# CytoAutoCluster: Semi-Supervised Deep Approach for Cytometry Data Analysis

## 1. Introduction and Background

### 1.1 Objectives

- Perform data preprocessing and exploratory analysis.
- Develop and test semi-supervised clustering algorithms for high-dimensional mass cytometry data.
- Evaluate clustering results against manually gated clusters.

### 1.2 Overview of CyTOF Data

Mass cytometry (CyTOF) is a state-of-the-art technology for high-throughput single-cell analysis. It enables the measurement of over 30 markers per cell and is widely used in immune profiling and biomarker discovery. However, the high dimensionality of CyTOF data introduces analytical challenges. This project explores semi-supervised clustering algorithms, including techniques like **autoencoder-based feature learning**, to address these challenges.

## 2. Background and Motivation

### 2.1 Cytometry Data Analysis

Cytometry measures cell properties, such as size, complexity, and protein expression, and is vital in immunology, cancer research, and disease diagnostics.

### 2.2 Challenges

1. High dimensionality.
2. Noise in data.
3. Limited labeled data availability.

### 2.3 Motivation for CytoAutoCluster

CytoAutoCluster leverages semi-supervised learning to overcome these challenges, improving cluster precision and interpretability.

## 3. Dataset Description

### 3.1 Properties

| **Characteristic** | **Details** |
|---|---|
| **Cells (n)** | 265,627 |
| **Markers (p)** | 32 |
| **Manually Labeled Cells** | 39% (104,184 cells) |
| **Clusters (k)** | 14 |
| **Unlabeled Cells** | 61%(161,443 cells) |

### 3.2 Source

The dataset, **Levine_32dim**, is sourced from Levine et al. (2015). It is a benchmark dataset for clustering algorithms and is publicly available on platforms like Cytobank.

### 3.3 Marker Details

- **Markers Used for Manual Gating**: CD3, CD4, CD7, CD8, HLA-DR, CD123, CD235a/b, etc.
- **Other Markers**: CD10, CD45RA, CD56, among others.

---

## 4. Methodology

### 4.1 Data Preprocessing

The preprocessing pipeline included:

- **Normalization:** Standardized feature distributions using StandardScaler**.**
- **Exploratory Analysis:** Visualized marker distributions through histograms and density plots.
- **Data Masking:** Simulated partially labeled data for real-world clustering scenarios.

**Code Snippet (Scaling)**:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
```

### 4.2 Data Splitting

The labeled data was split into training and testing sets using a 70-30 ratio.

**Code Snippet**:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x_labeled, y_labeled, test_size=0.3, random_state=42)
```

### 4.3 Data Exploration

#### 4.3.1 Cluster Distribution Analysis

Include a brief description of the cluster distribution, followed by the image:

> The distribution of clusters in the label column reveals an imbalance across categories. Cluster **7.0** is the most frequent, contributing to **9.9%** of the data. Smaller clusters, such as **5.0**, **12.0**, and **14.0**, account for less than **0.5%** of the dataset each.

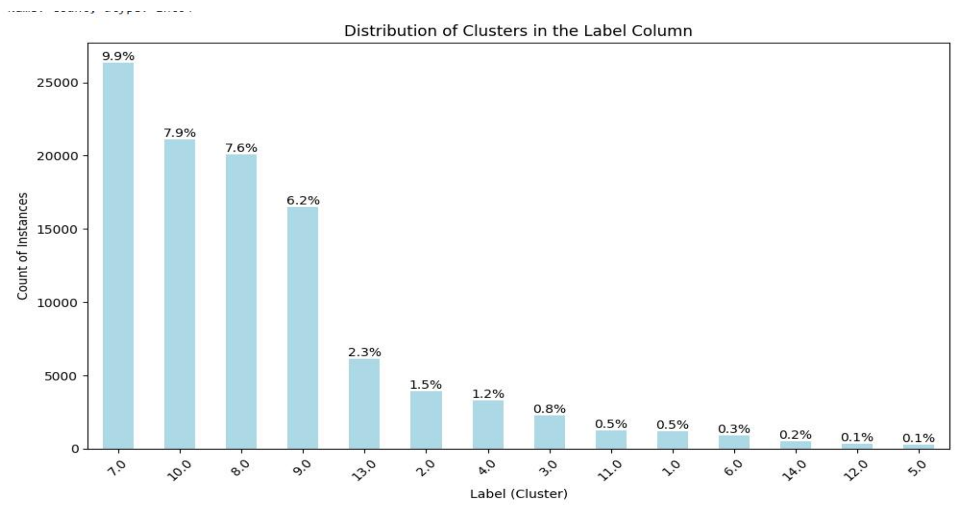

**Fig1:** *Cluster Distribution Analysis*

#### 4.3.2 Null Value Analysis

Insert the null vs. non-null analysis along with the corresponding image:

> The label column contains **39.2%** null values. Effective handling of these missing values is crucial for ensuring model accuracy and consistency. **60.8%** of the data is non-null, which can directly contribute to modeling efforts.

