Computational Mapping of Single-Cell Measurements in Low Dimension Feature Spaces

BLAKE MASTERS

Senior Research Project Report
Presented to the Faculty
of California Polytechnic State University, San Luis Obispo,
in Partial Fulfillment of the Requirements for the Degree
of Bachelor of Science in Computer Science

Adviser: Dr. Borislav Hristov

June 2025

Abstract

We propose a framework for analyzing paired single-cell measurement datasets, where each entity is represented by two corresponding observations across distinct but related modalities. Our goal is to learn correspondences and classify entities based on a subset of labeled pairs (truths). We present two complementary approaches: a supervised proximity-based method and an unsupervised neural network model. The supervised method assigns labels by identifying the most proximal labeled pair in one modality and transferring the corresponding label and pairing to the other. In contrast, the unsupervised approach leverages an encoder architecture to learn a cross-modal translation function, allowing more flexible mappings between paired measurements. This approach operates directly on raw data to preserve signal structures that may be lost by normalization. Our methods aim to improve the robustness of cross-modal correspondence and classification, particularly in settings where proximity-based transfer may be insufficient due to volatility or distributional shifts.

1 Introduction

Rapid advancements in next generation sequencing techniques has led to the development of a myriad of experimental techniques that allow researchers to measure different properties of single cells. However, these techniques are typically destructive, preventing scientists from measuring multiple properties at the same time, i.e paired measurements. Having disjoint datasets that reside in different spaces and that need to be integrated (paired) in order to build a fuller picture of the molecular processes in the cell is a key problem in computational multi-omics. We note that the problem is present in numerous domains, including genomics, neuroimaging, and sensor systems, where each entity is observed through distinct modalities. Integrating these modalities to find a pairing is a particularly challenging task because modalities may differ significantly in structure, scale, or noise characteristics, posing unique challenges for correspondence learning and classification. A common goal in such settings is to leverage a subset of labeled pairs, known as ground-truth correspondences, with class labels to infer relationships and make predictions across the dataset.

Traditional approaches to cross-modal analysis often rely on strong assumptions about the alignment or similarity of modalities. However, when modalities exhibit volatility, distributional shifts, or structural heterogeneity, these methods may fail to capture meaningful correspondences. In particular, simple proximity-based transfer of labels from one modality to another can lead to misclassifications if the underlying relationship between modalities is non-linear or non-local.

To address these challenges, we propose a framework that combines two complementary strategies for establishing cross-modal correspondence and classification. First, we introduce a supervised proximity-based method that assigns labels by identifying the closest labeled counterpart in one modality and transferring its label to the corresponding entity in the other. This method is intuitive and computationally efficient but may struggle in settings with complex or unstable modality relationships. To overcome this, we also develop an unsupervised neural network model that learns a cross-modal translation function through an encoder architecture, enabling more flexible mappings across modalities. Together, these methods aim to improve the robustness and accuracy of correspondence and classification in paired measurement data.

2 Methodology

Throughout this paper, Cosine Similarity and Euclidean Distance are the main metrics to determine how 'similar' a measurement is to another, where a high cosine similarity corresponds to a highly similar measurement. This can be substituted for the lowest Euclidean distance and is used in an interchangeable fashion. In addition, testing is reported over a 30-epoch range, where 30 samples n in a set is the common minimum to consider a bell curve distribution unless otherwise mentioned. The general notation is provided in the subsection below, where individual sections may predefine the notation used in formulaic representations as well.

2.1 General Notation

- $M_1 \in \mathbb{R}^{n_1 \times m_1}$: measurement matrix 1 with n_1 cells and m_1 features.
- $M_2 \in \mathbb{R}^{n_2 \times m_2}$: measurement matrix 2 with n_2 cells and m_2 features.
- $L \in \mathbb{R}^n$: vector of ground-truth labels of length n, the target array length.
- P_k: the set of predicted pairs generated via the supervised proximity algorithm.
- $W_k \in \mathbb{R}^{n \times n}$: labeling decay matrix used to modulate pairing confidence.
- Set_1: contains 100% correctly paired truths.
- Set_2: contains 0% correctly paired truths (completely random).
- Set_3: contains 50% correctly paired truths; the remaining 50% are randomly assigned.

2.2 Scoring Metrics

Scoring is essential to determining efficacy of the mapping algorithms and transformed measurements. In order to establish robust and consistent results that are not driven by a bias in an individual scoring metric, we devise three separate metrics attached to multiple aspects of the predicted measurement.

2.2.1 Percentage Closer Scoring

First, our main scoring schema assigns a percentage indicating the number of points in the set that are closer to the truthful representation (higher similarity or lower distance) than the prediction point created (lower scores are better as it means that fewer points are closer to the correct match). By choosing a metric like this, the value of a random generation will approximate to 0.5, establishing a clear and intuitive threshold to an algorithm that performs better than random selection. An outline for the Percentage Closer Scoring (PCS) mathematical representation along with the algorithm is found below.

$$S = \frac{1}{|P_k|} \sum_{(i,j) \in P_k} \text{rate}(i, j) \tag{1}$$

Algorithm 1 Scoring Function

1: **Input:** Set of predicted pairs P_k, ground truth T, distance matrices
2: **Initialize:** Scores list S
3: **for** each pair (i, j) in P_k **do**
4: Compute the similarity rate based on distance metric (Cosine or Euclidean)
5: Append score to S
6: **end for**
7: Compute final average score $S = \frac{1}{|P_k|} \sum S$
8: **Output:** Final score

2.2.2 Cosine Similarity

Second, we use cosine similarity of measurement transformation compared to the original solution point. Cosine similarity, measured -1 to 1, serves as a proxy for a performance metric for prediction identity that aligns with the target identity. Thresholds around 0.8 are used for meta-cell prediction, informed by trends in similar domains and mass spectrometry.

2.2.3 Label Transfer Accuracy

In addition to a percentage distribution scoring metric, a set of labels is provided for the set. The labels contain the name of the tissue that each measurement corresponds to. By implementing labeling predictions, discussing label accuracy as a classification problem broadens the scope of the project and is discussed below.

