Topology-Aware Features for Time-Series Classification on UCR ECG5000 and GunPoint

Claire Ming
MSc in Financial Mathematics, HKUST

January 2026

Abstract

This mini-project implements and evaluates topology-aware representations for time-series classification under a fully reproducible experimental protocol. I compare (i) standard hand-crafted statistical/FFT descriptors and (ii) persistent-homology features computed from Takens (sliding-window) embeddings. Experiments are conducted on two UCR benchmark datasets, ECG5000 and GunPoint, using repeated stratified 5-fold cross-validation (3 repeats; 15 folds total). Results are reported with mean, standard deviation, and standard error across folds, and all intermediate artifacts (summary tables, fold-level scores, and configuration JSON) are exported to disk for auditability. The empirical findings show consistent improvements from multi-scale TDA features, with the largest gains on GunPoint. The code for reproducing is all in here: https://github.com/13029991519-MM.

1 Motivation and Scope

Many time-series learning pipelines rely on either raw sequence models (e.g., deep networks) or hand-crafted features (e.g., moments and spectral peaks). Topological data analysis provides an alternative: it maps a signal into a point cloud via delay embeddings and extracts stable, shape-driven descriptors via persistent homology [4, 3]. The goal of TDA-TS Mini-Project is intentionally modest but research-aligned:

- implement a robust, end-to-end, topology-aware feature pipeline in Python (compatible with Python 3.14 on Windows),
- benchmark against a simple baseline feature set,
- use a transparent, leakage-safe evaluation protocol with uncertainty reporting, and
- provide reproducible outputs suitable for a research note or GitHub repository.

2 Datasets and Local Data Layout

I use two datasets from the UCR time-series classification archive [2]:

- **ECG5000**: electrocardiogram waveform classification, $N = 5000$, length $T = 140$, 5 classes.
- **GunPoint**: motion/gesture time series, $N = 200$, length $T = 150$, 2 classes.

Local paths (executed). Data were pre-downloaded and loaded locally from:

- `C:\Users\MM\Desktop\1\data\ECG5000`
- `C:\Users\MM\Desktop\1\data\GunPoint`

Expected files. Each dataset directory contains:

`<DATASET>_TRAIN.tsv` and `<DATASET>_TEST.tsv`

where the first column is the class label and the remaining columns are the time points.

3 Methodology

3.1 Baseline Feature Set

Given a univariate series $x \in \mathbb{R}^T$, I compute a fixed-length vector of hand-crafted descriptors:

$$\phi_{\text{base}}(x) = \big[\mu, \sigma, \text{skew}, \text{kurt}, \min, \max, q_{0.05}, q_{0.25}, q_{0.50}, q_{0.75}, q_{0.95}, \text{energy}, \rho_1, \text{fftpeak}\big],$$

where energy is $\frac{1}{T}\sum_{t=1}^{T} x_t^2$, ρ_1 is lag-1 correlation, and `fftpeak` is the maximum FFT magnitude excluding DC.

3.2 Topology-Aware Feature Pipeline

Takens / sliding-window embedding. I map x into a point cloud in $\mathbb{R}^d$ using delay coordinates:

$$\Psi_{d,\tau}(x) = \big\{\big(x_t, x_{t+\tau}, \ldots, x_{t+(d-1)\tau}\big) : t = 1, \ldots, T-(d-1)\tau\big\},$$

optionally sub-sampled by a stride parameter s to reduce computational cost.

Persistent homology via Vietoris–Rips filtration. On the embedded point cloud $\Psi_{d,\tau}(x)$ I compute Vietoris–Rips persistence diagrams in homology dimensions H_0 and H_1 using Ripser [1].