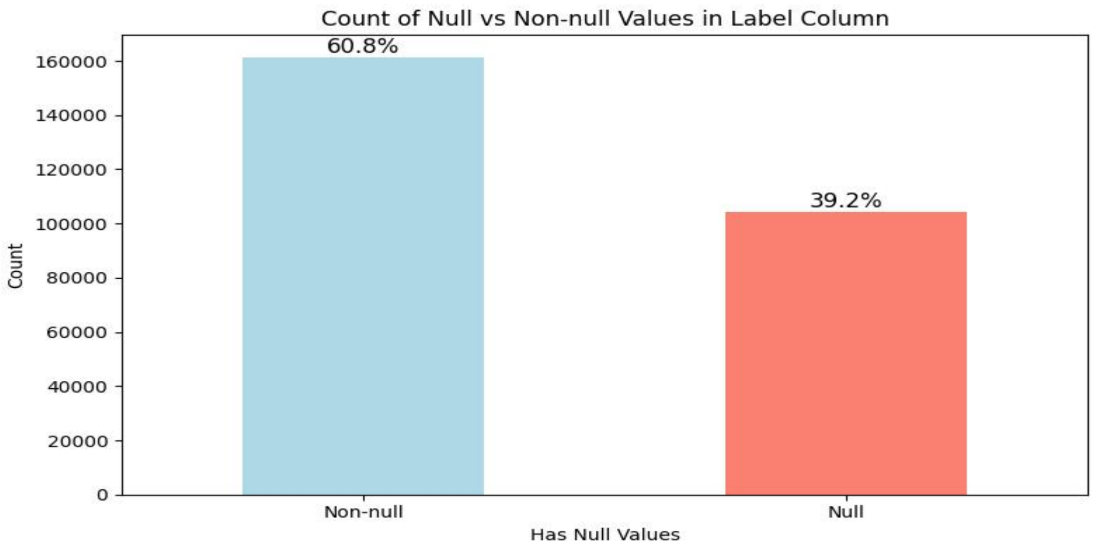

**Fig2:** *Null Vs Non-Null Value Analysis*

### 4.4 Clustering Techniques

- **Autoencoder-based Dimensionality Reduction**: Extracted compact latent representations.
- **t-SNE Visualization**: Identified clusters in two or three dimensions.

