CytoAutoCluster Project Documentation

Project by: Abhay Raj Singh

Internship at: Infosys Springboard

Start Date: October 5th, 2024

Table of Contents

i. Introduction

The project titled CytoAutoCluster: Enhancing Cytometry with Deep Learning aims to implement a machine learning model that employs semi-supervised learning for deep clustering on cytometry data. The overarching goal is to improve the accuracy and interpretability of cell population identification. Cytometry data, specifically mass cytometry (CyTOF) data, are highly multidimensional, posing challenges for traditional clustering techniques. This project, therefore, aims to leverage deep learning methods to extract meaningful patterns, addressing the imbalanced nature of the data, and ultimately enhancing the classification of unlabelled cell populations.

ii. Dataset Description

After exploring several resources, including Google Scholar and Kaggle, the dataset selected for this project is Levin 32, a well-known 32-dimensional mass cytometry (CyTOF) dataset. The dataset comprises the following characteristics:

- Cells: 265,627 individual cells

- Protein Markers (Dimensions): 32 protein markers
- Manually Gated Cell Populations (Clusters): 14 unique cell populations

- Cluster Labels: Available for 39% (104,184 cells) of the dataset

- Individuals Represented: 2 distinct individuals

iii. Initial Observations:

Based on initial exploration, the dataset exhibited several key characteristics typical of high-dimensional cytometry data:

· **High Dimensionality:** 32 dimensions create opportunities for deep feature extraction but pose challenges for computational efficiency.

· **Imbalanced Label Distribution:** Only 39% of the data has manual labels, making this dataset a suitable candidate for **semi-supervised learning**.

· **Variation Across Populations:** The cluster distribution varies significantly across the 14 cell populations, with some populations underrepresented.

Project Process Overview

1. Data Cleaning

Data cleaning is an essential step to ensure accurate model performance. The following steps were taken during the cleaning process.

· **Handling Null Values:** The dataset contains both labeled and unlabeled data. We identified null values in some columns, which represented unlabeled data. A key challenge was visualizing these null values accurately due to overlapping visualizations.

· **Action Taken:** We corrected this by creating non-overlapping histograms that accurately represented null and non-null values.

· · **Removing Unnecessary Columns:** We identified and removed several columns that were redundant or irrelevant to the clustering task, including those without labels such as "Viability", "file_number", and "event_number".

Challenges Faced:

· **Visualization Issues:** While plotting null/non-null value graphs, there was significant overlap in the values, initially making it difficult to gain insights.

· **Empty Columns:** Some columns were found to have no names and caused errors during processing. This was resolved by explicitly removing these unnamed columns.

1.1 . Exploratory Data Analysis (EDA)

EDA helped in understanding the underlying patterns and structure of the data.

Several statistical techniques were applied, each offering unique insights:

1.2 Techniques Used:

· **Histograms:** These were used to visualize the distribution of values in each feature. The histograms were particularly useful for highlighting **imbalances in cell populations** and **variations in feature values**.

· **Pair Plots:** Pairwise relationships between features were visualized to understand potential correlations and interactions among protein markers.

· **Skewness & Kurtosis:** The distribution of each feature was assessed through skewness and kurtosis metrics. These statistics provided insights into the asymmetry and peakedness of the feature distributions, which are crucial when selecting appropriate algorithms for machine learning.

1.3 Visualizations Created:

· **Null and Non-null Value Distribution:** Histograms were created to display the null and non-null values across the dataset. These graphs clearly distinguished between the labeled and unlabeled parts of the dataset.

· **Skewness and Kurtosis Graphs:** Individual plots for features like 'DNA1', 'DNA2', 'CD45RA', and others were generated to represent the positive and negative skewness of the data, as well as the kurtosis.

Interpretations:

· **Skewness:** Features like 'DNA1' exhibited positive skewness, indicating that a significant number of observations had lower values than the median.

· **Kurtosis:** The kurtosis analysis revealed that certain features, such as 'CD45RA', exhibited a high kurtosis, suggesting a presence of outliers.

1.4 Dimensionality Reduction Techniques: PCA and t-SNE

Purpose: PCA and t-SNE were employed for reducing data dimensions, allowing better visualization of clusters while preserving key patterns. PCA captures linear variance, while t-SNE provides a nonlinear view, clustering similar data points.
Challenges: Selecting optimal components for PCA and tuning t-SNE's perplexity and iterations were complex, as misconfiguration could distort cluster formations.

Outcomes: PCA highlighted primary variance directions, and t-SNE visualized distinct clusters, setting the foundation for further dimensionality reduction with Autoencoders.