Diagram vectorization (summary statistics). For each homology dimension $k \in \{0, 1\}$, given diagram $D_k = \{(b_i, d_i)\}$, I compute lifetimes $\ell_i = \max(d_i - b_i, 0)$ (excluding infinite deaths) and use the fixed-length summary:

$$\phi_k(D_k) = \big[\, \#\ell, \sum_i \ell_i, \text{mean}(\ell), \text{std}(\ell), \max(\ell), \text{entropy}(\ell) \,\big], \quad \text{entropy}(\ell) = -\sum_i p_i \log(p_i),\ p_i = \frac{\ell_i}{\sum_j \ell_j}.$$

The final single-scale TDA feature is

$$\phi_{\text{tda}}(x) = \big[\phi_0(D_0), \phi_1(D_1)\big] \in \mathbb{R}^{12}.$$

Multi-scale TDA features. To better capture structure across scales, I concatenate features computed from multiple embedding configurations:

$$\phi_{\text{ms-tda}}(x) = \big[\phi_{\text{tda}}^{(d_1,\tau_1,s_1)}(x),\ \phi_{\text{tda}}^{(d_2,\tau_2,s_2)}(x),\ \phi_{\text{tda}}^{(d_3,\tau_3,s_3)}(x)\big].$$

In the executed run, the multi-scale set is:

$$(d, \tau, s) \in \{(4, 1, 3), (5, 2, 3), (6, 3, 3)\},$$

and the single-scale setting is $(d, \tau, s) = (5, 2, 3)$.

3.3 Models

I evaluate three model pipelines:

- **Baseline(LogReg)**: $\phi_{\text{base}}(x)$ + standardization + logistic regression.
- **TDA single(LogReg)**: $\phi_{\text{tda}}(x)$ + standardization + logistic regression.
- **TDA multiscale(SVM)**: $\phi_{\text{ms-tda}}(x)$ + standardization + RBF-SVM ($C = 5$, $\gamma = \text{scale}$).

3.4 Evaluation Protocol and Uncertainty

Cross-validation. I use `RepeatedStratifiedKFold` with 5 splits and 3 repeats (15 test folds total). Stratification prevents class proportion drift between folds.

Metrics. I report:

- Accuracy
- Macro-F1 (uniform average across classes), which is sensitive to minority-class performance.

Uncertainty reporting. For each metric, I compute mean and standard deviation across the 15 folds. I also report the standard error (SE):

$$\text{SE} = \frac{\text{std}}{\sqrt{n}}, \qquad n = 15.$$

All fold-level values are saved to `fold_scores.csv` for audit and downstream statistical tests.

4 End-to-End Workflow

Figure 1 summarizes the executable pipeline.