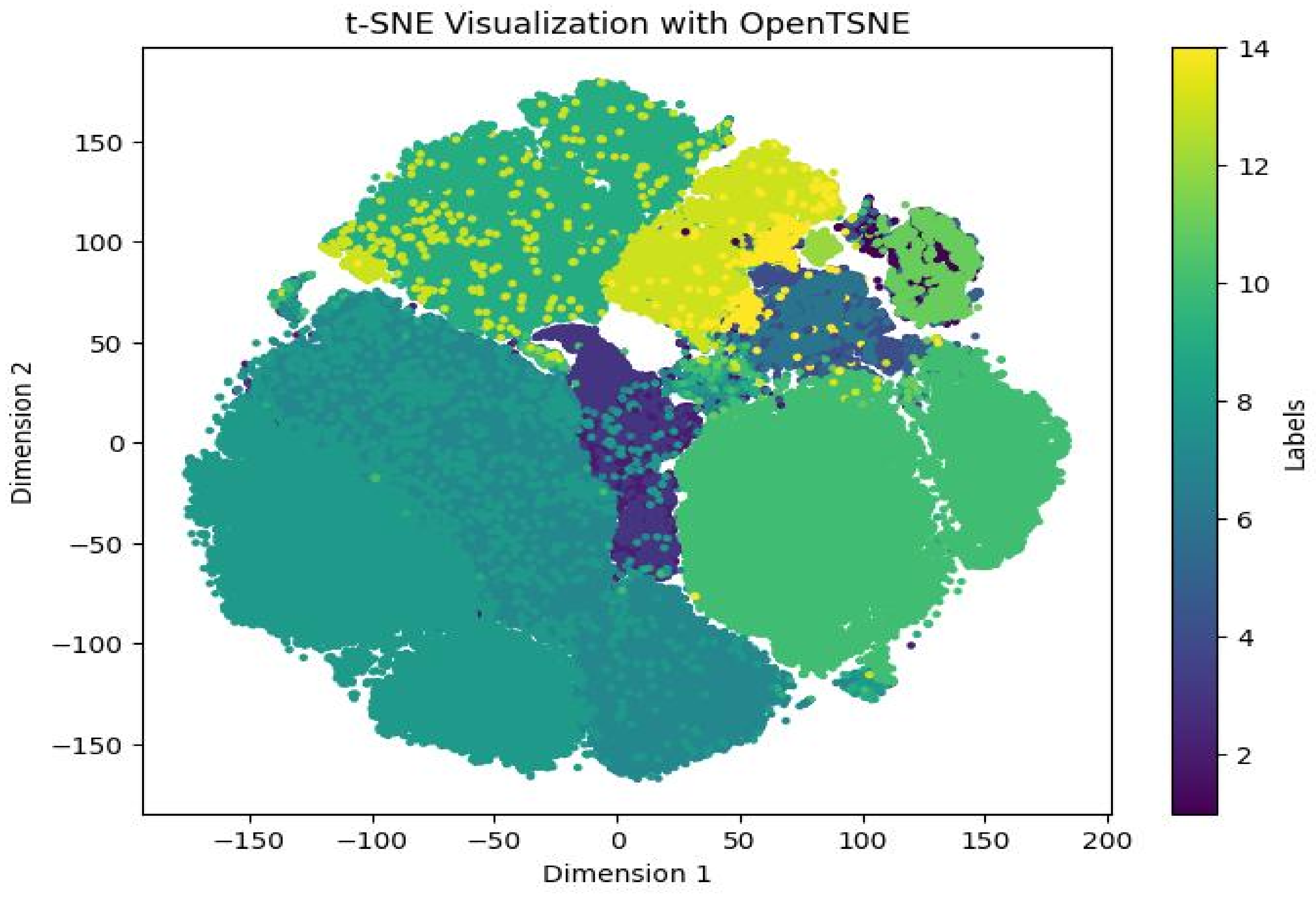

**Fig3:** *TSNE implementation on the preprocessed data*

## 5. Implementation

### 5.1 Feature Engineering

- **Label Encoding**: Transformed categorical labels into integers.
- **Standardization**: Scaled features for consistency.

### 5.2 Semi-Supervised Learning

Semi-supervised learning combines the strengths of labeled and unlabeled data to improve model performance. In this approach, we first utilized unsupervised methods for feature extraction and then fine-tuned using supervised algorithms. Below are the details of the models used and the results obtained:

#### 5.2.1 Data Preparation:

1. Unlabeled data was scaled using StandardScaler.
2. A binary mask was generated with a masking probability (p_m) of **0.5** to introduce controlled corruption in the data, forcing the model to learn data patterns.

#### 5.2.2 Model Architecture:

1. Input Dimension: 37 features.

2. Hidden Layer: A single fully connected layer with ReLU activation.

3. Outputs:

- **Mask Estimation Layer**: Binary cross-entropy loss for predicting missing values.
- **Feature Estimation Layer**: MSE loss to reconstruct original features, weighted by alpha = 1.0.

#### 5.2.3 Training:

1. **Batch Size**: 128, **Epochs**: 50.

2. Optimizer: RMSprop with weighted loss (mask_estimation_loss: 1.0, feature_estimation_loss: 1.0).

3. The model achieved convergence at **50 epochs**, with final feature estimation loss of **0.6936** and mask estimation loss of **0.8807**.

#### 5.2.4 Results:

The trained encoder was extracted, which transforms raw features into representations suitable for downstream tasks.

## 5.3 Supervised Fine-Tuning and Testing

After feature extraction, supervised learning models were applied for clustering and classification tasks on labeled data:

### 5.3.1 Logistic Regression

1. **Purpose**: A linear, interpretable model with probabilistic outputs for binary/multiclass classification.
2. **Procedure**:

    1. Input data was scaled using StandardScaler.
    2. A logistic regression model (LogisticRegression from sklearn) was trained with a maximum of **500 iterations**.

    3.Probabilities for each class were predicted using the predict_proba method.