Given the pairing approach of the supervised project, our labeling algorithm assigns the closest neighbor across measurement types as the predicted label and evaluates for accuracy. This is a supervised approach, and shown below.

$$\text{Accuracy} = \frac{1}{|P_k|} \sum_{(i,j) \in P_k} \mathbf{1}(\ell_i = \ell_j) \tag{2}$$

- P_k is the set of predicted index pairs (i, j), where i indexes points in dataset M_1 and j indexes points in dataset M_2
- ℓ_i is the ground-truth label associated with $M_1[i]$
- ℓ_j is the ground-truth label associated with $M_2[j]$
- $\mathbf{1}(\ell_i = \ell_j)$ is the indicator function, equal to 1 if the labels match, 0 otherwise
- $|P_k|$ is the total number of predicted pairs

In the case of neural architectures, encoders apply a decayed voting strategy to propagate labels by leveraging similarity to existing truths. For each prediction, the algorithm identifies the top-k most similar truths and assigns weighted votes to their labels. The vote weights decay exponentially with the neighbor's rank, combining both similarity and position to emphasize closer and more relevant matches.

If the top label receives a vote exceeding a confidence threshold, the prediction is added to the labeled set and used in subsequent rounds. This process iterates for a fixed number of steps or until no new labels are added. Because the method relies heavily on the proximity between predictions and known truths, it can introduce noise in soft label distributions. To mitigate this, we report hard labels by selecting the highest-voted label per prediction as the final output. The algorithm is described below.

Algorithm 2 Decayed Voting Label Propagation

Require:
- Prediction vectors: $\mathcal{P} = \{P_1, \ldots, P_n\}$
- Known vectors (truths): $\mathcal{T} = \{T_1, \ldots, T_m\}$
- Corresponding labels: $\mathcal{L} = \{\ell_1, \ldots, \ell_m\}$
- Parameters: number of neighbors k, decay rate λ, confidence threshold τ, max steps S

Ensure: Soft label scores, hard labels, updated truths and labels

for each step up to S **do**
  Initialize new labels and truths to be added
  for each prediction P **do**
    Compute similarity between P and each truth T
    Find top-k most similar truths
    For each of the k neighbors, compute:

$$\text{weight} = \text{similarity} \times e^{-\lambda \cdot \text{rank}}$$

    Add these weights to a vote for each label
    Normalize the vote to get a probability distribution
    Assign the most voted label (hard label)
    if top vote $\geq \tau$ **then**
      Add P to the list of known truths with its predicted label
    end if
  end for
  if no new labels were added **then**
    Stop early
  end if
  Update the known truths and labels
end for
return soft label scores, hard labels, updated truths, updated labels

2.3 Algorithmic Approaches

We operate under the assumption that a subset of labeled truths is available, with the goal of performing general cell-to-cell mapping. We devise two algorithms- a supervised and unsupervised. The supervised approach focuses on expanding the known label space by leveraging existing truths through direct classification. In contrast, the encoder-based method generates new representations in the feature space, aiming not just to classify but to model meaningful transformations of the data. This approach shifts the focus from pure classification to learning functional mappings that can generalize reliably across the dataset.

2.3.1 Supervised Pairing Algorithm

The supervised algorithm is designed to construct a set of mapped pairs between measurement spaces M_1 and M_2. For each measurement i in the truth set from M_1, the algorithm identifies its most similar counterpart j from the truths in M_2, forming a pair $\{i, j\}$, provided neither has been used in a previous pairing.

During development, we tested the strategy of adding these new pairs to the original truth set. However, this approach was found to degrade performance compared to excluding them. While including new pairings may offer short-term gains in low-density truth sets by reinforcing structure, it introduces compounding errors. These errors tend to grow in a slow exponential pattern, in contrast to the more bounded, linear degradation observed when new pairings are excluded. Below we lay out the algorithm blueprint.

The algorithm offers the potential to use an average highest similarity point, but this proved significantly worse than using the closest value during testing. Either the Closest (recommended) or Average strategy across the truth set can be selected, as outlined below:

New pairs are iteratively added based on the following selection criteria:

1. **Closest:** Assigns a pair (i_1, j_1) by finding the indexes with the minimum distance in M_1 and M_2.
2. **Average:** Computes the average distance from all existing pairs and assigns the closest match:

$$\frac{\sum_{i \in M_1} D(i)}{n}, \quad \frac{\sum_{j \in M_2} D(j)}{n} \tag{3}$$

The algorithm proceeds as follows:

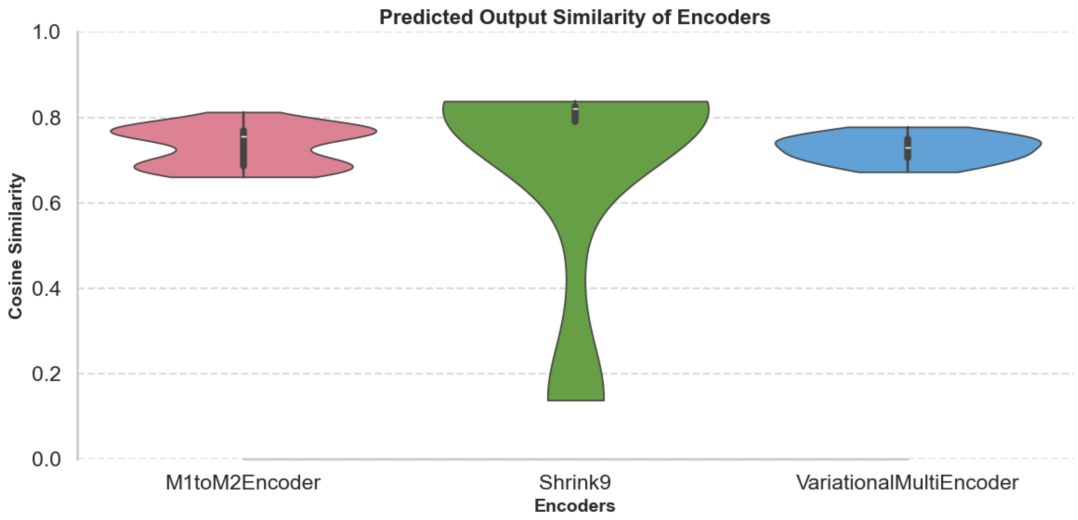
Algorithm 3 Proximal Pairing Algorithm

Input: Initial truth set T, distance matrices M_1, M_2
Initialize: Mark paired indices in M_1 and M_2
Set $P \leftarrow T$
while New pairs are found **do**
    Determine new pairs using selected heuristic:
    if Metric = Closest **then**
        Select (i_1, j_1) minimizing $D(i)$ and $D(j)$
    else if Metric = Average **then**
        Compute mean distance from all current pairs and assign closest
    end if
    Mark selected indices and update P
end while
Output: Set of new pairings P_k

