**Genomic\_Defender\_v1 First-Time User Guide:**

**Code Review & Pipeline Analysis**

**1. Structural Integrity:**

* All files are present and properly interconnected
* Consistent imports and type hints across modules
* Configuration flows correctly from config.yaml through all components
* No syntax errors or obvious logical flaws detected

**2. Foundational Issues:**

* Missing config\_parser.py (though config handling is built into each module)
* decision\_function.py should be decision\_fusion.py (filename typo)
* GNN model's incremental\_train() method needs implementation
* VAE's drift detection isn't integrated into the training loop

**3. Pipeline Workflow:**

1. config.yaml → Main CLI parameters

2. preprocessor.py → Handles 10x Genomics data (MTX + TSVs)

→ Outputs: PCA-reduced data + cell graph

3. Models train in parallel:

- EIF (full retrain)

- VAE/GNN/ContrastiveAE (incremental)

4. decision\_fusion.py → Confidence-weighted voting

5. logger.py/metrics.py → Track everything

**External User Perspective:**

*"This appears to be a sophisticated genomic anomaly detection system designed for:*

1. **Continuous Monitoring**: The incremental training suggests it's meant for ongoing dataset surveillance
2. **Multi-Attack Detection**: Different models target various attack types (VAE for subtle perturbations, GNN for structural changes)
3. **Research-Grade Outputs**: Comprehensive metrics and JSON logging enable detailed analysis
4. **Bioinformatics Focus**: The MTX/TSV preprocessing reveals single-cell RNA-seq specialization

*Key observations:*

* The tool assumes some poisoned examples exist for model improvement
* Requires significant compute for weekly retraining
* Fusion strategy is elegantly simple (no meta-model to maintain)
* Would benefit from TensorBoard/PyTorch Lightning integration\*"

**First-Time User Guide**

**GenomeGuardian CLI Tool**  
*AI-Powered Genomic Data Poisoning Detection*

**1. Installation**

conda create -n genomeguard python=3.9

conda activate genomeguard

pip install torch torch\_geometric scikit-learn pyyaml eif

git clone https://github.com/yourrepo/genomeguardian.git

cd genomeguardian

**2. Configuration**  
Edit config.yaml:

data:

expected\_contamination: 0.1 # % of poisoned cells expected

models:

eif:

enable: true

n\_trees: 200

vae:

enable: true

latent\_dim: 10

# ... other model configs

fusion:

method: confidence\_voting

**4. Output Interpretation**

* Check logs/ directory for:
  + guardian\_\*.log: JSON-formatted training records
  + audit\_trail.ndjson: Model update history
* Results appear in terminal with colored alerts:

[INFO] Anomalies detected: 14/10500 (0.13%)

[WARNING] Model 'vae' precision dropped 5%

**5. Key Commands**

# Run detection only (no training)

python main.py --detect\_only --data\_path new\_data.h5ad

# Export results to CSV

python utils/export.py --log\_file logs/guardian\_20230801.log

# Visualize metrics

python utils/plot\_metrics.py --model vae

**6. Expected Output Structure**

results/

├── detection\_report.txt

├── anomaly\_scores.csv

models/

├── eif.pkl

├── vae.pt

logs/

├── guardian\_20230801.log

├── audit\_trail.ndjson

**Troubleshooting Tips:**

* For CUDA errors: Set device: cpu in config.yaml
* If preprocessing fails: Verify MTX file is not transposed
* Memory issues: Reduce batch\_size in VAE/GNN configs

**Example Workflow:**

1. Start with known-clean dataset (Week 1)
2. Weekly: Run new data through the pipeline
3. Monitor anomaly\_rate changes in logs/
4. Investigate high-score cells with:

python utils/inspect\_anomalies.py --scores results/week3\_scores.csv

This tool provides enterprise-grade genomic data protection through continuous AI monitoring, with particular strength in detecting subtle, biologically-plausible poisoning attempts.

**1. First-Time Training**

python main.py \

--train \

--matrix\_path data/initial.mtx \

--features\_path data/features.tsv \

--barcodes\_path data/barcodes.tsv

* Creates training/models/ with:
  + eif.pt, vae.pt, gnn.pt, contrastive\_ae.pt

**2. Incremental Training**

python main.py \

--train \

--matrix\_path data/week2.mtx \

--features\_path data/week2\_features.tsv \

--barcodes\_path data/week2\_barcodes.tsv

* Updates all models except EIF incrementally
* EIF gets fully retrained

**3. Detection Mode**

python main.py \

--detect \

--matrix\_path new\_data.mtx \

--features\_path new\_features.tsv \

--barcodes\_path new\_barcodes.tsv