A Update the discrete dependent variable space via EMA

We can use this algorithm to update e in the space K. Now we consider the set $z_{i,1}, z_{i,2}, ..., z_{i,k_i}$ as the set of k_i samples from the encoder which are closed to e_i, and to update e_i, we can minimize the loss:

$$L = \sum_{j}^{k_i} \|z_{i,j=1} - e_i\|^2$$

Therefore, if we consider the cumulant of k_i at time t as K_i^t, and the cumulant of z_i at time t as Z_i^t, we can update K_i^t, Z_i^t, e_i^t using:

$$K_i^{(t)} := K_i^{(t-1)} * \gamma + k_i^{(t)}(1-\gamma)$$
$$Z_i^{(t)} := Z_i^{(t-1)} * \gamma + \sum_j z_{i,j}^{(t)}(1-\gamma)$$
$$e_i^{(t)} := \frac{Z_i^{(t)}}{K_i^{(t)}},$$

where $\gamma \in (0, 1)$ and we choose $\gamma = 0.99$ in our model.

B Multi-omics data integration results

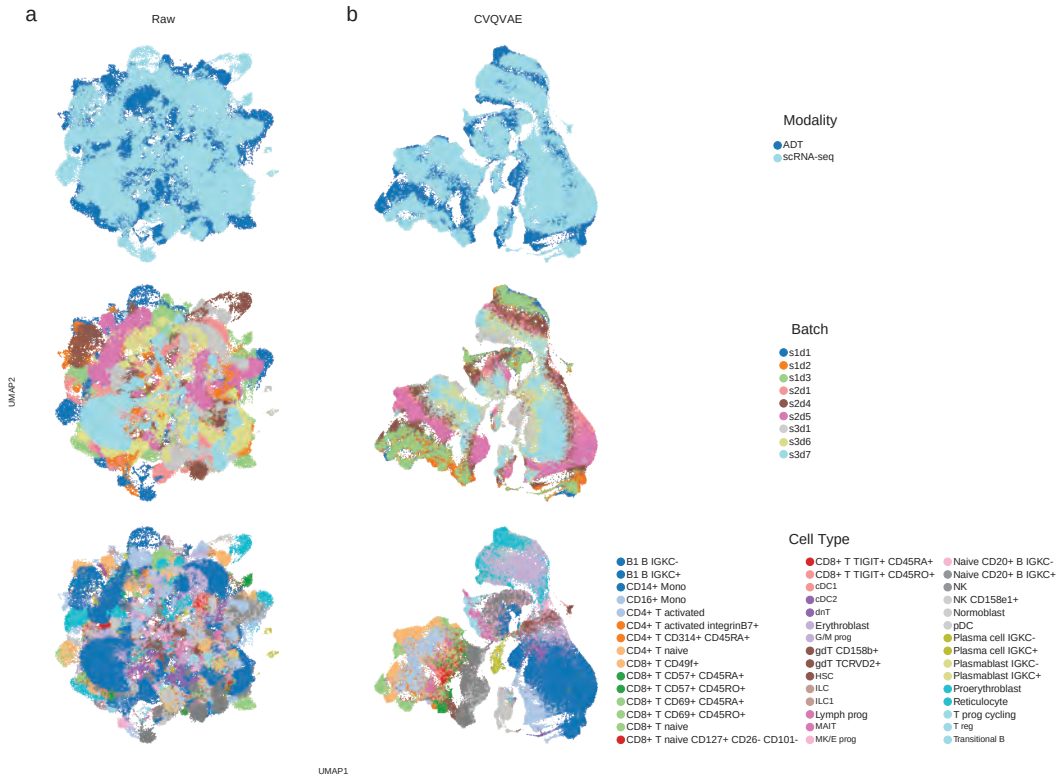

Figure 7: Protein embeddings data from raw data and CVQVAE

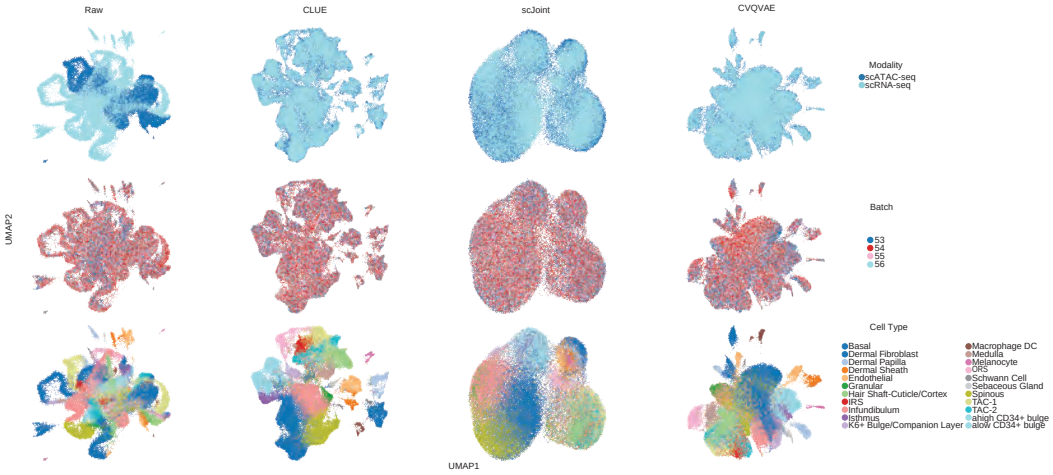

Figure 8: Embeddings data from raw data and different models of Ma dataset

Table 2: Overall score of different models on Ma dataset

Methods	NMI	Celltype ASW	CC	TC	Batch ASW	Graph connectivity	Score
PCA	0.582	0.491	0.661	0.484	0.964	0.730	0.671
scJoint	0.473	0.501	0.916	0.240	0.969	0.852	0.683
CLUE	**0.689**	**0.562**	**0.943**	0.477	**0.975**	**0.958**	0.787
CVQ-VAE	0.528	0.501	**0.943**	**0.846**	0.959	0.939	**0.802**

C Exploring variations of CVQVAE

When we designed our model, we considered several variants, including the removal of the vector quantization, and two ways to combine the latent representations z_1 and z_2 of the same cell and produce a single common representation z. The four variations we compare in this section are:

1. CVQVAE-ori: This is the main model described in previous sections.
