# **GenomeDefender User Guide**

This guide provides step-by-step instructions to test and use the **GenomeDefender** tool for detecting data poisoning attacks in single-cell RNA sequencing (scRNA-seq) data in 10x Genomics format. The tool supports four models: Contrastive Autoencoder (CAE) for synthetic cell injections, Variational Autoencoder (VAE) for gene scaling, Graph Neural Network Autoencoder (GNN-AE) for label flips, and Denoising Diffusion Probabilistic Model (DDPM) for injected biological noise. This guide includes instructions for setting up the environment, preparing data, training models, running detection, and debugging common errors.

## **Prerequisites**

### **1. Data**

* **Format**: 10x Genomics format, consisting of three files in a directory:
  + matrix.mtx: Sparse matrix of gene expression counts (rows: genes, columns: cells).
  + barcodes.tsv: Cell barcodes (one per line).
  + features.tsv: Gene IDs/names (one per line).
* **Example**: Download a dataset like GSE154826 from GEO or create a small test dataset (e.g., 10,000 cells).
* **Directory**: Place data in a directory, e.g., data/gse154826/, containing matrix.mtx, barcodes.tsv, and features.tsv.

### **2. Environment**

* Install required Python packages using pip:
  + Required packages: torch, torch-geometric, numpy, scanpy, pyyaml, tqdm, scikit-learn.
  + Command: Run pip install torch torch-geometric numpy scanpy pyyaml tqdm scikit-learn in your Python environment.
* **Hardware**: A GPU with CUDA support is recommended for faster training, but the tool falls back to CPU if CUDA is unavailable.
* **Python Version**: Python 3.8 or higher is recommended.

### **3. Project Structure**

* Create the following directories in your project root:
  + data/: For 10x Genomics datasets (e.g., data/gse154826/).
  + configs/: For YAML configuration files.
  + weights/: For saving trained model weights.
  + logs/: For training and detection logs.
  + results/: For detection output (anomaly scores).
* Place the provided files in the correct directories:
  + main.py: Project root.
  + training/trainer.py: Training script.
  + detection/detector.py: Detection script.
  + preprocessing/preprocess\_train.py, preprocessing/preprocess\_detect.py: Preprocessing scripts.
  + models/CAE.py, models/vae\_model.py, models/gnn\_model.py, models/ddpm\_model.py: Model implementations.
  + configs/cae\_config.yaml, configs/vae\_config.yaml, configs/gnn\_ae\_config.yaml, configs/ddpm\_config.yaml: Configuration files.

## **Testing Steps**

### **1. Prepare Data**

* **Obtain Data**: Download a 10x Genomics dataset (e.g., GSE154826 from GEO) or simulate a test dataset with known attacks for validation.
* **Simulated Attacks (Optional)**: To test detection, you can modify a clean dataset to include attacks like synthetic cells, gene scaling, label flips, or injected noise. Use a Python script with scanpy to simulate attacks (see Advanced Testing for details).
* **Directory Setup**: Ensure your dataset directory contains matrix.mtx, barcodes.tsv, and features.tsv. Example: data/test\_data/.

### **2. Train a Model**

* **Purpose**: Train a model on clean scRNA-seq data to learn normal patterns.
* **Command**: Use main.py with --mode train to train a model. Specify the model, dataset directory, config file, and output path for model weights.
* **Examples**:
  + Train CAE for synthetic cell detection:
    - Command: python main.py --mode train --model cae --dataset data/gse154826 --config configs/cae\_config.yaml --output weights/cae\_synthetic\_cells.pt --log\_level INFO
  + Train VAE for gene scaling detection:
    - Command: python main.py --mode train --model vae --dataset data/gse154826 --config configs/vae\_config.yaml --output weights/vae\_gene\_scaling.pt --log\_level INFO
  + Train GNN-AE for label flip detection:
    - Command: python main.py --mode train --model gnn\_ae --dataset data/gse154826 --config configs/gnn\_ae\_config.yaml --output weights/gnn\_ae\_label\_flips.pt --log\_level INFO
  + Train DDPM for injected noise detection:
    - Command: python main.py --mode train --model ddpm --dataset data/gse154826 --config configs/ddpm\_config.yaml --output weights/ddpm\_noise.pt --log\_level INFO
* **Expected Output**:
  + Logs in logs/genome\_defender.log and console, showing training progress (e.g., average loss per epoch).
  + Example:

2025-07-27 06:34:00 - INFO - Loading configuration from configs/cae\_config.yaml

2025-07-27 06:34:00 - INFO - Starting training for cae on data/gse154826

Epoch 0: Avg Train Loss: 0.1234 (Recon: 0.0987, Contrastive: 0.0247)

...

