL FEDERATED MULTI-VIEW CLUSTERING WITH ASYNCHRONOUS CLIENT ARRIVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing vertical federated multi-view clustering methods generally assume that all clients are simultaneously available and that the client set is fixed. In practice, however, each client holds one view of the same samples, while clients may not be simultaneously valid and may join asynchronously over time. This setting exposes two limitations of current approaches. First, they typically require previous clients to participate again, which is infeasible when old clients are no longer available. Even when they remain available, such repeated participation still incurs extra communication and privacy costs. Second, given only the historical global consensus and one newly uploaded label matrix, existing methods lack an effective fusion mechanism. To address these issues, we propose an asynchronous incremental vertical federated multi-view clustering framework. Each arriving client constructs an anchor graph from its own view and learns a local label matrix, which is uploaded to the server. The server then collaborates with the current client to alternately update the global consensus and the local label matrix via an augmented Lagrangian formulation until convergence. Only the current client is involved, and the server retains only the latest consensus. Experiments on multiple datasets confirm the applicability and effectiveness of the proposed framework in incremental scenarios.

1 INTRODUCTION

Federated multi-view clustering has attracted increasing attention because it simultaneously accommodates the heterogeneity of multi-view data and the privacy constraints of federated learningChen et al. (2024). Its central problem is to achieve effective multi-view information fusion while guaranteeing that no client's raw data leaves its local device. According to how data are partitioned across clients, federated multi-view clustering can be broadly categorized into horizontal and vertical settings. In the horizontal setting, clients hold different samples under the same view, whereas in the vertical setting, clients share a common set of samples but each holds a distinct view. This paper focuses on vertical federated multi-view clustering (VF-MVC), where multiple clients each hold one view of the same samples and collaborate to uncover a global cluster structure without exposing their raw features.

Due to privacy restrictions in the federated setting, raw data cannot be uploaded to the server. Instead, the server can only coordinate clustering through auxiliary variables exchanged with clients. These variables must not leak raw data privacy, and they must also capture view-specific information to support effective fusion of multi-view information. The choice of the communication variable is thus the key to addressing the central problem of VF-MVC, since it determines both the privacy guarantee and the server-side fusion mechanism.

One representative line of work employs matrix factorization quantities as the communication variable. FedMVL Huang et al. (2022), for instance, formulates federated multi-view learning as an orthogonal matrix factorization problem and allows each node to approximately solve its local subproblem, which accommodates communication variability and node heterogeneity. While PS-FedML Li et al. (2024) adopts a partial sharing strategy within a vertical federated framework, transmitting common latent representations between clients and the server to capture both consis-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

tent and complementary information while avoiding privacy leakage. CeFMC Liu et al. (2026) approximates data representations with pseudo-labels and a centroid matrix, reducing communication overhead compared with methods that directly transmit representations. Similarly, LLF-CM and LLF-FCM Bárcena et al. (2024) enable c-means and fuzzy c-means to operate on vertically partitioned data by alternately updating cluster assignments across participants, and TensorFMVC Feng et al. (2024b) constructs a third-order tensor from view-wise indicator matrices and factorizes it to exploit complementary information across views. However, they typically transmit latent representations or factor matrices derived from raw features, which may still carry feature-level information and thus pose privacy risks. They also generally assume that all clients remain online and respond to iterative optimization requests throughout the learning process.

To avoid transmitting feature-level representations, another line of research turns to graph-based summaries as the communication variable. These summaries are more privacy-preserving than explicit feature representations, as they convey structural information without revealing raw features. Among them, FMSC Wang et al. (2020) encrypts view-specific similarity matrices for server-side aggregation and applies differential privacy, so that neither the server nor the view parties can access each other's data. EFIMVC Liu et al. (2025), rather than transmitting sensitive embeddings, employs view-specific and shared anchor graphs as communication variables, which enhances privacy and also handles missing views. A different strategy is adopted by FFCMSP Feng et al. (2024a), which transmits membership matrices and incorporates sparsity regularization to learn a clustering consensus. VFDPC Li et al. (2025), on the other hand, transmits encrypted distance matrices and performs a modified density peaks clustering under nonlinear mapping. Finally, FIMVC-TLRC Feng et al. (2026) aligns local anchor graphs and imposes a tensorized low-rank constraint based on the tensor Schatten p-norm to enforce consistency across clients. These methods confirm that anchor graphs and graph-based summaries are viable alternatives to feature-level representations. Nevertheless, they still operate under the assumption that all participating clients are simultaneously available and can respond to the server's iterative optimization requests.

Despite these advances, existing VF-MVC methods are not designed for the scenario where clients join sequentially and asynchronously. In this setting, a new client may arrive at an arbitrary time, and previously participating clients may no longer be available. Requiring old clients to participate again is therefore infeasible, and even when they remain available, repeated participation would incur unnecessary communication and privacy costs. Moreover, when only the historical global consensus and one newly uploaded label matrix from the arriving client are accessible, existing methods lack a mechanism to fuse them effectively into an updated consensus. These limitations motivate us to develop an asynchronous incremental vertical federated multi-view clustering framework. Specifically, each newly arriving client can join without requiring previous clients to participate again, and the historical consensus is updated by solving an optimization problem that fuses it with the newly uploaded information.