2. CVAE: This is the original design of our model without the vector quantization. In this model, we do not combine the latent representation of the two encoders and use VAE.
3. CVQVAE-avg: This is the version of our model that averages the latent representations. Inspired by multiVI (2), in this version, we set the latent representation for the two modalities to be the average of the two encoder outputs and shared by both decoders.
4. CVQVAE-lin: This is the version of our model where we take a linear combination of the latent representations. Inspired by JAE (21), in this version, after we combine the outputs of the two encoders, we input them into a linear layer of the neural network to obtain the new embedding and use it as input to the decoder.

The comparison between these different variations can be found in Table 3. From Table 3, we see that

Table 3: Overall score for ablation test models

Methods	NMI	Celltype ASW	CC	TC	Batch ASW	Graph connectivity	Score
CVAE	0.474	0.488	0.625	0.823	0.788	0.855	0.690
CVQVAE-ori	**0.780**	**0.569**	**0.883**	**0.880**	0.883	**0.982**	**0.840**
CVQVAE-avg	0.778	0.554	0.842	0.578	0.901	0.976	0.788
CVQVAE-lin	0.767	0.555	0.848	0.879	**0.904**	0.970	0.832

the method considered in the previous sections, CVQVAE-ori, has the best overall results, followed by CVQVAE-lin. Therefore, we choose CVQVAE-ori as the final version of our model.

D Differential expression genes

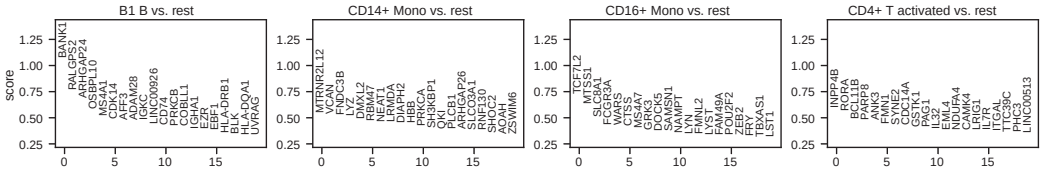

Figure 9: Differential expression test. For four different cell-type labels (B1 B, CD14+ Mono, CD16+ Mono, CD4+ T activated), we identify the genes that are most differentially expressed in the corresponding subset of cells.

