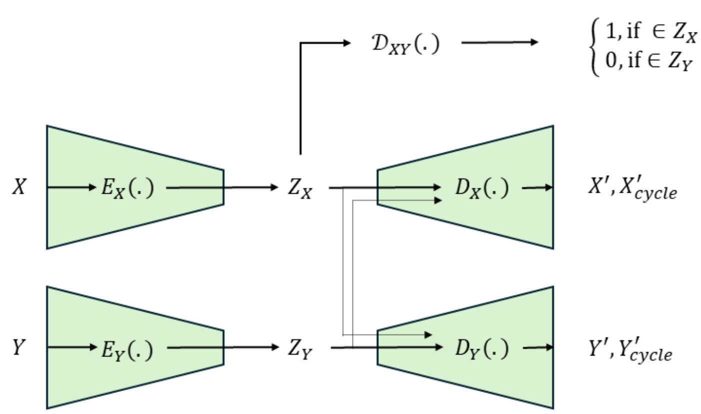
Cyclic multimodal autoencoder

We introduce a cycle-consistent autoencoder framework for integrating unpaired, multimodal single-cell RNA and ATAC data using a multiome-based bridge. We take inspiration from Seurat's bridge integration strategy, along with the cycleGAN framework (Zhu et al 2017). Our approach outperforms bridge integration based on benchmarking using a ground truth multiomic dataset.

The model first learns a shared latent representation for each assay contained in paired multiomic data, jointly embedding RNA (X_0) and ATAC (Y_0) into a unified space. By enforcing cycle consistency, the we encourage cross-modal translations to be mutually coherent, producing denoised, biologically meaningful reconstructions in both modalities.
After training on multiomic data, our model can map unpaired single-cell RNA or ATAC datasets (X, Y) into the shared latent space for downstream integrative analysis by feeding the assays through the pretrained encoders. This approach provides a strategy for leveraging limited paired data to integrate large unpaired datasets across modalities.



$$
\begin{aligned}
\mathcal{L}(E_X, E_Y, D_X, D_Y; X, Y) = \lambda_{recon} \cdot \big(MSE(D_X \circ E_X(X), X) + MSE(D_Y \circ E_Y(Y), Y)\big) \\
+\lambda_{cycle}\big(MSE(D_X \circ E_Y(Y), X) + MSE(D_Y \circ E_X(X), Y)\big) \\
+\lambda_{adv}(-\log \mathcal{D}_{XY}(E_Y(Y)))
\end{aligned}
$$

Figure 1 – graphical representation of the underlying model. X and Y are mapped to a joint latent space, $\mathcal{Z}$. Each set of embeddings, Z_X and Z_Y, are fed through both decoders $D_X(.), D_Y(.)$. This ensures the learned latent representation encodes the joint structure of the data.

Model details

We begin with two paired noisy datasets, X_0 and Y_0. We would like to learn two pseudo invertible mappings, $f: \mathcal{X} \to \mathcal{Z}$, and $g: \mathcal{Y} \to \mathcal{Z}$, to a latent space that encodes the joint structure of the data. By pseudo-invertible, we mean that each function f, g, has a reverse mapping $\hat{f}^{-1}, \hat{g}^{-1}$ that maps the latent embeddings in $\mathcal{Z}$, back to the original spaces, $\mathcal{X}, \mathcal{Y}$. To ensure that the latent embedding encodes both information about X_0 and Y_0, we employ a multimodal autoencoder framework that utilizes cycle consistent loss.

First, two encoders, $E_X(X)$, $E_Y(Y)$, map X and Y to their latent embeddings. We define the reconstruction loss as follows:

$$\mathcal{L}_{recon}(E_X, E_Y, D_X, D_Y; X, Y) = MSE\left(D_X\big(E_X(X)\big)\right) + MSE\left(D_Y\big(E_Y(Y)\big)\right)$$

This alone is not enough to ensure that the learned embeddings, $E_X(X)$, $E_Y(Y)$, encode the joint structure of X and Y. We use cycle consistency loss to ensure that the latent space is representative of the joint structure $[X, Y]$:

$$\mathcal{L}_{cycle}(E_X, E_Y, ; X, Y) = MSE\big(D_Y\big(E_X(X)\big), Y\big) + MSE\big(D_X\big(E_Y(Y)\big), X\big)$$

Note that during training, the gradient from $\mathcal{L}_{cycle}$ only corresponds to the parameters of the encoders E_X and E_Y. The decoders D_X and D_Y are only trained using their corresponding modalities. This encourages the encoders to mix X and Y in the latent space.

Additionally, we employ adversarial training to aid in the mixing of the latent space. We use a discriminator, $\mathfrak{D}_{XY}(Z)$, that classifies the modality of an embedding as X $(= 1)$ or Y $(= 0)$. The discriminator is trained using the following loss:

$$\mathcal{L}_{scrim}(\mathcal{D}_{XY}; X, Y) = -\log\big(\mathcal{D}_{XY}(E_X(X))\big) - \log\left(1 - \mathcal{D}_{XY}\big(E_Y(Y)\big)\right)$$

And the adversarial loss is defined as

$$\mathcal{L}_{adv}(E_Y; Y) = -\log(\mathcal{D}_{XY}\big(E_Y(Y)\big)$$

Finally, we define our overall loss function as follows:

$$\begin{aligned}\mathcal{L}(E_X, E_Y, D_X, D_Y; X, Y) = \lambda_{recon} \cdot \mathcal{L}_{recon}(E_X, E_Y, D_X, D_Y; X, Y) \\ +\lambda_{cycle} \cdot \mathcal{L}_{cycle}(E_X, E_Y, D_X, D_Y; X, Y) \\ +\lambda_{adv} \cdot \mathcal{L}_{adv}(E_Y; Y)\end{aligned}$$

Where $\lambda_{recon}, \lambda_{cycle}$, and λ_{adv} are user-specified weights, with default values of 1. A graphical overview of the model is shown in figure 1.