The contributions of our paper are summarized as follows:

- We propose, to the best of our knowledge, the first incremental vertical federated multi-view clustering framework. It accommodates clients that arrive sequentially, and thus avoids the need to re-coordinate all previous clients whenever a new view becomes available.
- We design an alternating optimization scheme in which the server interacts only with the current client, and previous clients are never re-engaged. This eliminates secondary communication with old clients and keeps the server-side storage independent of the number of historical clients.
- The proposed framework imposes no restriction on the local clustering model. Each client can adopt any clustering algorithm that suits its own view, and only needs to upload a label matrix that reflects its local cluster structure.
- Extensive experiments on multiple datasets show that the proposed method achieves clustering performance comparable to, and in some cases even better than, both centralized clustering methods and conventional federated multi-view clustering methods.

2 PROBLEM FORMULATION

In the asynchronous incremental federated learning setting, each client holds the complete data of one view, but the data of different views involved in federated clustering are not simultaneously available. After the federated clustering learning with the existing clients is completed, at any subsequent time, a new client may request to join the federated clustering learning. Assume that before the arrival of the v-th client, the server has already obtained the global clustering consensus $\mathbf{H}$ of the currently participating clients through incremental learning. When the v-th client arrives, the server needs to perform alternating iterative updates with the new client without "disturbing" the existing clients (i.e., without requiring the existing clients to participate in the federated learning process again), so as to absorb the clustering structure information of the new client and learn a new global clustering consensus $\mathbf{H}_{\text{new}}$.

3 METHODOLOGY

3.1 CLIENT-SIDE LOCAL CLUSTERING AND LABEL MATRIX GENERATION

To protect local privacy, clients cannot upload their raw data to the server. Each client is therefore responsible for the quality of its own local clustering, since a better local clustering result contributes more to the global federated clustering performance. To this end, we adopt the label matrix as the upload medium. This choice helps protect privacy and preserves the clustering structure of local samples. Moreover, it imposes no restriction on the local clustering model. The local clustering models across clients can be heterogeneous and autonomous, and the server only needs to receive the uploaded label matrix rather than the details of each local model. As long as the model is well-founded and capable of producing label information that reflects the local data structure, a client only needs to upload a label matrix that it considers representative of its own data structure.

To obtain a label matrix that faithfully reflects the local clustering structure, we adopt the anchor graph to capture the relationships between samples and anchors. Since clients cannot communicate with each other, each client has to select anchors independently and construct its own anchor graph. Such independent anchor selection enables the client to better characterize its local data distribution. Based on this anchor graph, we formulate a client-side objective function, through which the final local label matrix $\mathbf{H}_v$ is obtained. For the v-th client, let $\mathbf{S}_v \in \mathbb{R}^{n\times m}$ denote its anchor graph, where n and m are the numbers of samples and anchors, respectively, and let $\mathbf{H}_v \in \mathbb{R}^{n\times k}$ be the label matrix with k clusters. The objective function is defined as follows:

$$\begin{aligned} \max_{\mathbf{H}_v} \quad & \left\|\mathbf{S}_v^\top \mathbf{H}_v\right\|_{sp}^{p_1} + \lambda_v \mathrm{Tr}(\mathbf{H}_v^\top \mathbf{D}_v \mathbf{H}_v), \\ \text{s.t.} \quad & \mathbf{H}_v \geq 0,\ \mathbf{H}_v \mathbf{1} = \mathbf{1}, \end{aligned} \tag{1}$$

where $\mathbf{D}_v = \mathbf{S}_v \mathbf{S}_v^\top$, $\lambda_v > 0$ balances the two terms, and $0 < p_1 \leq 1$ controls the sharpness of the first term. The first term $\|\mathbf{S}_v^\top \mathbf{H}_v\|_{sp}^{p_1}$ encourages anchors to spread across different clusters rather than concentrating on a few dominant ones, thereby improving the representativeness of the anchors. The second term promotes smoothness of the label matrix over the anchor-induced sample manifold and enhances the discriminative capability of the learned clustering structure. The client solves this problem iteratively from an initial $\mathbf{H}_v$ and uploads the resulting label matrix to the server. The detailed optimization is presented in Section 3.3.

3.2 SERVER-SIDE INCREMENTAL GLOBAL LABEL MATRIX FUSION

After the newly arrived client uploads its label matrix, the server updates the global clustering consensus by interacting only with this client, while the previously participating clients are no longer involved. Consequently, the communication cost per update depends only on the current client and does not grow with the number of historical clients. This also alleviates the privacy concern arising from repeated participation of old clients. Moreover, the server only needs to retain the latest consensus label matrix, rather than the full history of client label matrices, which keeps the storage overhead low and independent of the number of historical clients. The previous consensus label matrix summarizes the clustering structure information of the clients that have already participated,

while the label matrix uploaded by the newly arrived client brings new clustering structure information. To capture the structural relationship between the previous consensus label matrix and the newly uploaded label matrix, we stack them into a tensor and formulate the server-side objective as follows.