E The advantages of VQ-VAE compared with VAE

In part 1, we will give a proof about how VQ-VAE can address the posterior collapse problem. Now we define the posterior collapse as:

Definition 1 (Posterior collapse). Given a probability model $p(\boldsymbol{x}, \boldsymbol{z}; \theta)$, a parameter value $\theta = \hat{\theta}$, and a dataset $\boldsymbol{x} = (x_1, \ldots, x_n)$, the posterior of the latent variables $\boldsymbol{z}$ collapses if

$$p(\boldsymbol{z} \mid \boldsymbol{x}; \hat{\theta}) \simeq p(\boldsymbol{z})$$

Proof If we intend to avoid this condition, we need to ensure that $\boldsymbol{x}, \hat{\theta}$ should always involve in the posterior calculation. Therefore, we consider such optimization process for the embedding:

$$z_{new}(f_e(x, \hat{\theta})) = argmin_i(||z_i^K - f_e(x, \hat{\theta})||^2)$$

where z_{new} represent the quantized vector, z_i^K represent a vector from discrete dependent variable space K, and $f_e(x, \hat{\theta})$ represent a function which can map the input data into the latent space, and we use neural networks here to fit this function. Here z_{new} will become the new input into the decoder model, which means that the decoder can be represented as:

$$D(z_{new}) = f_d(argmin_i(||z_i^K - f_e(x, \hat{\theta})||^2) = D(z_K, x, \hat{\theta})$$

Now consider the embedding generated by VAE model, which can be represented as:

$$z = f_\mu(x, \hat{\theta}) + s f_\sigma(x, \hat{\theta}) \text{ where } s \sim Normal(0, 1)$$

Therefore, in the case of using VQ-VAE, because z_i is an estimator of $f_e(x, \hat{\theta})$ under the squared error loss, the decoder can always decode the distribution constructed based on the input data. However, for the original VAE design, z is not an estimator. Based on this property, VQ-VAE is not affected by posterior collapse. That is why our model can address this problem.

In part 2, we will compare CVAE vs CVQVAE based on the loss curve analysis (Figure 10). In the loss curve of CVAE, the initial optimizing process is very fast till the blue star part. This indicates that the model was only optimized in the first few epochs, and the optimization of subsequent epochs was ignored due to the influence of posterior collapse, so the curve appeared very stable, which means that in the following training step the total loss will not change anymore. The results of latent space generated by CVAE are also more mixed.

To address this problem, we utilize VQ-VAE to replace the original VAE model. VQ-VAE discretized the embedding in the optimization process, and allowed us to select the most appropriate posterior vector according to the embedding database after obtaining the output of encoder. Therefore, randomness was added in the selection process. The increment of the complexity and flexibility of the embedding space allows it to contain more information. Our new loss function ensures that the representation of embedding space is optimized while the decoder reconstructs the input data. From the loss curve of CVQVAE, we can observe that the total loss keeps on reducing, which means that the optimizing process can work properly comparing with the VAE version.

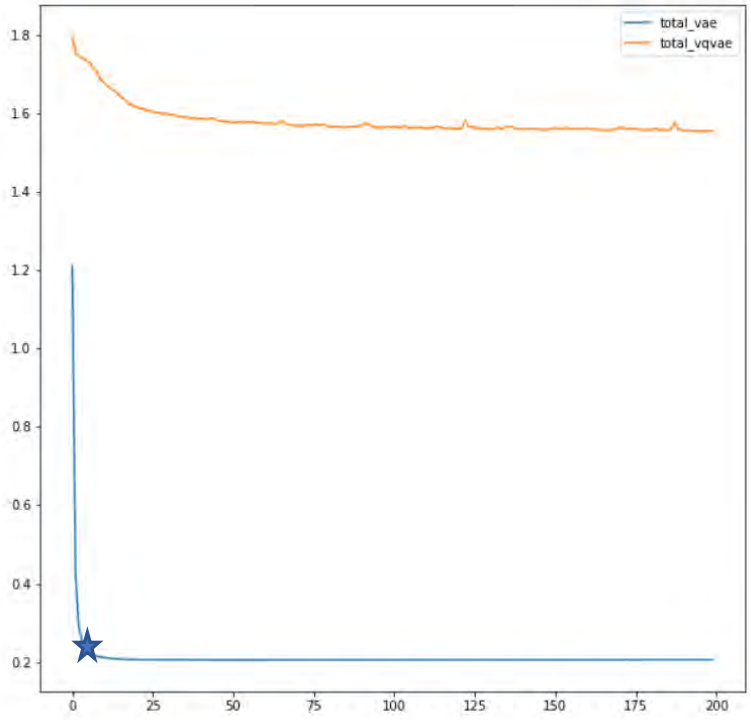

Figure 10: Training loss curves of CVAE (total_vae) & CVQVAE (total_vqvae)