2025-07-27 06:35:00 - INFO - Training completed successfully

* + Trained model weights saved to weights/ (e.g., weights/cae\_synthetic\_cells.pt).

### **3. Incremental Training**

* **Purpose**: Fine-tune a pre-trained model on new clean data to improve performance.
* **Command**: Add --incremental to the training command. Ensure pre-trained weights exist in weights/.
* **Example**:
  + Fine-tune VAE on new data:
    - Command: python main.py --mode train --model vae --dataset data/new\_data --config configs/vae\_config.yaml --output weights/vae\_finetuned.pt --incremental --log\_level INFO
  + Requires: weights/vae\_gene\_scaling.pt from prior training.
* **Expected Output**:
  + Similar to training, with logs indicating incremental training and lower learning rate.
  + Updated weights saved to weights/vae\_finetuned.pt.

### **4. Run Detection**

* **Purpose**: Detect poisoning attacks in a test dataset using a trained model.
* **Command**: Use main.py with --mode detect to run detection. Specify the model, dataset directory, config file, and output path for anomaly scores.
* **Examples**:
  + Detect synthetic cells with CAE:
    - Command: python main.py --mode detect --model cae --dataset data/test\_data --config configs/cae\_config.yaml --output results/cae\_anomalies.csv --log\_level INFO
  + Detect gene scaling with VAE:
    - Command: python main.py --mode detect --model vae --dataset data/test\_data --config configs/vae\_config.yaml --output results/vae\_anomalies.csv --log\_level INFO
  + Detect label flips with GNN-AE:
    - Command: python main.py --mode detect --model gnn\_ae --dataset data/test\_data --config configs/gnn\_ae\_config.yaml --output results/gnn\_anomalies.csv --log\_level INFO
  + Detect injected noise with DDPM:
    - Command: python main.py --mode detect --model ddpm --dataset data/test\_data --config configs/ddpm\_config.yaml --output results/ddpm\_anomalies.csv --log\_level INFO
* **Expected Output**:
  + Logs in logs/genome\_defender.log and console, showing detection results (percentage of dataset potentially poisoned, suspected cells/genes).
  + Example:

2025-07-27 06:36:00 - INFO - Loading configuration from configs/cae\_config.yaml

2025-07-27 06:36:00 - INFO - Starting detection for cae on data/test\_data

Detection completed, 4.75% of the dataset could be poisoned, suspected cells: [12, 45, 67, 89, ...]

Anomaly scores saved to results/cae\_anomalies.csv

2025-07-27 06:36:10 - INFO - Detection completed successfully

* + Anomaly scores saved as CSV in results/ (e.g., results/cae\_anomalies.csv).

### **5. Verify Outputs**

* **Training Outputs**:
  + Check weights/ for model files (e.g., cae\_synthetic\_cells.pt, vae\_finetuned.pt).
* **Detection Outputs**:
  + Check results/ for CSV files containing anomaly scores (e.g., cae\_anomalies.csv).
  + Each CSV contains one score per cell (CAE, GNN-AE, DDPM) or gene (VAE).
* **Logs**:
  + Check logs/genome\_defender.log for detailed training/detection logs, useful for debugging.

## **Debugging Common Errors**

1. **Error: Dataset directory does not exist**
   * **Cause**: The --dataset path doesn’t point to a valid directory containing matrix.mtx, barcodes.tsv, and features.tsv.
   * **Fix**: Verify the directory exists and contains all three files. Example:
     + Run ls data/gse154826/ to check for matrix.mtx, barcodes.tsv, features.tsv.
     + Ensure the path matches the --dataset argument (e.g., data/gse154826).
2. **Error: Model file not found**
   * **Cause**: Pre-trained weights (e.g., weights/cae\_synthetic\_cells.pt) are missing for --incremental training or detection.
   * **Fix**: Train the model first to generate weights:
     + Example: python main.py --mode train --model cae --dataset data/gse154826 --config configs/cae\_config.yaml --output weights/cae\_synthetic\_cells.pt
     + Then retry incremental training or detection.
3. **Error: Unsupported model**
   * **Cause**: Invalid --model argument (must be cae, vae, gnn\_ae, or ddpm).
   * **Fix**: Use a valid model name:
     + Example: python main.py --mode detect --model cae ...
4. **Error: MemoryError during preprocessing**
   * **Cause**: Dataset too large for available memory.
   * **Fix**: Edit the YAML config (e.g., configs/cae\_config.yaml) to reduce batch\_size:
     + Example: Set batch\_size: 5000 instead of 10000.
     + Retry the command with the updated config.
5. **Error: CUDA out of memory**
   * **Cause**: GPU memory exhausted during training or detection.
   * **Fix**:
     + Reduce train\_batch\_size in the YAML config (e.g., train\_batch\_size: 16).
     + Switch to CPU by setting device: cpu in the YAML config.
     + Reduce input\_dim (e.g., input\_dim: 1000) for lower memory usage.
     + Example config change:

device: cpu

train\_batch\_size: 16

input\_dim: 1000

1.   
   **Poor Detection Performance**
   * **Cause**: Model not trained sufficiently, preprocessing inconsistent, or detection threshold too strict.
   * **Fix**:
     + Increase epochs (e.g., epochs: 200) or adjust learning\_rate (e.g., learning\_rate: 0.0005) in the YAML config.
     + Ensure input\_dim is the same for training and detection.
     + Adjust detection\_threshold (e.g., detection\_threshold: 0.90 for top 10% anomalies) in the YAML config.
     + Validate on simulated data with known attacks (see Advanced Testing).

## **Advanced Testing**

### **1. Simulate Attacks for Validation**

* Create a test dataset with known attacks to verify detection performance.
* Steps:
  + Load a clean 10x Genomics dataset using scanpy.
  + Simulate attacks:
    - **Synthetic Cells**: Add rows with random gene expression.
    - **Gene Scaling**: Multiply random genes by a factor (e.g., 2).
    - **Label Flips**: Shuffle labels for a subset of cells (if labels are available).
    - **Injected Noise**: Add Gaussian noise to a subset of cells/genes.
  + Save the modified dataset to a new directory (e.g., data/test\_data).
* Run detection on the simulated dataset and check if the suspected indices match the attacked cells/genes.

### **2. Evaluate Metrics**

* If ground-truth labels are available (e.g., known poisoned cells), compute AUC-ROC:
  + Load anomaly scores from the output CSV (e.g., results/cae\_anomalies.csv).
  + Use scikit-learn to compute metrics:
    - Example: Calculate AUC-ROC with roc\_auc\_score(true\_labels, scores).
  + Check logs for training loss trends to ensure convergence.

### **3. Tune Hyperparameters**

* **Detection Threshold**: Adjust detection\_threshold in YAML configs (e.g., 0.90 for top 10%) to balance sensitivity and specificity.
* **Graph Neighbors (GNN-AE)**: Experiment with k\_neighbors (e.g., 3 or 10) to optimize graph-based detection.
* **Training Parameters**: Increase epochs or adjust learning\_rate for better model performance.
* **Preprocessing**: Reduce input\_dim (e.g., 1000) or batch\_size (e.g., 5000) for faster processing on low-memory systems.

## **Example Workflow**

1. **Train All Models**:  
   * CAE: python main.py --mode train --model cae --dataset data/gse154826 --config configs/cae\_config.yaml --output weights/cae\_synthetic\_cells.pt
   * VAE: python main.py --mode train --model vae --dataset data/gse154826 --config configs/vae\_config.yaml --output weights/vae\_gene\_scaling.pt
   * GNN-AE: python main.py --mode train --model gnn\_ae --dataset data/gse154826 --config configs/gnn\_ae\_config.yaml --output weights/gnn\_ae\_label\_flips.pt
   * DDPM: python main.py --mode train --model ddpm --dataset data/gse154826 --config configs/ddpm\_config.yaml --output weights/ddpm\_noise.pt
2. **Detect Attacks**:  
   * CAE: python main.py --mode detect --model cae --dataset data/test\_data --config configs/cae\_config.yaml --output results/cae\_anomalies.csv
   * VAE: python main.py --mode detect --model vae --dataset data/test\_data --config configs/vae\_config.yaml --output results/vae\_anomalies.csv
   * GNN-AE: python main.py --mode detect --model gnn\_ae --dataset data/test\_data --config configs/gnn\_ae\_config.yaml --output results/gnn\_anomalies.csv
   * DDPM: python main.py --mode detect --model ddpm --dataset data/test\_data --config configs/ddpm\_config.yaml --output results/ddpm\_anomalies.csv
3. **Fine-Tune a Model**:  
   * Example: python main.py --mode train --model ddpm --dataset data/new\_data --config configs/ddpm\_config.yaml --output weights/ddpm\_finetuned.pt --incremental

## **Notes**

* **Logs**: Always check logs/genome\_defender.log for detailed error messages and training/detection progress.
* **Config Files**: Ensure YAML configs (configs/\*.yaml) are correctly set up with matching input\_dim, device, and other parameters for training and detection.
* **Data Consistency**: Use the same dataset directory structure and preprocessing settings for both training and detection to avoid mismatches.
* **Performance**: For large datasets, adjust batch\_size and train\_batch\_size in YAML configs to manage memory usage.

This guide should enable you to test **GenomeDefender** effectively and debug any issues. For further assistance, refer to the logs or contact the developer for support.