$$\begin{aligned}\min_{\mathbf{H}_s^{(t)},\alpha_s^{(t)},\mathbf{H}_v^{(t)},\alpha_v^{(t)}} \quad & \frac{1}{\alpha_s^{(t)}}\left\|\mathbf{H}_s^{(t)}-\mathbf{H}_s^{(t-1)}\right\|_F^2+\frac{1}{\alpha_v^{(t)}}\left\|\mathbf{H}_v^{(t)}-\mathbf{H}_v^{(t-1)}\right\|_F^2+\lambda\left\|\mathcal{H}^{(t)}\right\|_{\circledS p}^p \\ \text{s.t.} \quad & \mathbf{H}^{(t)}\mathbf{1}=\mathbf{1}, \quad \mathbf{H}^{(t)}\geq 0, \quad \alpha_v^{(t)}\geq 0, \alpha_v^{(t)}\geq 0, \quad \alpha_v^{(t)}+\alpha_s^{(t)}=1,\end{aligned} \tag{2}$$

where t represents the current iteration round, $\lambda>0$, $0<p\leq 1$, and $\mathcal{H}^{(t)}$ is the third-order tensor formed by $\mathbf{H}_s^{(t)}$ and $\mathbf{H}_v^{(t)}$. $\alpha_s^{(t)}$ and $\alpha_v^{(t)}$ represent the dynamic learnable weights of the server and newly joined clients at the t-th iteration, respectively. Before the iterations begin, $\mathbf{H}_s^{(0)}$ denotes the consensus label matrix stored by the server before the iterative update, and $\mathbf{H}_v^{(0)}$ denotes the label matrix uploaded by the newly joined client. Using this objective function, the server updates $\mathbf{H}_s$ and $\mathbf{H}_v$ by performing iterative alternating optimization with the newly joined client. Once stable convergence is reached, the server will compute the globally optimal label matrix:

$$\mathbf{H}^*=\frac{\frac{1}{\alpha_s}\mathbf{H}_s+\frac{1}{\alpha_v}\mathbf{H}_v}{\frac{1}{\alpha_s}+\frac{1}{\alpha_v}}. \tag{3}$$

3.3 Update Mechanism and Alternating Optimization

We now present the optimization and update mechanism for the client-side and server-side models. The client-side objective in equation 1 and the server-side objective in equation 2 are both non-convex with respect to the involved variables, and their solutions are coupled through the uploaded label matrix. To address this, we adopt an alternating optimization strategy between the newly joined client and the server. Specifically, the newly joined client uploads its label matrix to the server, and the server performs one iteration to update both the previous consensus and the newly received label matrix. The updated client label matrix is then sent back to the current client, which takes it as the initialization and refines it locally by solving the client-side objective. The refined label matrix is uploaded again, and this process repeats until the server-side consensus converges, yielding the final global clustering consensus. In what follows, we first derive the optimization steps for the client-side model, then present the solution to the server-side model.

Since the client-side objective involves only $\mathbf{H}_v$ as the optimization variable, we solve it by projected gradient ascent. At each iteration, the gradient is computed from the current $\mathbf{H}_v$, and $\mathbf{H}_v$ is updated and projected onto the probability simplex. This procedure repeats until convergence. The detailed derivation is given below.

To solve equation 1, we first note that

$$\left\|\mathbf{S}_v^\top\mathbf{H}_v\right\|_{S_{p_1}}^{p_1}=\sum_{i=1}^{r}\sigma_i^{p_1}, \tag{4}$$

where $\sigma_1,\ldots,\sigma_r$ are the non-zero singular values of $\mathbf{Z}=\mathbf{S}_v^\top\mathbf{H}_v$, with $r=\operatorname{rank}(\mathbf{Z})$. Since $p_1\in(0,1)$, the function σ^{p_1} is concave on $\sigma\geq 0$, and its derivative becomes unbounded as $\sigma\to 0$. To enable gradient-based optimization, we introduce a smoothing parameter $\epsilon>0$ and approximate

$$\sigma_i^{p_1}\approx\left(\sigma_i^2+\epsilon^2\right)^{p_1/2}. \tag{5}$$

The smoothed first term is then

$$f_\epsilon(\mathbf{H}_v)=\sum_{i=1}^{r}\left(\sigma_i^2+\epsilon^2\right)^{p_1/2}. \tag{6}$$

At the t-th iteration, let $\mathbf{Z}^{(t)}=\mathbf{S}_v^\top\mathbf{H}_v^{(t)}$ and compute its singular value decomposition as

$$\mathbf{Z}^{(t)}=\mathbf{U}^{(t)}\mathbf{\Sigma}^{(t)}(\mathbf{V}^{(t)})^\top, \tag{7}$$

where $\mathbf{\Sigma}^{(t)} = \mathrm{diag}(\sigma_1^{(t)}, \ldots, \sigma_r^{(t)})$. Define

$$\mathbf{P}^{(t)} = \mathrm{diag}\left(\left((\sigma_1^{(t)})^2 + \epsilon^2\right)^{\frac{p_1-2}{2}}, \ldots, \left((\sigma_r^{(t)})^2 + \epsilon^2\right)^{\frac{p_1-2}{2}} \right). \tag{8}$$

The gradient of f_ϵ with respect to $\mathbf{Z}$ at $\mathbf{Z}^{(t)}$ is

$$\nabla_{\mathbf{Z}} f_\epsilon(\mathbf{Z}^{(t)}) = p_1 \, \mathbf{U}^{(t)} \mathbf{\Sigma}^{(t)} \mathbf{P}^{(t)} (\mathbf{V}^{(t)})^\top. \tag{9}$$