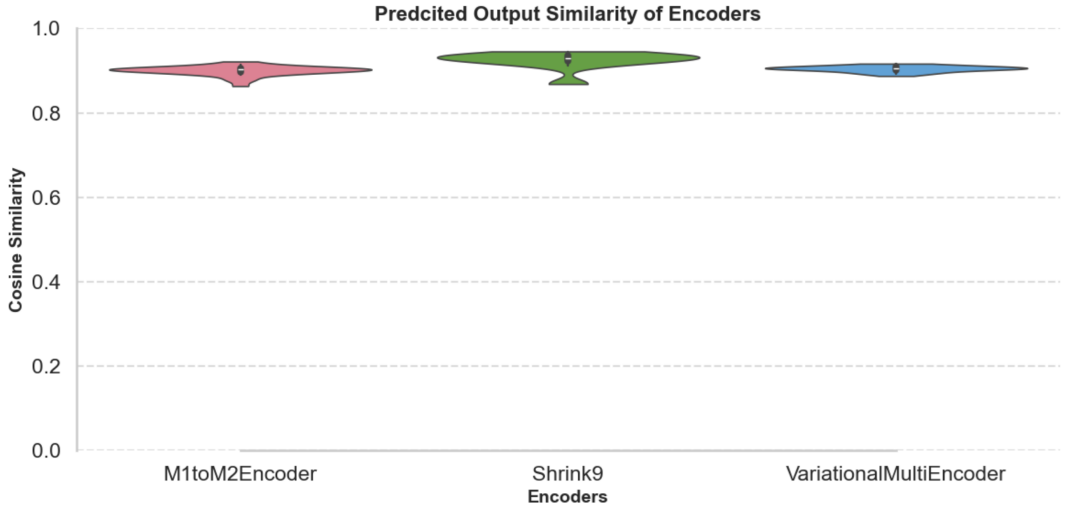
2.3.2 Unsupervised Solution - Encoding Measurements

The next stage of the project focused on developing an unsupervised approach using raw measurement data. In contrast to common practices, which often normalize or preprocess inputs to standardize training, our method deliberately retains raw variability. This decision preserves biologically meaningful differences in the data, relying on the encoder to learn which features contribute to accurate mappings while filtering out noise.

Our chosen unsupervised strategy uses neural networks to map measurements from M_1 to M_2 by learning transformations that expand and contract the feature space. Among tested architectures, shallow single-layer transformations offered competitive performance, while deeper encoders with 9–10 layers achieved comparable results but were more susceptible to local minima. Included below is a sample run using standard parameters.



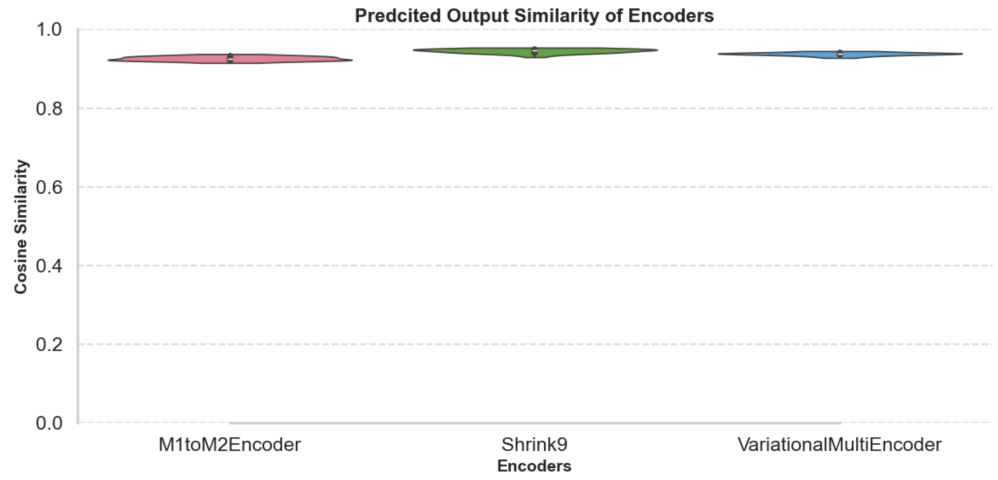
(a) Panel A: M1 → M2 mappings



(b) Panel B: M2 → M1 mappings

Figure 1: Mapped measurement similarity score evaluation to truth across top-performing encoders.

Our figure displays the generalized trend where deeper network architectures (Shrink9) offer the strongest potential results with the highest degree of variance at our standardized set values in terms of cosine similarity. Despite this higher similarity, the reported percentage closer metric appears worse, implying some degree of contradiction between the evaluation metrics used. Seen below in an example of such on a larger truth set (100 Set 1) to exemplify repeatability of this result



(a) Panel A: Cosine similarity of M2 → M1 mapped predictions

(b) Panel B: Percentage Closer Score (PCS) for M2 → M1 mappings

Figure 2: M2 → M1 mapped prediction similarity scores and PCS evaluation over 100 Set 1 Truths. Shrink9 has the highest similarity and worst PCS.

2.4 Encoder Nomenclature

`M1M2` : A direct PReLU transformation from size n_1 to n_2 (Single Layer transformation)

`Shrink` : Start from n_1 neurons, immediately expand to a large quantity of neurons and reduce to a size n_2 over x layers

`Diamond` : Rhombus architecture starting from n_1 neurons and expanding/contracting in a rhombus format over x layers

`Pyramid` : Start from n_1 neurons, immediately shrinks to a small quantity of neurons and expands to a large count before sizing to n_2 over x layers

`VariationalMultiEncoder (VME)` : Wraps multiple base encoders and uses a learned gate (Z score encoding) to weigh contributions per sample. See 2.4.1

All encoder architectures were trained using the Adam optimizer, selected for its general reliability and stable convergence during preliminary testing. Although a brief comparison of optimization methods was conducted, initial trials indicated that Adam provided consistent results and the highest similarity results. While testing, all sizable architectures (Shrink, Diamond, Pyramid) performed best around 9–10 layers with noticeable falloffs in performance at 15 layers, and their layer can be found as an integer following the name in figures.

2.4.1 VME Architecture

Given the complexity of a new encoder design, it has been given a designated section to discuss architecture and design. The `VariationalMultiEncoder` wraps multiple base encoders and employs a learned gating mechanism to dynamically weight their contributions for each input sample, akin to a Transformer architecture.

The gating network receives a z-score normalized version of each input, ensuring that expert selection is based on the relative pattern of features rather than their absolute magnitudes. This sample normalization encourages encoder specialization across different input profiles.