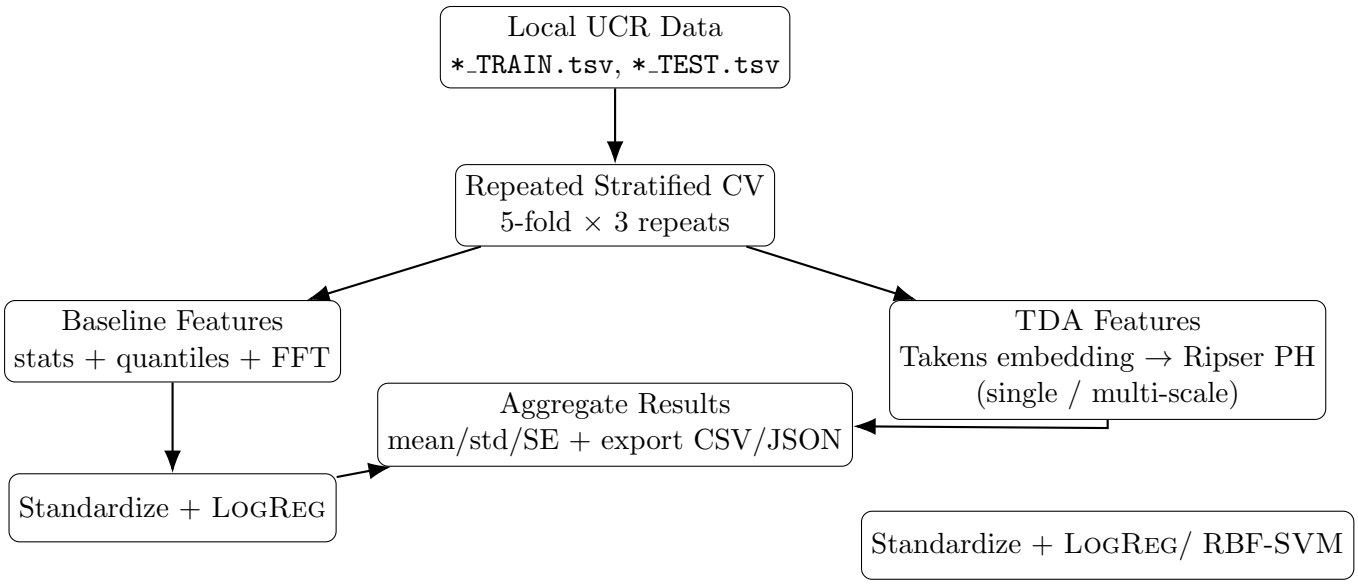

Figure 1: Reproducible workflow of TDA-TS Mini-Project.

5 Results

This section reports the exact executed outputs provided in the console log. Tables 1 and 2 correspond to `outputs/<dataset>/summary.csv`.

5.1 ECG5000 (5 classes; $N = 5000$, $T = 140$)

Table 1: ECG5000 results (15 folds). Values are mean ± std with SE reported separately.

Model	Acc mean	Acc std	Acc SE	Macro-F1 mean	F1 std	F1 SE
TDA_multiscale(SVM)	0.937 333	0.005 538	0.001 430	0.553 186	0.042 184	0.010 892
Baseline(LogReg)	0.928 000	0.005 542	0.001 431	0.493 192	0.034 160	0.008 820
TDA_single(LogReg)	0.919 400	0.007 500	0.001 937	0.435 702	0.034 074	0.008 798

Interpretation. The multi-scale TDA representation improves mean accuracy relative to the baseline, suggesting that geometric/topological structure extracted from delay embeddings carries discriminative information beyond hand-crafted moments. Meanwhile, the gap between accuracy and macro-F1 indicates that class imbalance or minority-class difficulty may limit uniform performance across the 5 classes; macro-F1 penalizes poor minority-class recall/precision even when overall accuracy remains high.

5.2 GunPoint (2 classes; $N = 200$, $T = 150$)

Table 2: GunPoint results (15 folds). Values are mean ± std with SE reported separately.

Model	Acc mean	Acc std	Acc SE	Macro-F1 mean	F1 std	F1 SE
TDA_multiscale(SVM)	0.961 667	0.029 681	0.007 664	0.961 622	0.029 731	0.007 676
TDA_single(LogReg)	0.936 667	0.036 433	0.009 407	0.936 548	0.036 536	0.009 433
Baseline(LogReg)	0.913 333	0.039 940	0.010 313	0.913 132	0.040 019	0.010 333

Interpretation. On GunPoint, topology-aware features yield substantial gains over the baseline under the same evaluation protocol. Because GunPoint is binary, accuracy and macro-F1 are closely aligned, making the improvements easier to interpret and directly attributable to feature representation differences.

6 Reproducibility and Artifacts

6.1 Execution Environment (as run)

- OS: Windows (PowerShell execution shown in logs)
- Python: 3.14 (explicit requirement)
- Key dependency for persistence: `ripser` [1]

6.2 Outputs (as generated)

For each dataset, the script produces:

- `outputs/<dataset>/summary.csv` — table-level metrics (mean/std/SE)
- `outputs/<dataset>/fold_scores.csv` — fold-level scores for statistical analysis
- `outputs/<dataset>/config.json` — exact evaluation parameters (CV seed/splits/repeats; embedding configs)

6.3 Local Data Loader Details

The loader reads `<dataset>_TRAIN.tsv` and `<dataset>_TEST.tsv`, concatenates them, then parses:

$$y = \text{first column}, \qquad X = \text{remaining columns}.$$

TSV parsing is attempted first, with whitespace fallback to support `.txt` variants.

7 Discussion: Why Multi-Scale TDA Helps

Two design choices are salient:

1. **Delay embedding induces geometry.** Takens embedding transforms temporal structure into geometric structure in $\mathbb{R}^d$ [4]. This makes shape-based invariants (e.g., loops in H_1) accessible to analysis.
2. **Scale diversity improves robustness.** Varying (d, τ) changes the reconstructed attractor geometry and the sensitivity to local noise. Concatenating multiple configurations can reduce brittleness and capture complementary signal properties, aligning with prior signal-analysis uses of sliding windows + persistence [3].