Accordingly, the gradient with respect to $\mathbf{H}_v$ at $\mathbf{H}_v^{(t)}$ is

$$\nabla_{\mathbf{H}_v} f_\epsilon(\mathbf{H}_v^{(t)}) = \mathbf{S}_v \nabla_{\mathbf{Z}} f_\epsilon(\mathbf{Z}^{(t)}). \tag{10}$$

For the second term, let

$$g(\mathbf{H}_v) = \lambda_v \mathrm{Tr}\left(\mathbf{H}_v^\top \mathbf{D}_v \mathbf{H}_v\right). \tag{11}$$

Its gradient is

$$\nabla_{\mathbf{H}_v} g(\mathbf{H}_v) = 2\lambda_v \mathbf{D}_v \mathbf{H}_v. \tag{12}$$

Combining the two parts, the overall gradient at the t-th iteration is

$$\mathbf{G}^{(t)} = \nabla_{\mathbf{H}_v} f_\epsilon(\mathbf{H}_v^{(t)}) + 2\lambda_v \mathbf{D}_v \mathbf{H}_v^{(t)}. \tag{13}$$

We update $\mathbf{H}_v$ by projected gradient ascent:

$$\mathbf{H}_v^{(t+1)} = \mathrm{Proj}_{\mathcal{C}}\left(\mathbf{H}_v^{(t)} + \eta \mathbf{G}^{(t)}\right), \tag{14}$$

where $\eta > 0$ is the step size and $\mathcal{C} = \{\mathbf{H}_v \geq 0, \ \mathbf{H}_v \mathbf{1} = \mathbf{1}\}$. The projection $\mathrm{Proj}_{\mathcal{C}}(\cdot)$ is performed row-wise onto the probability simplex, ensuring that each row of $\mathbf{H}_v$ remains a valid soft cluster assignment. The above steps are repeated until the change in $\mathbf{H}_v$ falls below a predefined threshold.

For the server-side problem in equation 2, we introduce an auxiliary variable $\mathcal{J}$ and a Lagrange multiplier $\mathcal{Y}$, and adopt the augmented Lagrangian method with a penalty parameter $\rho > 0$ to reformulate the original objective as follows:

$$\begin{aligned} \min_{\mathbf{H}_s^{(t)}, \alpha_s^{(t)}, \mathbf{H}_v^{(t)}, \alpha_v^{(t)}} \quad & \frac{1}{\alpha_s^{(t)}} \left\| \mathbf{H}_s^{(t)} - \mathbf{H}_s^{(t-1)} \right\|_F^2 + \frac{1}{\alpha_v^{(t)}} \left\| \mathbf{H}_v^{(t)} - \mathbf{H}_v^{(t-1)} \right\|_F^2 \\ & + \lambda \left\| \mathcal{J}^{(t)} \right\|_{\mathfrak{D}}^p + \frac{\rho}{2} \left\| \mathcal{H}^{(t)} - \mathcal{J}^{(t)} + \frac{\mathcal{Y}^{(t)}}{\rho} \right\|_F^2 \\ \text{s.t.} \quad & \mathbf{H}^{(t)} \mathbf{1} = \mathbf{1}, \quad \mathbf{H}^{(t)} \geq 0, \\ & \alpha_s^{(t)} \geq 0, \alpha_v^{(t)} \geq 0, \quad \alpha_v^{(t)} + \alpha_s^{(t)} = 1, \end{aligned} \tag{15}$$

The optimization proceeds as follows:

• **Update of $\mathbf{H}_s^{(t)}$ with the Remaining Variables Fixed.** After fixing the remaining variables, the problem becomes:

$$\min_{\mathbf{H}_s^{(t)} \mathbf{1} = \mathbf{1}, \mathbf{H}_s^{(t)} \geq 0} \frac{1}{\alpha_s^{(t)}} \left\| \mathbf{H}_s^{(t)} - \mathbf{H}_s^{(t-1)} \right\|_F^2 + \frac{\rho}{2} \left\| \mathbf{H}_s^{(t)} - \mathbf{J}_s^{(v)} + \frac{\mathbf{Y}_s^{(t)}}{\rho} \right\|_F^2. \tag{16}$$

Expanding the Frobenius norms yields:

$$\min_{\mathbf{H}_s^{(t)} \mathbf{1} = \mathbf{1}, \mathbf{H}_s^{(t)} \geq 0} \mathrm{tr}\left(\mathbf{H}_s^{(t)\top} \mathbf{H}_s^{(t)}\right) - \frac{2\alpha_s^{(t)}}{2 + \alpha_s^{(t)} \rho} \mathrm{tr}\left(\mathbf{H}_s^{(t)\top} \mathbf{B}^{(t)}\right), \tag{17}$$

where $\mathbf{B}^{(t)} = \frac{2}{\alpha_s^{(t)}} \mathbf{H}_s^{(t-1)} + \rho \mathbf{J}_s^{(t)} - \mathbf{Y}_s^{(t)}$. Since the constraints are imposed row-wise, the problem in equation 17 can be decomposed into n independent subproblems, each corresponding to one row

of $\mathbf{H}_s^{(t)}$. Let $\mathbf{h}_i$ denote the i-th row of $\mathbf{H}_s^{(t)}$, and let $\mathbf{c}_i = \frac{2\alpha_s^{(t)}}{2+\alpha_s^{(t)}\rho}\mathbf{b}_i$, where $\mathbf{b}_i$ is the i-th row of $\mathbf{B}^{(t)}$. The subproblem for $\mathbf{h}_i$ is

$$\min_{\mathbf{h}_i^\top \mathbf{1}=1,\, \mathbf{h}_i \geq 0} \ \mathbf{h}_i^\top \mathbf{h}_i - \mathbf{c}_i^\top \mathbf{h}_i. \tag{18}$$