3. **Results**:

    1. Log loss on the test data: **0.0299**, indicating excellent probabilistic predictions.

### 5.3.2 XGBoost

1. **Purpose**: A non-linear, ensemble-based model that uses gradient boosting for improved performance, especially on complex datasets.
2. **Procedure**:

    1. Labels were adjusted by subtracting 1 (required by XGBClassifier).
    2. The XGBClassifier was trained with **log loss** as the evaluation metric.
    3. Probabilities for each class were predicted.

3. **Results**:

Log loss on the test data: **0.0039**, showcasing the power of ensemble methods in reducing errors.

**5.4 Model Comparison**

| Model | Log Loss | Advantages |
|---|---|---|
| Logistic Regression | 0.0299 | Interpretable, efficient for high-dimensional data. |
| XGBoost | 0.0039 | Handles non-linear relationships, robust against missing and imbalanced data. |

**Table 1:** *Model Comparision*

**5.5 Self-Supervised Learning**:

The **self-supervised encoder** was utilized to generate feature representations from unlabeled data, which were subsequently used in semi-supervised learning. This approach combines unsupervised feature learning with supervised fine-tuning, leveraging both labeled and unlabeled datasets.

**5.5.1 Encoder Model Architecture**

The encoder model was trained using self-supervised learning, as described in Section 5.4. It outputs encoded features for further use in downstream tasks. The main components of the encoder include:

1. **Input Layer**: Processes input features from unlabeled data.
2. **Hidden Layers**: Two fully connected layers with **ReLU activation**.
3. **Outputs**: Mask and feature reconstruction outputs for self-supervised learning.

The encoder was trained with a combination of binary cross-entropy and mean squared error losses. Upon training, the encoder achieved the ability to extract meaningful feature representations, which were then frozen and used for supervised tasks.

**5.5.2 Semi-Supervised Classification**

Once the encoder was trained, it was used to transform both labeled and unlabeled datasets. These encoded features were then passed into a **custom semi-supervised model** designed to classify data by optimizing both supervised and unsupervised losses. Key details of this step are:

1. **Custom Model**:

- **Input Dimension**: Encoder's output dimension.

- **Hidden Layers**: Two dense layers with **ReLU activation**.
- **Output Layer**: A softmax layer for classification probabilities.

2. **Training Process**:

- **Supervised Loss:** Cross-entropy loss for labeled data.
- **Unsupervised Loss:** Regularization term to minimize variance in predicted logits for unlabeled data.
- **Total Loss:** Combined as total_loss = supervised_loss + β * unsupervised_loss, where β = 1.0.
- **Optimizer:** Adam optimizer with default learning rate.

3. **Masking Mechanism**:
Unlabeled data was corrupted using a random binary mask (`mask_probability = 0.3`), and multiple views of the data were generated (`K = 3`). These masked views ensured robustness and better generalization.

### 5.5.3 Performance Evaluation

After training, the semi-supervised model was evaluated on test data using two key metrics:

1. **Accuracy**: Measures the proportion of correct predictions among all samples.
2. **AUROC**: Evaluates the model's ability to distinguish between classes, particularly for imbalanced datasets.

#### 5.5.3.1 Self Learning Results:

- **Accuracy**: **93.54%**
- **AUROC**: **99.09%**

#### 5.5.3.2 Sample Results:

- Ground Truth (Y_test): [10, 9, 9, 2, 10]
- Predictions (Y_pred): [10, 9, 9, 2, 10]

## 6. Results and Evaluation

### 6.1 Quantitative Metrics

- **Log Loss**: Used to measure classification accuracy.

- **Accuracy**: Evaluated on testing sets.

### 6.2 Visual Insights

1. **t-SNE Plots**: Clear separations between identified clusters.

