UNITO

Recent advances in cytometry technology have enabled high-throughput data collection with multiple single-cell protein expression measurements. The significant biological and technical variance between samples in cytometry has long posed a formidable challenge during the gating process, especially for the initial pre-gates which deal with unpredictable events, such as debris and technical artifacts. Even with the same experimental machine and protocol, the target population, as well as the cell population that needs to be excluded, may vary across different measurements. To address this challenge and mitigate the labor-intensive manual gating process, we propose a deep learning framework UNITO to rigorously identify the hierarchical cytometric subpopulations. The UNITO framework transformed a cell-level classification task into an image-based semantic segmentation problem. For reproducibility purposes, the framework was applied on three independent cohorts (two mass cytometry cohorts and one flow cytometry dataset) and successfully detected initial gates that were required to identify single cellular events as well as subsequent cell gates. We validated the UNITO framework by comparing its results with previous automated methods and the consensus of at least four experienced immunologists. UNITO outperformed existing automated methods and differed from human consensus by no more than each individual human. Most critically, UNITO framework functions as a fully automated pipeline after training for either mass cytometry and flow cytometry, and it does not require human hints or prior knowledge for automatic gating. Unlike existing multi-channel classification or clustering pipelines, UNITO can reproduce a similar contour compared to manual gating for each intermediate gating to achieve better interpretability and provide post hoc visual inspection. Beyond acting as a pioneer framework that uses image segmentation to do auto-gating, UNITO gives an interpretable way to assign the cell subtype membership, and it also allows easy parallelization of samples for faster processing. The pre-gating and gating inference takes approximately 10 seconds for each sample gate using our pre-defined 9 gates system, and it can also adapt to any sequential prediction with different configurations.

License

Installation

We highly recommend that users install Anaconda3 on their machines. After installing Anaconda3, UNITO can be used by installing the following packages: numpy, pandas, matplotlib, torch, torchvision, seaborn, scipy, sklearn, cv2, albumentations, datashader.

We recommend that users use the Conda virtual environment:

```
$ conda create --name UNITO_demo python=3.9
```

Activate the virtual environment

```
$ conda activate UNITO_demo
```

Install required packages

```
$ conda install pytorch::pytorch torchvision torchaudio -c pytorch
$ conda install anaconda::numpy pandas matplotlib seaborn scipy scikit-learn
$ conda install -c conda-forge opencv albumentations datashader
```

Input structure

The main functions of UNITO takes the cytometric measurement data as input and user can provide names for the two protein channel for bivariate plot and the gate name to perform automatic gating. Users can also provide gate name for previous gate to filter out-of-gate cells from previous gate. The input format of UNITO is essentially two columns of protein measurement and an addition binary column for cell types (cell type label required only for training UNITO), and we provide an example script preparing the input data from OMIQ platform.

Example for **cytometric data (flow cytometry below as an example)**:

```
FSC_A   FSC_W   SSC_A   SSC_W   LIVEDEAD    CD3Q605
54436.3789  80784.9141  51817.4023  75125.6641  1.2608  2.4109
38505.3594  78259.7969  37893.0508  76034.3828  0.7803  2.9049
14750.7598  86645.6797  33867.5508  78849.8281  1.011   2.8193
29421.8398  84699.7578  21625.1797  78976.1875  1.0272  3.8933
8042.6997   77592.5703  4556.0903   75918.6172  2.3909  0.2797
20202.1992  83441.8203  47338.9102  77531   1.1574  1.0303
35963.0703  90115.3125  38573.0195  93939.8594  1.6715  3.2541
26660.4297  89495.3594  32338.8301  73013.3125  0.7821  3.4795
```

Available Versions

2. Python Script for Training a Customized Model + Automatic Gating The Python script versions are designed for users who are comfortable working in a command-line environment and want direct control over the process. These scripts allow flexibility in configuring gating structures and model training process, and is easier to implement on cloud for large scale datasets.

This version allows you to train a custom model using your own labeled data and then apply automatic gating. It provides the flexibility to modify the training process and model architecture. You start by feeding in your manually labeled data to train a deep learning model. Once the model is trained, you can save it and apply it to new, unlabeled data for automatic gating. The script also allows tweaking hyperparameters to improve model performance.

Implementation

After installing the relevant packages into your target virtual environment, follow these simple steps to get started:

1. **Prepare Your Data**:
   Organize your CSV data into two folders:

   - `Raw_Data_train` for training data
   - `Raw_Data_pred` for prediction data

2. **Specify the Gating Sequence**:
   Provide the gating sequence you wish to use. An example is included in this repository for reference.

3. **Adjust Parameters (Optional)**:
   Review and modify any parameters to suit your needs. Some key parameters you might want to adjust include:

   - **Output Path**: Defaults to the project folder.
   - **Epochs**: Defaults to 1000.
   - **Computing Device**: Defaults to `"mps"` for MacOS. You can change this based on your system's available hardware (e.g., `"cuda"` for GPU on Windows/Linux).

4. **Run the Script**:
   Execute the script by running `main.py` to start the process.

   ```
   python main.py
   ```

5. **Check Output**:
   Output are stored in three folders:

   - **figures**: containing all post-hoc visualization of the gating polygons
   - **model**: UNITO models for future gating
   - **prediction**: prediction of cell-level labels in spreadsheets format.