Completing the square, this is equivalent to

$$\min_{\mathbf{h}_i^\top \mathbf{1}=1,\, \mathbf{h}_i \geq 0} \ \left\| \mathbf{h}_i - \frac{1}{2}\mathbf{c}_i \right\|_2^2, \tag{19}$$

which is the Euclidean projection of $\frac{1}{2}\mathbf{c}_i$ onto the probability simplex. This projection admits a closed-form solution that can be computed efficiently by the algorithm in Duchi et al. (2008).

• **Update of $\mathbf{H}_v^{(t)}$.** The solution for $\mathbf{H}_v^{(t)}$ follows a procedure analogous to that for $\mathbf{H}_s^{(t)}$ and is therefore omitted.

• **Update of $\mathcal{J}^{(t)}$ with the Remaining Variables Fixed.** With the remaining variables fixed, the objective for $\mathcal{J}^{(t)}$ is:

$$\min_{\mathcal{J}^{(t)}} \lambda \|\mathcal{J}^{(t)}\|_{\circledS p}^p + \frac{\rho}{2} \left\| \mathcal{H}^{(t)} + \frac{\mathcal{Y}^{(t)}}{\rho} - \mathcal{J}^{(t)} \right\|_F^2. \tag{20}$$

Applying Theorem 3 in Gao et al. (2021), equation 20 has a closed-form solution. The optimal $\mathcal{J}^{(t)}$ is:

$$\mathcal{J}^{(t)^*} = \Gamma_{\frac{\lambda}{\rho}} \left(\mathcal{H}^{(t)} + \frac{\mathcal{Y}^{(t)}}{\rho} \right). \tag{21}$$

• **Theorem 3 Gao et al. (2021).** *Given $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ with t-SVD $\mathcal{B} = \mathcal{U} * \mathcal{M} * \mathcal{V}^\top$, the optimal solution of*

$$\min_{\mathcal{A}} \frac{1}{2}\|\mathcal{A} - \mathcal{B}\|_F^2 + \tau \|\mathcal{A}\|_{\circledS p}^p \tag{22}$$

is

$$\mathcal{A}^* = \Gamma_{\tau \cdot n_3}(\mathcal{B}) = \mathcal{U} * \text{ifft}\left(\mathbf{P}_{\tau \cdot n_3}(\overline{\mathcal{B}})\right) * \mathcal{V}^\top, \tag{23}$$

where $\overline{\mathcal{B}}$ is obtained by applying FFT along the third dimension of $\mathcal{B}$, and ifft *denotes the inverse FFT. The tensor $\mathbf{P}_{\tau \cdot n_3}(\overline{\mathcal{B}})$ can be computed using the GST algorithm from Lemma 1 of Gao et al. (2021).*

• **Update of $\alpha_s^{(t)}$ with the Remaining Variables Fixed.** With the remaining variables fixed, the subproblem for $\alpha_s^{(t)}$ becomes:

$$\begin{aligned} \min_{\alpha_s^{(t)}} \quad & \frac{1}{\alpha_s^{(t)}} \left\| \mathbf{H}_s^{(t)} - \mathbf{H}_s^{(t-1)} \right\|_F^2 + \frac{1}{\alpha_v^{(t)}} \left\| \mathbf{H}_v^{(t)} - \mathbf{H}_v^{(t-1)} \right\|_F^2 \\ \text{s.t.} \quad & \alpha_s^{(t)} \geq 0, \alpha_v^{(t)} \geq 0, \quad \alpha_v^{(t)} + \alpha_s^{(t)} = 1. \end{aligned} \tag{24}$$

Introducing a Lagrangian and setting the derivative to zero yields:

$$\alpha_s^{(t)} = \frac{\left\| \mathbf{H}_s^{(t)} - \mathbf{H}_s^{(t-1)} \right\|_F}{\left\| \mathbf{H}_s^{(t)} - \mathbf{H}_s^{(t-1)} \right\| + \left\| \mathbf{H}_v^{(t)} - \mathbf{H}_v^{(t-1)} \right\|_F}. \tag{25}$$

• **Update of $\alpha_v^{(t)}$.** The solution for $\alpha_v^{(t)}$ follows a procedure analogous to that for $\alpha_s^{(t)}$ and is therefore omitted.

The two subproblems are solved alternately until the server-side consensus converges, i.e., $\|\mathbf{H}_*^{(t+1)} - \mathbf{H}_*^{(t)}\|_F < \epsilon$. Upon convergence, the final global consensus is obtained. We evaluate the proposed framework on multiple datasets in the next section.