Each expert processes the raw input independently, and their outputs are combined using the softmax-derived weights from the gate. To evaluate the diversity of encoder usage across samples, we introduce a utility function called `volatility_index`, which calculates the standard deviation of encoder assignments from input-output pair indices. Higher volatility values may suggest broader encoder coverage across different samples. A walkthrough of the encoder is below.

VME: Let $\mathbf{x}_i \in \mathbb{R}^d$ denote the i-th input vector in a batch of size B. The encoder consists of N independent base encoders $\{E_1, E_2, \ldots, E_N\}$, each mapping inputs from $\mathbb{R}^d$ to a latent space $\mathbb{R}^k$.

- Normalize each input sample:

$$\tilde{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i) + \epsilon}$$

  where $\mu(\mathbf{x}_i)$ and $\sigma(\mathbf{x}_i)$ are the mean and standard deviation of $\mathbf{x}_i$'s features.

- Compute gate logits and softmax weights:

$$\mathbf{w}_i = \text{softmax}(G(\tilde{\mathbf{x}}_i)) \in \mathbb{R}^N$$

  where $G(\cdot)$ is the learned gating network.

- Let $\mathbf{z}_{ij} = E_j(\mathbf{x}_i)$ be the output of expert j on input $\mathbf{x}_i$.

- The final latent representation is the weighted sum of expert outputs:

$$\mathbf{z}_i = \sum_{j=1}^{N} w_{ij} \cdot \mathbf{z}_{ij}$$

Here, $\mathbf{z}_i \in \mathbb{R}^k$ is the final output of the VariationalMultiEncoder for input $\mathbf{x}_i$.

2.5 Activation Functions

The selection of activation function for the project came from four pre-researched options, with the project ultimately settling on the Parameterized Rectified Linear Unit (PReLU) after testing across different encoders.

`Tanh`: A smooth, bounded activation that maps inputs to the range $(-1, 1)$. Often used in earlier architectures, it compresses extreme values but can saturate and suffer from vanishing gradients.

`PReLU`: An extension of ReLU where the negative slope is a learnable parameter, allowing the model to adaptively determine how much negative input should pass through.

`ReLU`: The most widely used modern activation. It outputs zero for negative values and identity for positives, introducing sparsity and reducing computational overhead.

`LLReLU (Learnable Leaky Rectified Linear Unit)`: A custom activation that mimics leaky ReLU behavior but learns the leakage rate during training. It improves flexibility over fixed-slope variants by tuning the negative slope to the data. In practice this proved unreliable due to local minima.

The PReLU function used is defined as:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \quad \text{where } \alpha \text{ is learned}$$

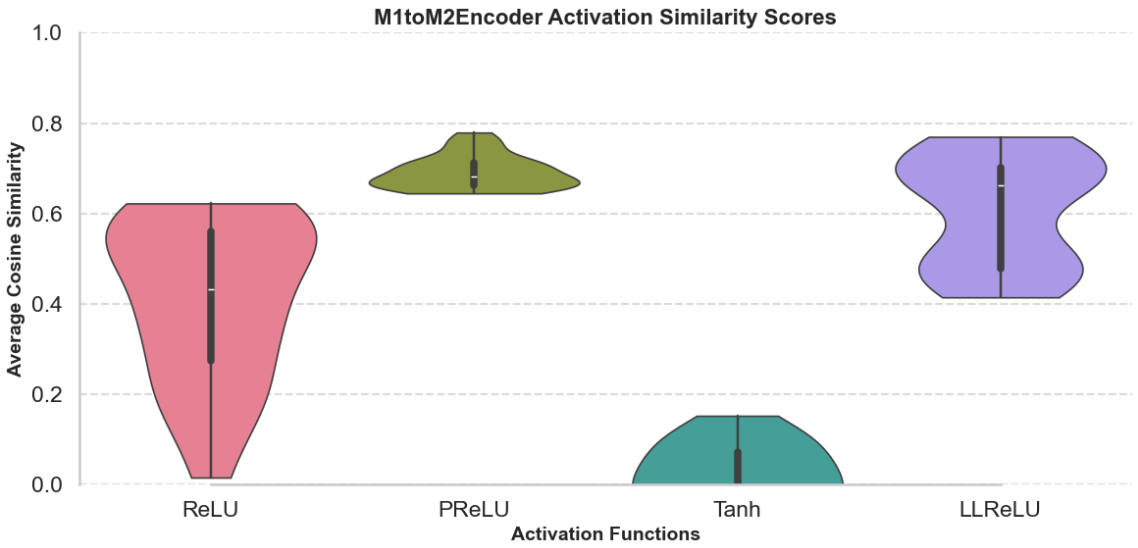


Figure 3: Distribution of predicted sample similarity scores for M1toM2Encoder outputs over various considered activation functions. PReLU shows the highest average and tightest distribution over the set.

Given a singular generated truth set, multiple activation functions were selected for potential usage cases. Given that raw data is passed to initial activation, we note that activation functions containing zeroing for negatives such as ReLU perform worse. The curvature given by hyperbolic tangent yields results in the realm of random, and thus is not a consideration. The Learnable function shows potential in the maximal results, but has a wider and lower distribution than PReLU. It may succeed with a well-developed architecture.

2.6 Loss Function

We implemented a simple similarity-based loss function designed to encourage alignment between output vectors and a fixed target matrix. The base module computes the average cosine similarity between paired vectors and penalizes the model proportionally to their dissimilarity. When specific index pairs are provided, the loss is computed only across those pairs; otherwise, it defaults to comparing the first N entries in the output and target matrices. The walkthrough can be found below.

The `base_sim_loss` function computes a cosine similarity-based loss between model outputs and a fixed target matrix. It supports both pairwise matching and a default full-alignment mode.

Notation:

- $$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \quad \text{where } \alpha \text{ is learned}$$

- $\mathbf{O} \in \mathbb{R}^{n \times d}$: Output matrix from the encoder, where n is the number of output vectors and d is the latent dimensionality.

- $\mathbf{T} \in \mathbb{R}^{m \times d}$: Target matrix of reference embeddings.
- $P = \{(i, j)\}$: Optional set of index pairs indicating matched output and target indices.
- $\mathbf{o}_i$: The i-th row of $\mathbf{O}$ (an output vector).
- $\mathbf{t}_j$: The j-th row of $\mathbf{T}$ (a target vector).
- $\cos(\mathbf{o}, \mathbf{t})$: Cosine similarity between vectors $\mathbf{o}$ and $\mathbf{t}$.

