IntCyt v2 Repository Documentation

Reproducibility guide for MNIST / Fashion-MNIST recall benchmarks and ablation experiments

Version: 0.1.0 | Date: 2026-01-26 | Package name: **intcyt**

This document is meant to be **ready to implement**: it lists the exact commands to install the repository, reproduce the test-recall numbers (Table S2 / Table 2), and run the ablation experiments used in the manuscript.

The repository is organized as an installable Python package (**intcyt**) that includes: a lightly-modernized legacy IntCyt runtime (intcyt/legacy), paper evaluation scripts (examples/), thermodynamic ledger utilities (intcyt/thermo), and derived metrics (intcyt/metrics).

1. Installation

Requirements: Python 3.10+ (recommended).

Recommended install (evaluation + ablations):

```
# From the repository root:
pip install -r requirements.txt
pip install -e .
```

Optional extras (defined in pyproject.toml):

```
pip install -e ".[eval]"   # torch/torchvision/scikit-learn (evaluation scripts)
pip install -e ".[dev]"    # pytest (tests)
```

2. Repository structure

Path	Contents / purpose
intcyt/	Installable package (top-level API + CLI implementation).
intcyt/legacy/	Legacy IntCyt runtime (celloperad/cellint/useful) and legacy scripts (software_implementation).
intcyt/core/	Programmatic wrapper around the legacy runtime (simulator + API helpers).
intcyt/thermo/	Ledger + snapshot utilities for counting irreversible operations (Z/M events).
intcyt/metrics/	Entropy, LND, DOE, TUR helper functions for analyzing traces.
examples/	Reproduce Table S2/Table 2 test recall and run ablation scripts.
tests/	Small smoke tests (optional).
cli.py	Compatibility wrapper so 'python cli.py ...' works (installed entry point is 'intcyt').

3. Data setup

Legacy scripts expect these directories at the repository root:

```
mkdir -p data result-load
```

MNIST / Fashion-MNIST sources

All evaluation scripts support two dataset sources via **--source**: **auto** (default, uses torchvision and downloads automatically) and **idx** (loads IDX gzip files from the local **data/** directory).

If you use IDX gz files, place these under **data/**:

Dataset	Required files (relative to repo root)
MNIST	data/train-images-idx3-ubyte.gz data/train-labels-idx1-ubyte.gz data/t10k-images-idx3-ubyte.gz data/t10k-labels-idx1-ubyte.gz
Fashion-MNIST	data/train-images-idx3-ubyte.gz data/train-labels-idx1-ubyte.gz data/t10k-images-idx3-ubyte.gz data/t10k-labels-idx1-ubyte.gz

4. Test recall benchmark (Table S2 / Table 2)

The benchmark reports **macro-averaged recall** across the 10 classes for MNIST and Fashion-MNIST. For single-label multiclass classification on balanced datasets, macro recall is closely related to balanced accuracy.

IntCyt readout used in this repository: Train IntCyt on the training split without labels (self-organization). Assign each terminal compartment/prototype a label by **majority vote** over the labels of training samples routed to that compartment. Route test samples through the learned hierarchy and predict the assigned label. Compute test recall on the held-out test set (both micro and macro are reported by the scripts).

4.1 Run commands (K = 10)

These reproduce the manuscript configuration with K = 10 initial prototypes (arity) and a flat tree (levels = 0):

```
python examples/evaluate_test_recall.py \
  --dataset mnist \
  --runs 5 --seed 0 \
  --iterations 30000 \
  --levels 0 --arity 10

python examples/evaluate_test_recall.py \
  --dataset fashion-mnist \
  --runs 5 --seed 0 \
  --iterations 30000 \
  --levels 0 --arity 10
```

Optional: force IDX loading (no torchvision)

```
python examples/evaluate_test_recall.py \
  --dataset mnist \
  --source idx --data-dir data \
  --runs 5 --seed 0 \
  --levels 0 --arity 10
```

The scripts print mean ± standard deviation across independent seeds and can optionally save a JSON summary via --save-json.

5. Ablation experiments

The ablations in this repository are implemented as separate scripts under **examples/**. They are designed to hold hyperparameters fixed while disabling a single mechanism during learning.

5.1 Residual ablation (disable residual accumulation)

Residual accumulation is disabled during training by temporarily monkey-patching the legacy `Cell.spontaneous_reaction` method so that the residual variable is held at 0 while bookkeeping updates and cytosol clearing still occur.

```
python examples/ablation_residuals.py \
  --dataset both \
  --runs 5 --seed 0 \
  --iterations 30000 \
  --levels 0 --arity 10
```

5.2 Structural/fluctuation ablation (disable fission/fusion; optional no_struct_ops)

This script compares three training conditions: **full** (default schedule; fission+fusion+compose enabled), **no_fluctuations** (fission and fusion disabled; compose kept enabled), and **no_struct_ops** (fission, fusion, and compose all disabled; numeric updates only).

```
python examples/ablation_fluctuations.py \
  --dataset both \
  --runs 5 --seed 0 \
  --iterations 30000 \
  --levels 0 --arity 10
```

5.3 Strong ablation via the standard eval script

```
python examples/evaluate_test_recall.py \
  --dataset mnist \
  --runs 5 --seed 0 \
  --iterations 30000 \
  --levels 0 --arity 10 \
  --disable-struct-ops
```

6. CLI entry points (legacy scripts + ledger)

After installation, the package provides a command-line entry point named **intcyt**. For convenience, the repository also includes a wrapper script `cli.py` so the same subcommands can be executed as `python cli.py ...` from the repo root.

Examples

```
# Legacy training loop (example)
intcyt legacy-main MNIST -iterations 1000 --data-dir data

# Challenge generator (example)
intcyt legacy-challenge fashion-MNIST 20 -right

# Thermodynamic ledger helper (prints a minimal example)
intcyt ledger-info
```

7. License and citation metadata

This repository includes: **LICENSE** (Apache-2.0) and **CITATION.cff** (citation metadata for GitHub / scholarly tooling). Update the CITATION.cff author list and manuscript metadata as needed for your publication workflow.