4 EXPERIMENT

4.1 DATASETS AND COMPETITORS

Our experiments were conducted on six widely-used benchmark datasets: **Yale** Georghiades et al. (2002), **3-sources** Feng et al. (2024b), **Sonar** Yang & Sinaga (2019), **RGB-D** Silberman et al. (2012), **Vehicle Sensor (VS)** Duarte & Hu (2004), and **NUS-WIDE (NUS)** Chua et al. (2009). All experiments are conducted a Windows 11 with AMD Ryzen 5 9600X 6-Core Processor and 32 GB RAM, using MATLAB R2025b. To evaluate the performance of federated clustering, we adopt three widely used metrics: Accuracy (ACC), Normalized Mutual Information (NMI), and Purity.

To evaluate the performance of the proposed incremental vertical federated multi-view clustering algorithm, we conduct experiments comparing it with representative centralized (GMC Wang et al. (2019), UDBGL Fang et al. (2023), FastMICE Huang et al. (2023)) and federated (FFCMSP Feng et al. (2024a), LLF-CM and LLF-FCM Bárcena et al. (2024), TensorFMVC Feng et al. (2024b), VFDPC Li et al. (2025), and CeFMC Liu et al. (2026), which exchange matrix factorization quantities or graph-based summaries between clients and the server.

4.2 CLUSTERING PERFORMANCE

Tables 1 and 2 summarize the clustering results, where the best value in each column is highlighted in bold and the second-best is underlined. All datasets are preprocessed via max-normalization to standardize feature scales.

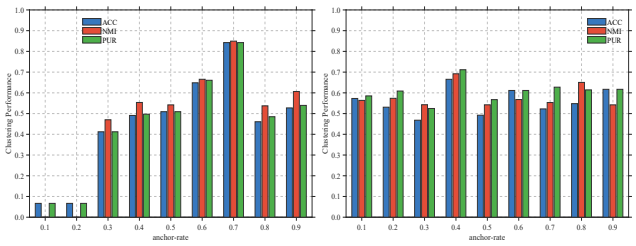
Table 1: The results on the Yale, 3-sources, Sonar, and RGB-D datasets.

Datasets	Yale			3-sources			Sonar			RGB-D		
Methods	ACC	NMI	PUR	ACC	NMI	PUR	ACC	NMI	PUR	ACC	NMI	PUR
GMC	0.212	0.275	0.242	0.707	0.652	0.793	0.505	0.045	0.534	0.402	0.331	0.465
UDBGL	0.527	0.659	0.546	0.349	0.056	0.355	0.572	0.016	0.572	0.439	0.360	0.536
FastMICE	0.624	0.570	0.655	0.556	0.503	0.710	0.582	0.032	0.582	0.418	0.326	0.495
FFCMSP	0.676	0.737	0.739	0.805	0.661	0.805	0.726	0.171	0.726	0.467	0.271	0.518
LLF-CM	0.439	0.505	0.464	0.426	0.150	0.460	0.555	0.009	0.555	0.240	0.151	0.388
LLF-FCM	0.472	0.556	0.500	0.637	0.379	0.660	0.553	0.009	0.553	0.263	0.014	0.266
TensorFMVC	0.794	0.785	0.794	0.722	0.662	0.811	**0.986**	**0.892**	**0.986**	0.578	0.638	**0.801**
VFDPC	0.563	0.613	0.570	0.404	0.189	0.466	0.526	0.003	0.537	0.410	0.300	0.486
CeFMC	0.602	0.659	0.623	0.651	0.575	0.757	0.591	0.048	0.592	0.442	0.333	0.545
Our work	**0.842**	**0.879**	**0.848**	**0.882**	**0.785**	**0.882**	0.880	0.475	0.880	**0.665**	**0.692**	0.712

It is worth noting that the proposed method operates in an asynchronous incremental setting, where clients arrive sequentially and previously participating clients are no longer involved. This setting is more challenging than the conventional synchronous setting adopted by the compared methods. Despite this, the proposed method remains highly competitive. On Yale, 3-sources, VS, and NUS, it achieves the best or joint-best performance. For instance, on Yale, it outperforms the second-best method by 4.8, 9.4, and 5.4 percentage points in ACC, NMI, and PUR, respectively. On RGB-D, it surpasses the second-best method by 8.7 and 5.4 percentage points in ACC and NMI, while its PUR is only slightly below the best result. On Sonar, it ranks second, behind the centralized TensorFMVC. These results indicate that the proposed method maintains clustering quality comparable to centralized and conventional federated approaches, while its primary advantage lies in addressing the incremental scenario where previous clients cannot be revisited.

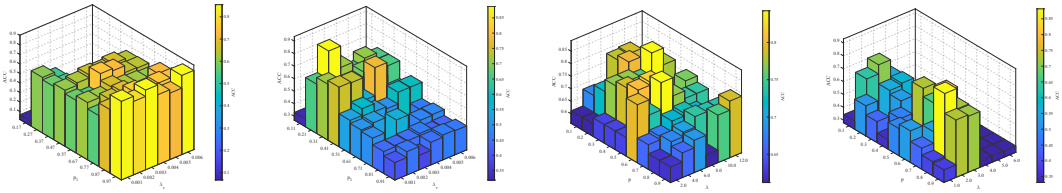
Table 2: The results on the VS and NUS datasets.