Pairwise Loss (if P is provided):

$$\mathcal{L} = 1 - \frac{1}{|P|} \sum_{(i,j) \in P} \cos(\mathbf{o}_i, \mathbf{t}_j)$$

Fallback Average Loss (if P is not provided): Let $N = \min(n, m)$ be the number of aligned vectors from both matrices.

$$\mathcal{L}_{\text{avg}} = 1 - \frac{1}{N} \sum_{k=1}^{N} \cos(\mathbf{o}_k, \mathbf{t}_k)$$

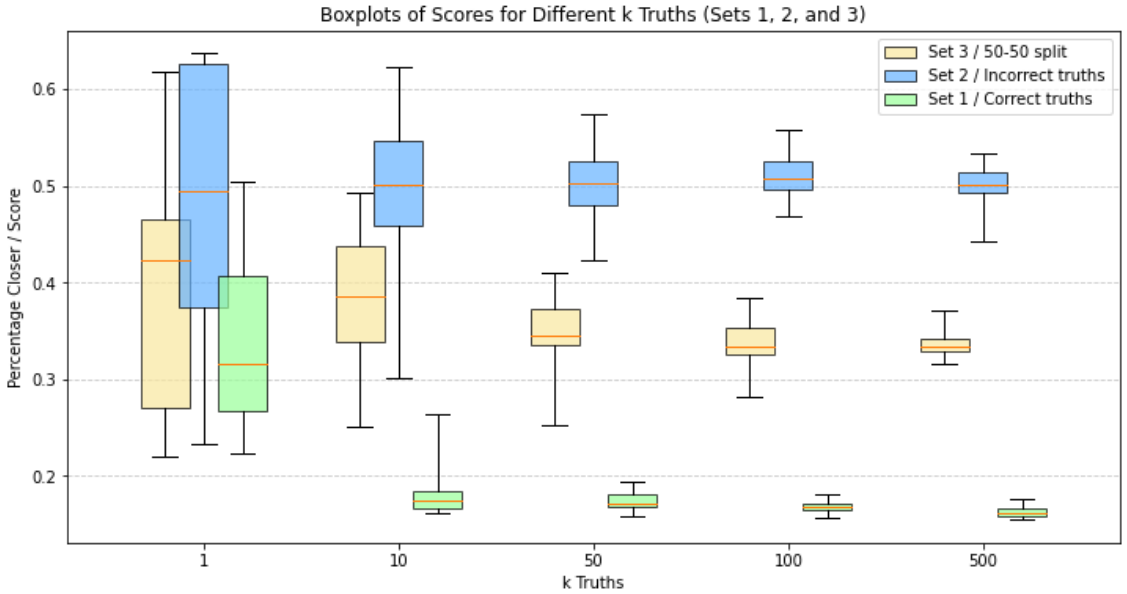
Initial experiments also explored penalty-based objectives aimed at maximizing the average similarity across all mapped samples. However, these alternative loss functions underperformed in preliminary tests and were excluded from further development. While there is potential to improve performance with more complex or dynamic loss formulations, this work prioritizes a reliable and interpretable baseline.

3 Empirical Results

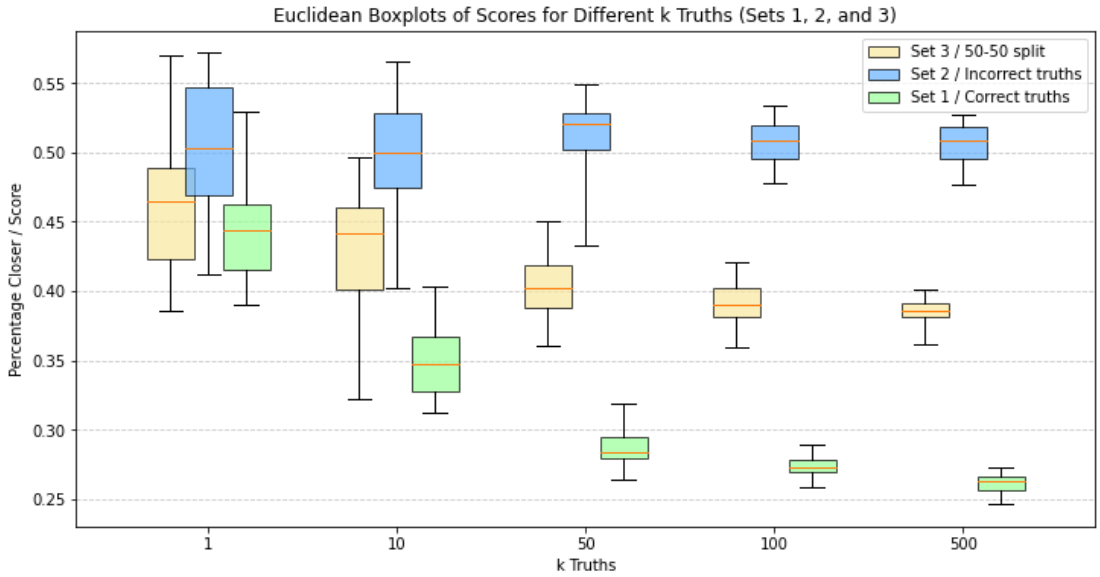
This section presents a comprehensive evaluation of both the supervised and encoder-based approaches. We use data from Rithambra Singh's lab, specifically the scGEM co-assay dataset, which simultaneously profiles gene expression and DNA methylation in 1,047 human somatic cells. In this case, we know the correct correspondence between all cells. To test performance, we randomly split the data into three types of sets: one with entirely correct pairings and labels (Set 1), one with entirely incorrect pair/label assignments (Set 2), and one with half correct and half incorrect pair/label assignments (Set 3). These sets are generated at runtime to evaluate performance across varying subsets of the data, with a single set used consistently across tests to measure performance differences.

The goal is to assess performance under varying levels of label quality, simulating the imperfect conditions often found in real-world datasets. By including a fully correct set, we establish a baseline for performance under ideal conditions. In contrast, the inclusion of partially or entirely incorrect truth sets enables an assessment of each method's resilience to noise and its capacity to generalize under uncertainty. The fully incorrect set also serves as a control, providing a reference point for random or non-informative behavior.