8 Limitations and Extensions

This mini-project is intentionally constrained; several extensions would make it more "paper-like":

- **Class-imbalance diagnostics.** For ECG5000, report class counts, balanced accuracy, per-class F1, and confusion matrices.
- **Nested CV / tuning.** Jointly tune (d, τ, s) and SVM hyperparameters under nested CV to avoid optimistic bias.
- **Richer diagram vectorizations.** Replace summary statistics with persistence images/landscapes (if dependency constraints allow), and compare representational trade-offs.
- **Statistical testing.** Use fold-level scores to run paired tests (e.g., Wilcoxon signed-rank) between representations.

9 Conclusion

I implemented a reproducible, Python-3.14-compatible topology-aware time-series classification pipeline and benchmarked it on two UCR datasets under a leakage-safe evaluation protocol. Multi-scale persistent-homology features consistently improved classification performance over a hand-crafted baseline, particularly on GunPoint. The exported fold-level artifacts support transparent reporting and future statistical analyses.

A Appendix A: Executed Console Output (Verbatim)

The following console output was produced by running:

`python run_tda_ucr_miniproject_local.py`

```
PS C:\Users\MM\Desktop\1> C:\Users\MM\AppData\Local\Programs\Python\Python314\python.
    exe c:/Users/MM/Desktop/1/run_tda_ucr_miniproject_local.py

=== Loading LOCAL dataset: ECG5000 ===
Data dir listing: ['ECG5000_TEST.tsv', 'ECG5000_TRAIN.tsv', 'README.md']
X shape: (5000, 140) y shape: (5000,) classes: ['1' '2' '3' '4' '5']

--- CV running: Baseline(LogReg) ---

--- CV running: TDA_single(LogReg) ---

--- CV running: TDA_multiscale(SVM) ---

=== Summary ===
dataset model acc_mean acc_std acc_se f1_mean f1_std f1_se n_folds_total
ECG5000 TDA_multiscale(SVM) 0.937333 0.005538 0.001430 0.553186 0.042184 0.010892 15
ECG5000 Baseline(LogReg) 0.928000 0.005542 0.001431 0.493192 0.034160 0.008820 15
ECG5000 TDA_single(LogReg) 0.919400 0.007500 0.001937 0.435702 0.034074 0.008798 15

Saved:
  outputs\ECG5000\summary.csv
  outputs\ECG5000\fold_scores.csv
  outputs\ECG5000\config.json
Runtime: 152.2s

=== Loading LOCAL dataset: GunPoint ===
Data dir listing: ['GunPoint_TEST.tsv', 'GunPoint_TRAIN.tsv', 'README.md']
X shape: (200, 150) y shape: (200,) classes: ['1' '2']

--- CV running: Baseline(LogReg) ---

--- CV running: TDA_single(LogReg) ---

--- CV running: TDA_multiscale(SVM) ---

=== Summary ===
 dataset model acc_mean acc_std acc_se f1_mean f1_std f1_se n_folds_total
GunPoint TDA_multiscale(SVM) 0.961667 0.029681 0.007664 0.961622 0.029731 0.007676 15
GunPoint TDA_single(LogReg) 0.936667 0.036433 0.009407 0.936548 0.036536 0.009433 15
GunPoint Baseline(LogReg) 0.913333 0.039940 0.010313 0.913132 0.040019 0.010333 15

Saved:
  outputs\GunPoint\summary.csv
  outputs\GunPoint\fold_scores.csv
  outputs\GunPoint\config.json
Runtime: 5.5s
```

B Appendix B: Exact Hyperparameters (as executed)

- CV: 5 splits, 3 repeats, seed 42 ($n = 15$ folds total)

- Baseline: StandardScaler + LogisticRegression(max_iter=800)
- TDA single: (dim=5, tau=2, stride=3), maxdim=1; StandardScaler + LogisticRegression(max_iter=800)
- TDA multi-scale: (4,1,3), (5,2,3), (6,3,3), maxdim=1; StandardScaler + RBF-SVM(C=5, gamma=scale)

References

[1] Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 2021.

[2] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The UCR time series archive, 2018.

[3] Jose A. Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 2015.

[4] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer, 1981.