Datasets	VS			NUS		
Methods	ACC	NMI	PUR	ACC	NMI	PUR
GMC	0.804	0.288	0.804	0.345	0.403	0.365
UDBGL	0.513	0.001	0.513	0.123	0.070	0.170
FastMICE	0.515	0.001	0.517	0.143	0.132	0.238
FFCMSP	**0.999**	**0.993**	**0.999**	0.251	0.195	0.341
LLF-CM	0.701	0.142	0.701	0.122	0.116	0.213
LLF-FCM	0.726	0.163	0.726	0.118	0.107	0.212
TensorFMVC	**0.999**	**0.993**	**0.999**	0.292	0.364	0.315
VFDPC	0.789	0.278	0.789	0.104	0.073	0.185
CeFMC	0.786	0.252	0.786	0.118	0.111	0.207
Our work	**0.999**	**0.993**	**0.999**	**0.533**	**0.639**	**0.533**



(a) Yale (b) RGB-D

Figure 1: Effect of anchor rate on clustering results



(a) Yale:client (b) 3-sources:client (c) Yale:server (d) 3-sources:server

Figure 2: Joint effects of the client-side parameters p_1 and λ_v in (a)(b) and the server-side parameters p and λ in (c)(d) on ACC.

4.3 Parameter Analysis

To investigate the impact of the anchor rate on clustering performance, we conduct experiments on the Yale and RGB-D datasets, with results shown in Fig. 1, respectively. On Yale, the performance varies non-monotonically with the anchor rate. It remains low when the anchor rate is small, improves as more anchors are used, and reaches its peak at around 0.7. Beyond that point, further increasing the anchor rate leads to a clear decline. This suggests that Yale is highly sensitive to the anchor rate, and an excessively large number of anchors may introduce redundancy or noise. In contrast, RGB-D exhibits a much more stable trend, with only mild fluctuations across different anchor rates. The optimal anchor rate for RGB-D also differs from that of Yale, indicating that the best choice is dataset-dependent. Neither a larger nor a smaller anchor rate is universally beneficial; an appropriate rate helps anchors better represent the underlying data structure, thereby improving clustering performance.

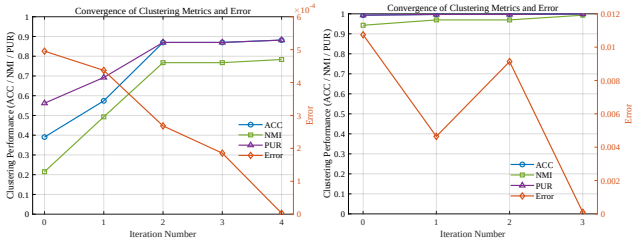
To investigate the joint effect of the client-side parameters p_1 and λ_v on clustering performance, we conduct experiments on the Yale and 3-sources datasets, as shown in Fig. 2 (a) and (b). On Yale, the accuracy improves steadily as p_1 increases and remains high over a broad range of λ_v, indicating that Yale favors a relatively large p_1 and is not sensitive to λ_v. In contrast, 3-sources achieves its peak accuracy at a small p_1 and a small λ_v, and its performance drops sharply once p_1 exceeds this narrow region, showing strong sensitivity to p_1. These results suggest that the optimal combination of p_1 and λ_v is dataset-dependent, and a proper parameter pair is essential for the model to capture the underlying data structure and achieve satisfactory clustering performance.

The joint effects of the server-side parameters p and λ on ACC are shown in Fig. 2 (c) and (d). On Yale, the performance is mainly affected by p and peaks at a moderate value, while λ has only a mild influence. In contrast, for 3-sources, the performance is highly sensitive to λ. A clear peak appears at a small λ combined with a moderate p, while increasing λ beyond this region causes a sharp drop in ACC across almost all p values. This indicates that 3-sources favors a small λ and a well-matched p, and that an inappropriate combination can severely degrade clustering performance.

4.4 Convergence Analysis

When the last client arrives, we evaluate the convergence of the alternating iterative optimization between the server and the newly arrived client. we plot the clustering metrics (ACC, NMI, PUR)

and the server-side error versus the iteration number on the 3-sources and VS datasets, as shown in Fig. 3. On both datasets, the clustering performance improves steadily as the iteration proceeds, and the error generally decreases and eventually approaches zero. For 3-sources, the metrics increase rapidly in the first two iterations and then stabilize, while the error drops to near zero within four iterations. For VS, the metrics remain at a high level and converge quickly, and the error approaches zero in three iterations despite a slight fluctuation in the intermediate step. These results suggest that the proposed alternating optimization can converge within a few iterations on both datasets.



(a) 3-sources (b) VS

Figure 3: Error and clustering performance versus iteration on the two datasets.

5 Conclusion

In this paper, we addressed vertical federated multi-view clustering in a practical scenario where clients arrive sequentially and asynchronously, and previously participating clients may no longer be available. Existing methods generally assume that all clients are simultaneously present and require their repeated participation, which is impractical in such a setting. To overcome this limitation, we proposed an asynchronous incremental framework in which each arriving client learns a local label matrix from its own view and uploads only this matrix to the server, without imposing any restriction on the local clustering model. The server then collaborates exclusively with the current client to alternately update the global consensus and the newly received label matrix until convergence, without re-engaging any previous client. This design reduces communication and privacy costs and keeps the server-side storage independent of the number of historical clients. Experiments on multiple datasets demonstrated the feasibility and effectiveness of the proposed framework in the incremental setting.