3.1 Supervised Close-Score Evaluation



(a) Panel A: Cosine Similarity boxplots across varying k-truth values



(b) Panel B: Euclidean Distance boxplots across varying k-truth values

Figure 4: Percentage Scoring metric evaluated across different truth and set values. Panel A shows Cosine Similarity and Panel B shows Euclidean Distance. Set 1 and Set 3 show percentage declines with larger sets.

A general improvement trend is observed on datasets containing accurate truth labels, whereas sets with a high proportion of incorrect truths exhibit more random or inconsistent behavior. In particular, performance improvements on Set 3 are minimal compared to the substantial gains seen on Set 1. This highlights a critical limitation of the supervised approach: its sensitivity to noisy or incorrect labels. In fact, under these conditions, the model treats equivalently about 1 percent of Set 3 values as it would Set 1 values, underscoring its inability to generalize effectively in the presence of noise in the provided truth set.

3.1.1 Supervised Labeling Evaluation

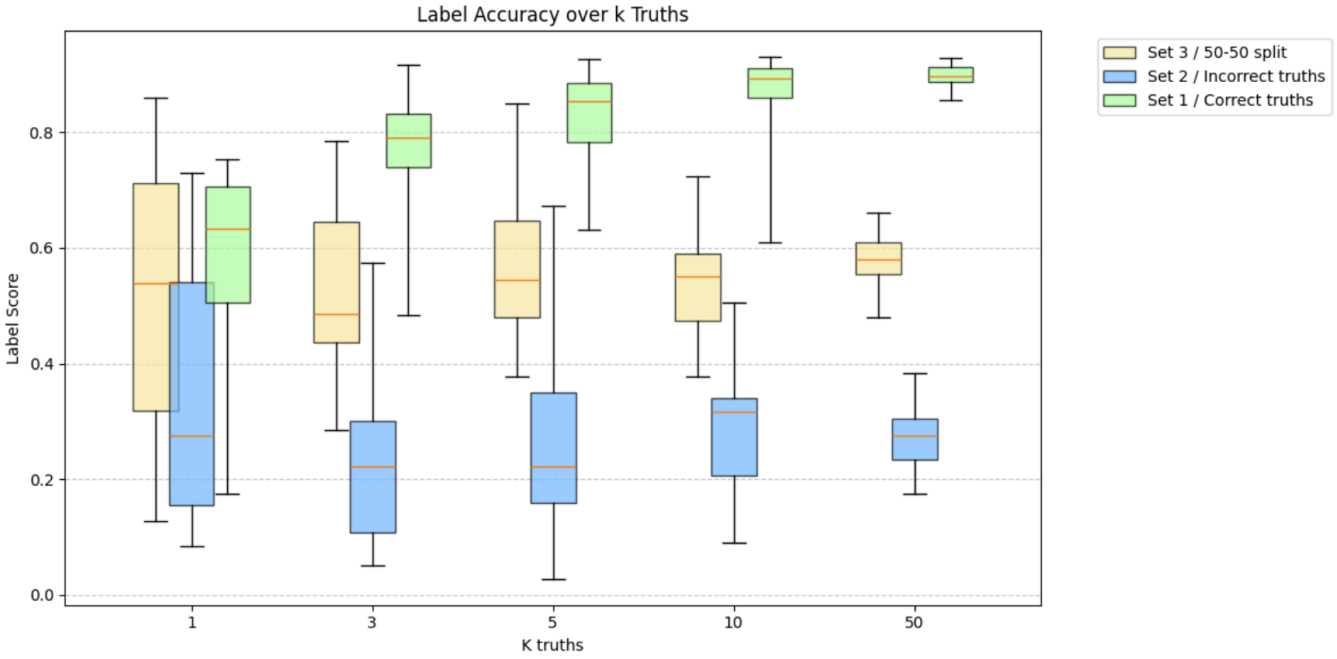
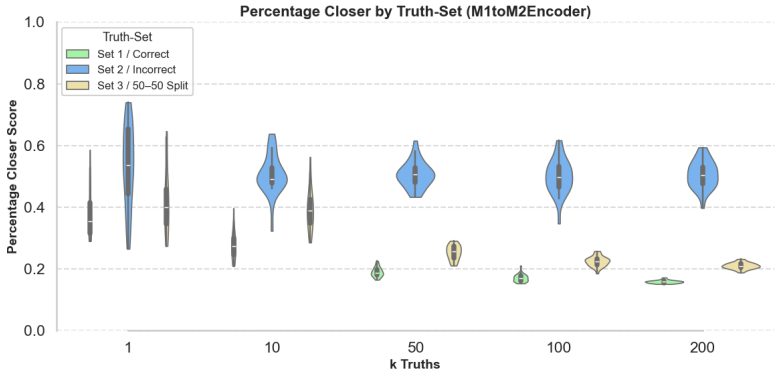


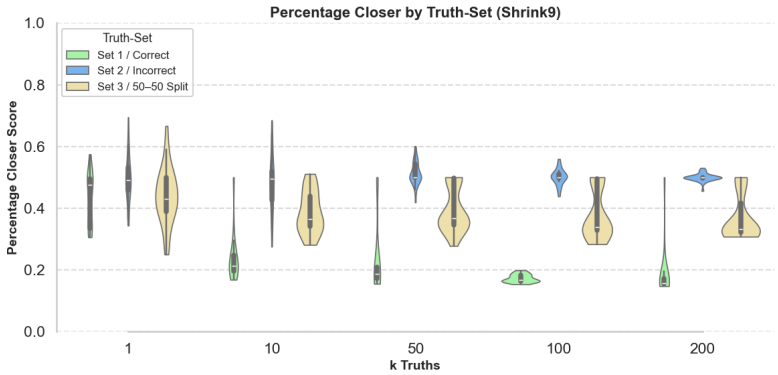
Figure 5: Boxplot of labeling accuracy from the supervised proximity-pairing algorithm across all truth sets. A clear improvement is observed on Set 1 (high-quality truths), while Set 3 (noisy) shows only marginal gains.

The improvement in labeling accuracy with increasing Set 1 truth size follows a logarithmic trend, ultimately reaching around 92 percent accuracy. Although less visually apparent, Set 3 also shows slight improvement. This aligns with the trend observed in the PCS figure, where 1 percent of the set size could be Set 1 truths to approximate to the same result. These results highlight the model's strong reliance on high-quality truths and its limited ability to generalize in the presence of noisy or incorrect labels, making it a complicated choice for imperfect information. Below we will look into models that have better performance under uncertain truths in unsupervised learning territory.

