GOTabPFN: From Feature Ordering to Compact Tokenization for Tabular Foundation Models on High-Dimensional Data

Al Zadid Sultan Bin Habib[1] Md Younus Ahamed[2] Prashnna Kumar Gyawali[3]
Gianfranco Doretto[4] Donald A. Adjeroh[5]

[1,2,3,5]Lane Department of Computer Science and Electrical Engineering,
West Virginia University, Morgantown, WV 26506, USA

[4]Scientific Computing and Imaging Institute & Department of Biomedical Informatics,
University of Utah, Salt Lake City, UT 84112, USA

Corresponding author: [1]ah00069@mix.wvu.edu

ICML 2026

Why GOTabPFN?

HDLSS tabular data

High-Dimensional, Low-Sample Size means:

$$m \gg n$$

where m is the number of features and n is the number of samples.

TabPFN bottleneck

TabPFN-style models are strong general-purpose predictors, but direct use becomes difficult when m is very large.

$$m \rightarrow \text{too many tokens}$$

Goal: make frozen TabPFN-style prediction practical for high-dimensional tabular data by turning many raw features into fewer structured meta-features.

Core Idea: Order, Compress, Predict

1. Order

Use GO-LR to place related features close together on a structured feature axis.

2. Compress

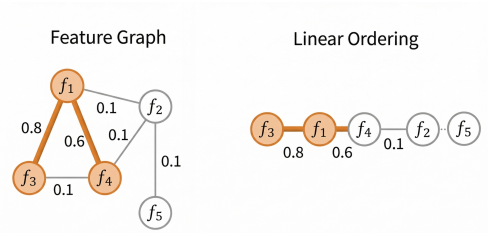
Use NSC to pool local ordered neighborhoods into compact meta-features.

3. Predict

Use the compact representation with a frozen TabPFN-2.5 prediction head.

Pipeline: Raw high-dimensional table $\rightarrow$ GO-LR $\rightarrow$ NSC $\rightarrow$ Frozen TabPFN-2.5

GO-LR: Graph-guided Ordering with Local Refinement



Goal: Place connected nodes on a line such that the most strongly related nodes are close to each other

Input

Build a weighted graph where nodes are features and edges encode feature relationships.

Goal

Place strongly related features close together on a one-dimensional feature axis.

Output: an ordered feature sequence for NSC compression.

GO-LR: Optimization View

Local objective

GO-LR minimizes a MinLA-style dispersion cost:

$$D_G(\pi) = \sum_{(i,j)\in E} w_{ij}|\pi(i) - \pi(j)|$$

Large-weight feature pairs are penalized more when placed far apart.

Practical solver

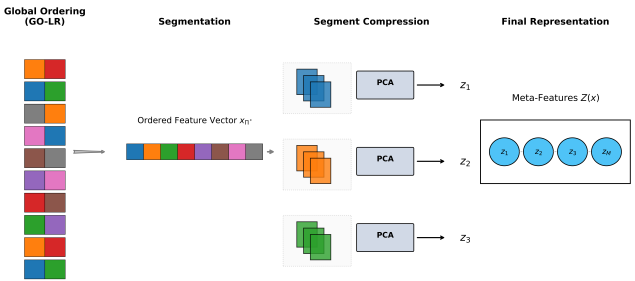
GO-LR uses:

- TSP-path-style initialization
- direction selection
- local adjacent-swap refinement

Result: one global order Π^*.

The ordered axis is passed to NSC, where nearby features are compressed together.

NSC: Neuro-Inspired Subunit Compression



Segmentation

After GO-LR, the ordered feature vector is partitioned into contiguous local segments.

Compression

Each segment is compressed into one meta-feature, producing a smaller structured representation.

Output: a compact meta-feature vector $Z(x) = (z_1, z_2, \ldots, z_M)$.

NSC: From Ordered Features to Meta-Features

Ordered input

After GO-LR, features lie on a structured axis:

$$x^{\Pi} = \left(x_{\Pi^*(1)}, \ldots, x_{\Pi^*(m)}\right)$$

NSC groups nearby ordered features into local segments

$$\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_M.$$

Segment compression

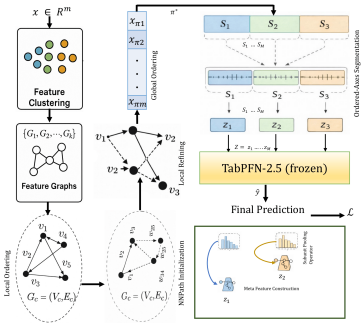
For each segment t, one meta-feature is computed as:

$$z_t(x) = \left(u_t(x) - \mu_t\right)^\top v_t$$

where $u_t(x) = x^{\Pi}_{\mathcal{S}_t}$. This reduces m raw features to M compact meta-features, with $M \ll m$.

These compact tokens are then passed to the frozen TabPFN-2.5 prediction head.

End-to-End GOTabPFN Pipeline



GO-LR
Orders raw features into a structured feature axis.

NSC
Compresses local ordered neighborhoods into meta-features.

TabPFN-2.5
Predicts from compact tokens using a frozen backbone.

Design: adapt the input, not the backbone.

Results: HDLSS and Cross-Domain Performance

8/8
HDLSS wins

1.00
HDLSS rank
average rank across core tasks

7/8
Cross-domain wins

1.12
Expanded rank
average rank over 16 datasets

Core HDLSS setting

GOTabPFN leads on all 8 core HDLSS datasets under 5×5 repeated cross-validation.

Beyond HDLSS

The same ordering-compression interface generalizes to image-derived, biological, text-like, and sensor features.

Takeaway: compact ordered meta-features make frozen TabPFN-style prediction practical for high-dimensional data.

GO-LR: Ordering Evidence

10.07s
Colon runtime
faster than heavier ordering baselines

1.4743×10^{10}
Colon MinLA cost
lowest among compared ordering methods

8.14×10^{11}
Cell Cycle cost
lower than SA: 8.51×10^{11}

Against metaheuristics

Compared with Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Christofides, GO-LR gives the best Colon MinLA cost.

Why it matters

Lower MinLA-style cost means related features are placed closer before NSC local compression.

Message: GO-LR is fast, objective-aligned, and useful before compact tokenization.

Takeaway, Resources, and Acknowledgment

GOTabPFN makes frozen TabPFN-style prediction practical for high-dimensional tabular data by combining **feature ordering** with **compact tokenization**.

Project links

ICML: `icml.cc/virtual/2026/poster/62523`

Project: `zadidhabib.com/gotabpfn.html`

GitHub: `github.com/zadid6pretam/GOTabPFN`

Try or ask

Demo: ZeroGPU `zadid6pretam/GOTabPFN`; CPU Backup `zadid6pretam/GOTabPFN_CPU`

Install: `pip install gotabpfn`

Contact: `ah00069@mix.wvu.edu`

Acknowledgment: This work was supported in part by the U.S. National Science Foundation (NSF) Awards #1920920, #2125872, and #2223793. We thank the anonymous ICML reviewers for their constructive feedback.

Thank You!

Questions?

github.com/zadid6pretam/GOTabPFN/issues

ah00069@mix.wvu.edu

zadidhabib.com/gotabpfn.html