AI use statement

(This section is **required** and does not count toward the page limit.)

In this work, we did not use generative AI tools for any tasks. We take responsibility for the final content of this work.

Ethics statement

(This section is **recommended** and does not count toward the page limit.)

This work focuses on multi-view clustering using publicly available benchmark datasets. It does not involve human subjects, personal data, or any sensitive information. All datasets are publicly released and properly cited, and their licenses are respected. We do not foresee any harmful insights, discrimination, or fairness concerns arising from this work. The authors declare no conflicts of interest. No IRB approval was required because no human subjects were involved. We take responsibility for the research integrity of this work.

REFERENCES

José Luis Corcuera Bárcena, Francesco Marcelloni, Alessandro Renda, Alessio Bechini, and Pietro Ducange. Federated c-means and fuzzy c-means clustering algorithms for horizontally and vertically partitioned data. *IEEE Transactions on Artificial Intelligence*, 5(12):6426–6441, 2024. doi: 10.1109/TAI.2024.3426408.

Xinyue Chen, Yazhou Ren, Jie Xu, Fangfei Lin, Xiaorong Pu, and Yang Yang. Bridging gaps: Federated multi-view clustering in heterogeneous hybrid views. In *Advances in Neural Information Processing Systems*, volume 37, pp. 37020–37049, 2024. doi: 10.52202/079017-1168.

Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pp. 1–9, 2009.

Marco F Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004. doi: https://doi.org/10.1016/j.jpdc.2004.03.020.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 272–279, New York, NY, USA, 2008. ISBN 9781605582054. doi: 10.1145/1390156.1390191.

Si-Guo Fang, Dong Huang, Xiao-Sha Cai, Chang-Dong Wang, Chaobo He, and Yong Tang. Efficient multi-view clustering via unified and discrete bipartite graph learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):11436–11447, 2023. doi: 10.1109/TNNLS.2023.3261460.

Wei Feng, Zhenwei Wu, Qianqian Wang, Bo Dong, and Quanxue Gao. Federated fuzzy c-means with schatten-p norm minimization. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pp. 9407–9416, 2024a. doi: 10.1145/3664647.3681557.

Wei Feng, Zhenwei Wu, Qianqian Wang, Bo Dong, Zhiqiang Tao, and Quanxue Gao. Federated multi-view clustering via tensor factorization. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pp. 3962–3970. ijcai.org, 2024b.

Wei Feng, Danting Liu, Qianqian Wang, Mengping Jiang, and Bin Liu. Federated incomplete multi-view clustering with tensorized low-rank constraint. In *AAAI*, pp. 21083–21091, 2026.

Quanxue Gao, Pu Zhang, Wei Xia, Deyan Xie, Xinbo Gao, and Dacheng Tao. Enhanced tensor rpca and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):2133–2140, 2021. doi: 10.1109/TPAMI.2020.3017672.

Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2002. doi: 10.1109/34.927464.

Dong Huang, Chang-Dong Wang, and Jian-Huang Lai. Fast multi-view clustering via ensembles: Towards scalability, superiority, and simplicity. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11388–11402, 2023. doi: 10.1109/TKDE.2023.3236698.

Shudong Huang, Wei Shi, Zenglin Xu, Ivor Wai-Hung Tsang, and Jiancheng Lv. Efficient federated multi-view learning. *Pattern Recognit.*, 131:108817, 2022.

Chao Li, Shifei Ding, Xiao Xu, Lili Guo, Ling Ding, and Xindong Wu. Vertical federated density peaks clustering under nonlinear mapping. *IEEE Transactions on Knowledge and Data Engineering*, 37(2):1004–1017, 2025. doi: 10.1109/TKDE.2024.3487534.

Daoyuan Li, Zuyuan Yang, Jiawen Kang, Minfan He, and Shengli Xie. Partially shared federated multiview learning. *Knowledge-Based Systems*, 301:112302, 2024. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2024.112302.

Jiyuan Liu, Xinwang Liu, Siqi Wang, Xinhang Wan, Dongsheng Li, Kai Lu, and Kunlun He. Communication-efficient federated multi-view clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(1):17–32, 2026. doi: 10.1109/TPAMI.2025.3601533.

Suyuan Liu, Hao Yu, Hao Tan, Ke Liang, Siwei Wang, Shengju Yu, En Zhu, and Xinwang Liu. Efficient federated incomplete multi-view clustering. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 40100–40114, 13–19 Jul 2025.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pp. 746–760, 2012. doi: https://doi.org/10.1007/978-3-642-33715-4_54.

Hao Wang, Yan Yang, and Bing Liu. Gmc: Graph-based multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1116–1129, 2019. doi: 10.1109/TKDE.2019.2903810.

Hongtao Wang, Ang Li, Bolin Shen, Yuyan Sun, and Hongmei Wang. Federated multi-view spectral clustering. *IEEE Access*, 8:202249–202259, 2020. doi: 10.1109/ACCESS.2020.3036747.

Miin-Shen Yang and Kristina P Sinaga. A feature-reduction multi-view k-means clustering algorithm. *IEEE access*, 7:114472–114486, 2019. doi: 10.1109/ACCESS.2019.2934179.