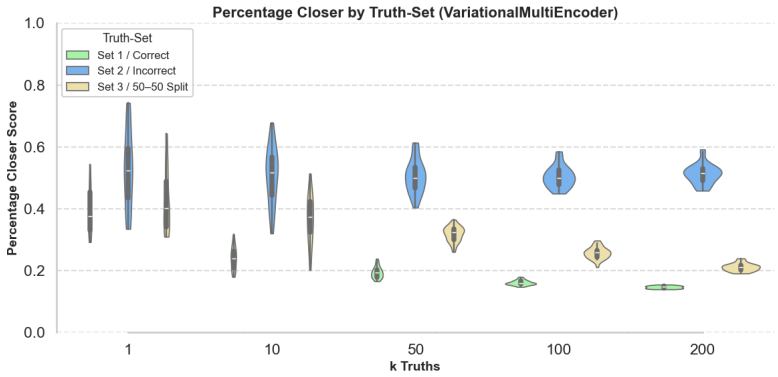
3.2 Encoder PCS Evaluation



(a) Panel A: Percentage Closer Score by Truth-Set (M1toM2Encoder)



(b) Panel B: Percentage Closer Score by Truth-Set (Shrink9)

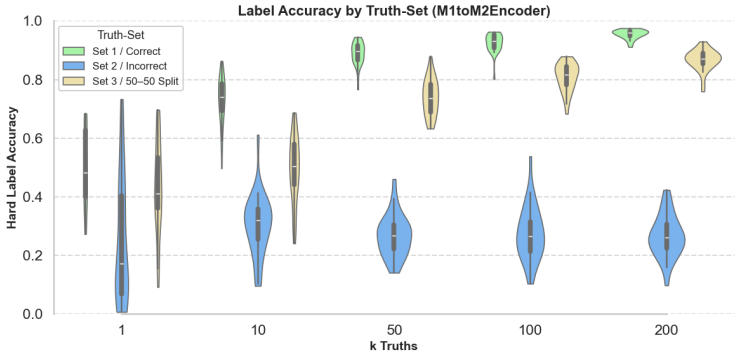


(c) Panel C: Percentage Closer Score by Truth-Set (VariationalMultiEncoder)

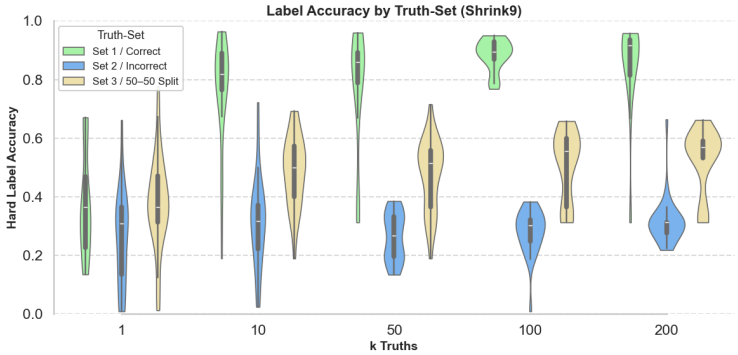
Figure 6: Top selected encoders and the PCS evaluation of their mapping predictions over a range of truth set sizes and types [M1toM2Encoder : Panel A, Shrink9 : Panel B, VME : Panel C]. Shrink9 performs noticeably worse on noisy results.

As the size of the truth set increases, both the M1toM2 and VME approaches converge toward similar performance limits on Sets 1 and 3. This convergence appears to be architecture-related, as VME encoders are based on the M1toM2 structure but include additional mechanisms to suppress noise. The consistent performance ceiling across encoders and the supervised method suggests that the variance is primarily driven by the dataset itself, rather than the model. Despite similar upper bounds, the models differ significantly in how they handle noise: while VME produces slightly more stable (less dispersed) outputs, both single-layer transformation encoders perform surprisingly well under high-noise conditions. This supports the idea that meaningful measurement transformations can be learned even from noisy data.

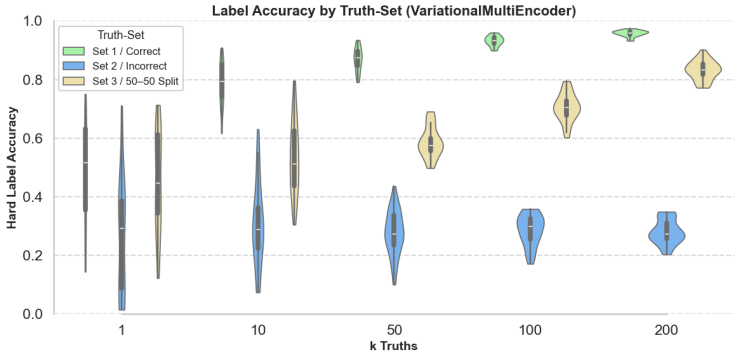
3.2.1 Encoder Labeling Evaluation



(a) Panel A: Hard Label Accuracy by Truth-Set (M1toM2Encoder)



(b) Panel B: Hard Label Accuracy by Truth-Set (Shrink9)



(c) Panel C: Hard Label Accuracy by Truth-Set (VariationalMultiEncoder)

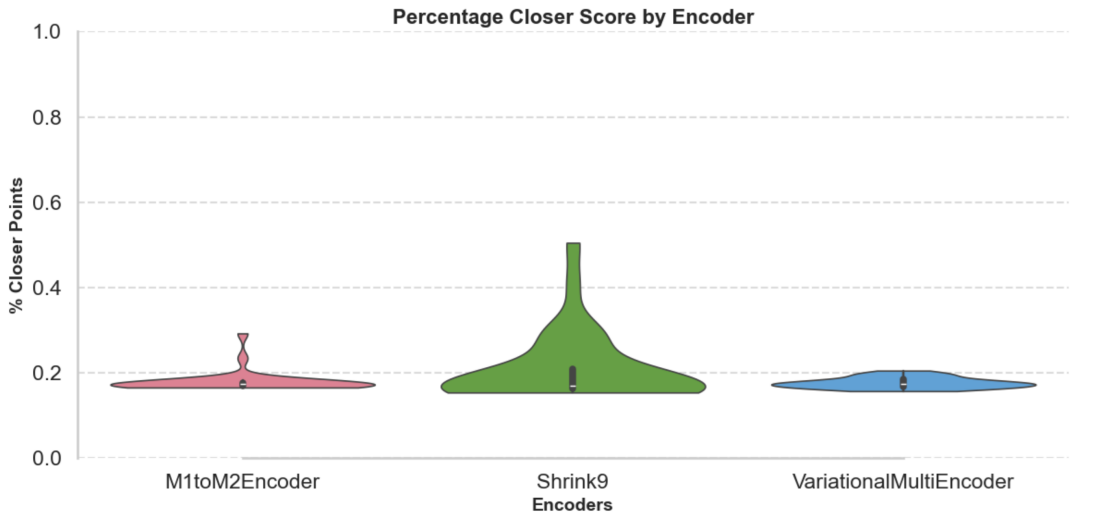
Figure 7: Top selected encoders and the Hard Label Accuracy evaluation of their mapping predictions over a range of truth set sizes and types [M1toM2Encoder : Panel A, Shrink9 : Panel B, VME : Panel C]. Shrink9 performs noticeably worse on noisy results.