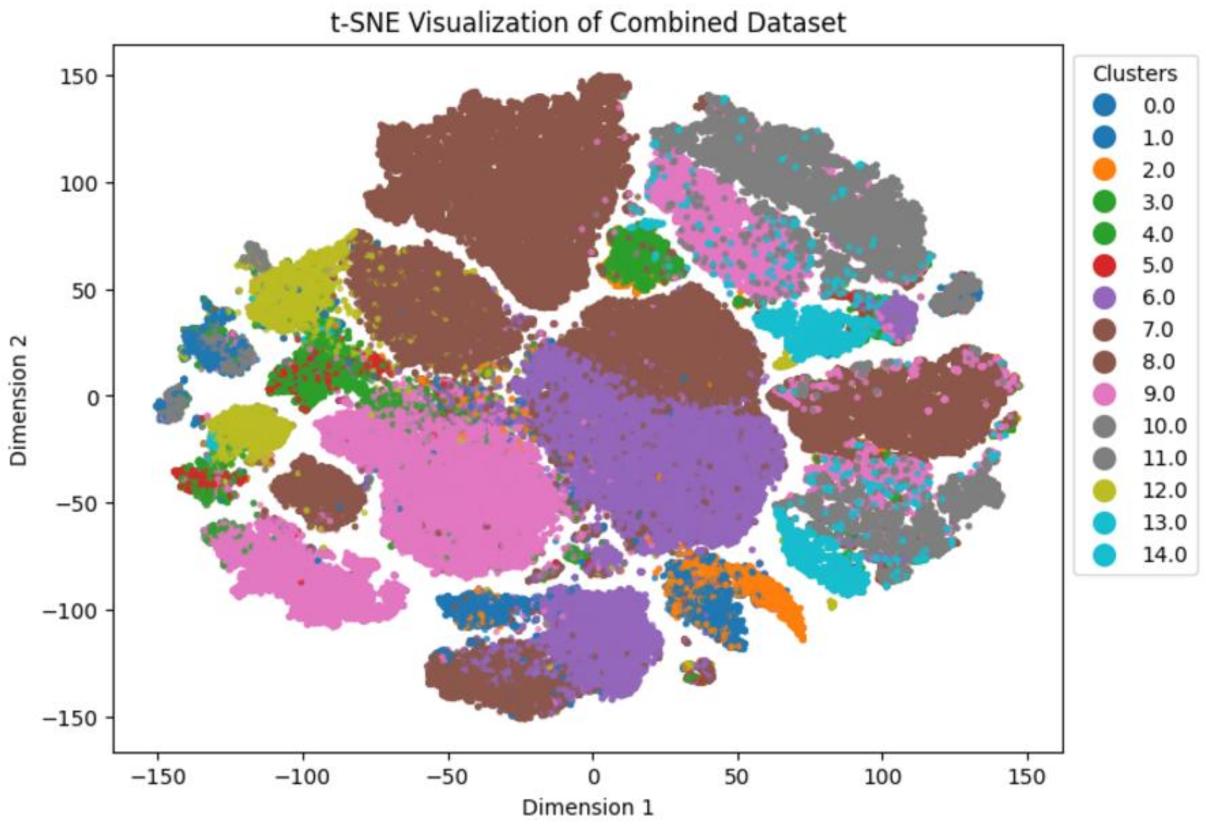

**Fig 4:***TSNE implementation after Encoder predictions*

2. **Marker Distributions**: Highlighted biological marker significance.

## 7. Gradio Interface for Visualization

To simplify user interaction with the system and provide dynamic visualizations, a Gradio-based interface was created. The interface processes a subset of unlabeled data, predicts cell types, and visualizes them with **t-SNE**.

### 7.1 Code Explanation for Gradio Interface

1. **Prediction Function (**generate_unlabeled_predictions**)**:
   1. Encodes unlabeled data using the pre-trained encoder.
   2. Applies the trained model to generate predicted labels.

2. **t-SNE Visualization (**plot_tsne_opentsne**):**

    1. Projects high-dimensional data into two dimensions using t-SNE.
    2. Distinct colors represent predicted clusters, improving interpretability.

3. **Gradio Integration (**process_and_visualize**):**

    1. Accepts start and end rows for the subset of unlabeled data.
    2. Processes data, predicts labels, and returns both a t-SNE plot and predictions in a tabular format.