2. Autoencoders for Feature Extraction

Purpose: Autoencoders compressed the high-dimensional data into latent features, improving representation for clustering. The aim was to balance efficient encoding with minimal information loss.
Challenges: Hyperparameter tuning to prevent overfitting and deciding the bottleneck layer size were key challenges.
Outcomes: Autoencoders successfully captured core features, enhancing data representation and paving the way for semi-supervised learning on imbalanced data.

3. Semi-Supervised Learning (SSL)

Purpose: SSL was applied to leverage both labeled and unlabeled data, utilizing techniques like entropy minimization and consistency regularization to maximize the model's learning potential.
Challenges: Implementing regularization methods to generalize the model effectively was challenging, requiring extensive experimentation with SSL parameters.
Outcomes: SSL enhanced model performance by effectively learning from unlabeled data, supporting data corruption techniques in the next phase.

4. Data Corruption Techniques

Purpose: Binary masking and data shuffling introduced controlled noise, encouraging model robustness by simulating data variability.
Challenges: Balancing data masking levels to obscure features without distorting patterns was essential.
Outcomes: Data corruption techniques improved the model's adaptability, enabling it to identify key patterns in noisy or partial data. This prepared the data for initial model testing.

5. Model Development and Evaluation

Logistic Regression and XGBoost: These models provided a baseline (Logistic Regression) and high performance (XGBoost) with encoded data. XGBoost's ability to handle complex structures complemented Logistic Regression's simplicity.
Challenges: Overfitting prevention through feature scaling and loss value adjustments were necessary to optimize performance.
Outcomes: Both models leveraged encoded features effectively. XGBoost, in particular,

showed strong accuracy on complex data, leading to Encoder model exploration.

6. Encoder Model Development

Purpose: The Encoder model refined latent features, generating high-quality representations for downstream training.
Challenges: Dimension balancing and training speed optimization required careful adjustments.
Outcomes: Encoder-generated features improved model accuracy when used in supervised classifiers, building the foundation for further training on latent data.

The encoder model was trained using self-supervised learning. 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.

6.1 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.

7. 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.

7.1 Self Learning Results:

```
Accuracy: 0.8908
```

```
AUROC: 0.9888
```

7.2 Sample Results:

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

Predictions (Y_pred): [10, 9, 9, 2, 10]

8. Results and Evaluation

8.1 Quantitative Metrics

Log Loss: Used to measure classification accuracy.

Accuracy: Evaluated on testing sets.

8.2 Visual Insights

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

2.Marker Distributions: Highlighted biological marker significance.

9. 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.

8.1 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.

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

9. Challenges and Solutions

9.1 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.

9.2 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.

9.3 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.

10. 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.

11. 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

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).

Final Timeline

October 17, 2024

Agenda: Complete Data Exploration, implement PCA and t-SNE

Action Items: Gain deeper understanding of PCA and t-SNE

October 18, 2024

Agenda: Review PCA and t-SNE outputs

Action Items: Finalize t-SNE and compare with other results

October 22, 2024

Agenda: Begin work on Autoencoders

Action Items: Basic understanding of Autoencoders

October 23, 2024

Agenda: Identify Autoencoder models for tabular data

Action Items: Understand Autoencoders for feature extraction

October 25, 2024

Agenda: Learn Semi-Supervised Learning basics

Action Items: Study SSL concepts

October 28, 2024

Agenda: Discuss entropy minimization and consistency regularization

Action Items: Implement SSL techniques

October 29, 2024

Agenda: Review binary mask implementation

Action Items: Evaluate binary mask outputs

October 30, 2024

Agenda: Data corruption methods

Action Items: Evaluate corrupted data for training

November 1, 2024
Agenda: Data splitting and train-test setup

Action Items: Complete model preparation

November 5, 2024
Agenda: Implement Logistic Regression

Action Items: Finalize model parameters

November 7, 2024
Agenda: Develop Encoder model

Action Items: Initiate latent feature training

November 8, 2024
Agenda: Train on Logistic Regression and XGBoost

Action Items: Troubleshoot Encoder-based training

November 12, 2024
Agenda: Begin Semi-Supervised Learning model

Action Items: Implement SSL on encoded data

November 13, 2024
Agenda: Data splitting and train-test setup

Action Items: Complete model preparation

November 15, 2024
Agenda: semi supervised learning model

Generate labels for unlabeled data

November 19, 2024
Agenda: outputs of semi supervised model

Action Items: Calculate performance metrics

November 20, 2024
Agenda: Calculate performance metrics

Action Items: Generate labels for unlabeled data

November 21, 2024
Agenda: Complete training semi supervised learning model

Action Items:Input predicted labels into dataframe

November 22, 2024

Agenda: Check generated predictions on unlabeled data

Action Items: Compute tsne outputs

November 25, 2024

Agenda: Check generated predictions on unlabeled data

Compute tsne outputs

November 26, 2024

Agenda: Check tsne and gradio outputs

Action Items: Complete gradio