As the truth set size increases, both the M1toM2 and VME approaches converge toward similar accuracy limits on Sets 1 and 3, based on hard label assignments using the weighted neighbor vote. This convergence appears to reflect underlying architectural similarities again, as VME builds upon the M1toM2 encoder with added mechanisms to suppress the influence of noise during label propagation. The fact that all models approach the same performance ceiling suggests that variability in results is largely attributable to the quality of the dataset rather than the model itself.

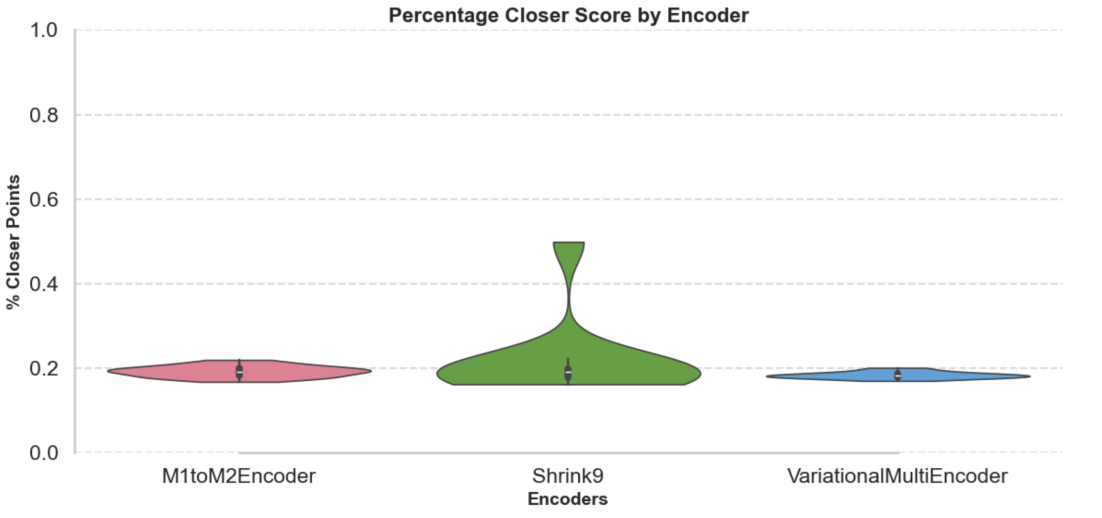
Despite similar limits, the methods differ notably in how they respond to noise. Shrink9 produces noticeably less consistent label accuracy across noisy conditions with essentially no improvements, but both M1toM2 and VME demonstrate strong performance under high-noise settings when using hard label voting. This indicates that the

encoders are capable of learning meaningful transformations that align predictions with high-confidence label regions, even when truth sets are partially incorrect, given the similarity based scaling factor in label propagation.

3.3 Reported Encoder performance



(a) Panel A: Percentage Closer Score by Encoder (M1 → M2)



(b) Panel B: Percentage Closer Score by Encoder (M2 → M1)

Figure 8: Mapped measurement PCS (Section 2.2.1) evaluated across top performing encoders. Panel A shows M1 → M2 mapping results and Panel B shows M2 → M1 mapping results. Encoder descriptions can be found in Section 2.4.

Scoring for the supervised proximity algorithm is reported as the average of both M1 → M2 and M2 → M1 directions, which differs from the encoder-based evaluations shown above. The encoder results do not exhibit significant variation across the evaluated sets and are therefore reported independently. Notably, there is a substantial difference in distribution patterns, which will be examined below.

4 Discussion

Our analysis shows that both supervised and unsupervised approaches perform well when trained on large, high-quality truth sets (Set 1). However, the substantial variance observed on partially correct sets, particularly Set 3, underscores the importance of selecting encoder architectures that are robust to label noise. While this report focuses on M1toM2, Shrink9, and VME, we find that most architectures tend to converge near the performance ceiling when sufficient clean labels are available.

Notably, the maximum achievable label accuracy is slightly higher for the encoder-based methods pushing 1 or 2 percentage points higher than the supervised proximity algorithm. We speculate that this marginal gain is due to the encoders' use of a meta-cell voting strategy, which improves upon the limitations of single-point similarity by aggregating weighted neighbor information.

These findings support the idea that meaningful transformations of raw measurements across feature spaces can be reliably learned across different encoder architectures and varying levels of label quality. Looking forward, further exploration of how these transformations relate to underlying cellular structure may offer insights into latent measurement representations and biologically relevant manipulations into how known effects of one measurement may affect another.

5 Future Developments

This project presents several promising directions for future research and refinement. One key area involves revisiting the training methodology of the VariationalMultiEncoder (VME). Rather than training each encoder on the full input dataset, future work could explore partitioning input samples among encoders based on their initial z-score distributions, or alternative statistical metrics, prior to applying softmax-based gating. This pre-assignment strategy may yield greater encoder specialization and improve mapping accuracy.

Additionally, the current design focuses on one-way, direct encoding of measurements. Expanding this to incorporate multi-modal data jointly within a shared latent space could uncover richer internal structure and interdependencies across measurement types. This shared latent representation may serve as the foundation for a future encoder-decoder architecture, enabling more expressive mappings and potential reconstruction capabilities beyond a single direct transformation.

Lastly, further enhancements to supervised approaches may include the integration of a weight-based voting mechanism similar to that used in encoder label propagation for more robust label prediction. These developments represent incremental yet impactful opportunities to improve both the flexibility and interpretability of the system.

Code and Data Availability

All code and data are available at: `https://github.com/CalPoly-MLBio/2025_Blake_SeniorPoject`

Acknowledgments

We use data from Rithambra Singh's lab, specifically the scGEM co-assay dataset, which simultaneously profiles gene expression and DNA methylation in 1,047 human somatic cells.