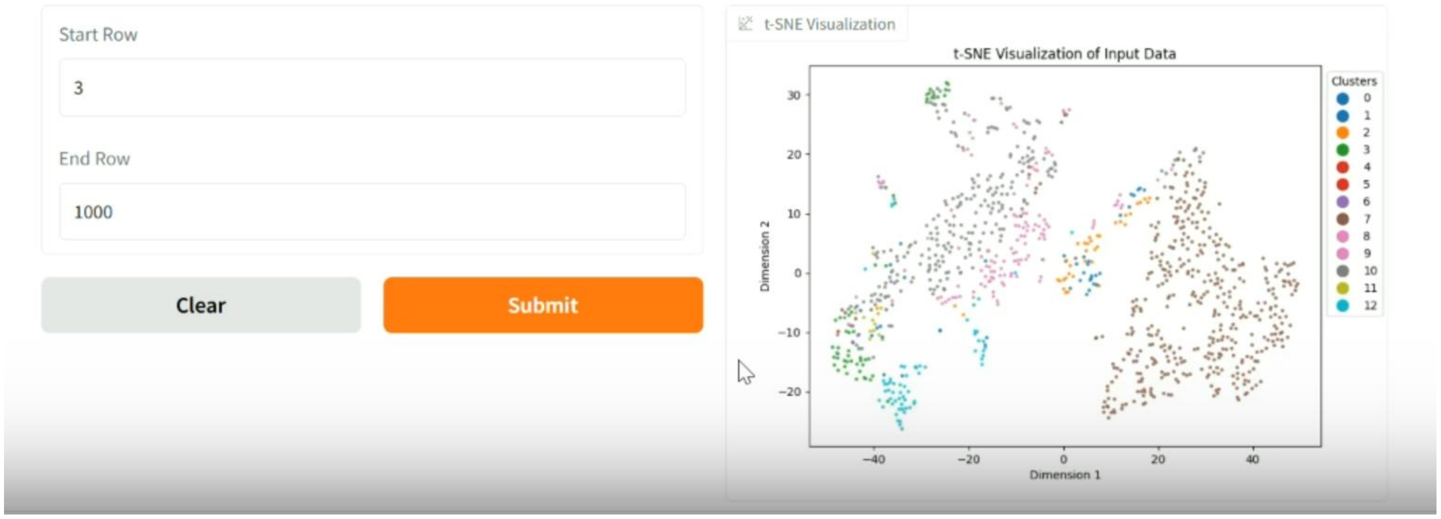
Start Row: 3
End Row: 1000

**Fig 5:***Gradio Interface*

**This interface enables seamless exploration of large cytometry datasets, empowering researchers to interactively analyze and validate results. It bridges the gap between advanced machine learning models and**

## 8. Challenges and Solutions

### 8.1 Noisy Data

**Challenge**: Cytometry datasets often include noise due to instrument variability, sample preparation, or biological factors.

**Solution**:

- Preprocessing pipelines were implemented, including scaling, normalization, and filtering techniques to reduce noise.
- Data augmentation strategies, such as random masking and dropout during training, were applied to improve model robustness.

### 8.2 Label Imbalance

**Challenge**: Labeled cytometry data is often imbalanced, with rare cell populations underrepresented.

**Solution**:

- **Semi-Supervised Learning**: Utilized both labeled and unlabeled data to enhance model generalization and accuracy.
- **Class Weights**: Adjusted loss functions to penalize errors on underrepresented classes more heavily.
- **Data Augmentation**: Synthetic oversampling was used for rare classes.

### 8.3 High Dimensionality

**Challenge**: Cytometry datasets can include thousands of features, making analysis computationally expensive and prone to overfitting.

**Solution**:

- **Dimensionality Reduction**: Used self-supervised autoencoders to reduce input dimensionality while retaining meaningful features.
- **Feature Selection**: Important biological markers were identified to limit the input space.

## 9. Future Work

1. **Explore Multi-Omics Datasets**: Extend the model to analyze and integrate multi-omics data, such as genomics, transcriptomics, and proteomics.
2. **Diverse Cytometry Datasets**: Apply the framework to datasets from different cytometry modalities, such as flow cytometry and mass cytometry.
3. **Scalability**: Enhance the model's ability to handle large-scale datasets with millions of cells, utilizing distributed and parallel computing techniques.
4. **Domain Adaptation**: Address cross-laboratory variability by incorporating domain adaptation techniques.

## 10. Conclusion

The **CytoAutoCluster** project presents a novel semi-supervised clustering framework for cytometry data. By leveraging self-supervised learning and combining it with a robust clustering model, this method:

- **Identifies cell populations effectively**, even with limited labeled data.

- **Improves biomedical research** by providing interpretable results for rare and complex cell subtypes.
- **Advances diagnostics** by enhancing the ability to classify and study cellular phenotypes in various diseases.

## 11. References and Acknowledgments

**DemoLink:**

https://www.loom.com/share/2f2860854bc4432782f21f0d5a2f412e?sid=6caf1aaf-7012-4e7a-8686-17867349d4ee

**References**

1. Levine, J. H., et al. (2015). *Data-Driven Phenotypic Dissection of AML*. Cell, **162**, pp. 184–197.
2. Publicly available cytometry datasets (e.g., Kaggle).