ENHANCING DEEP TABULAR MODELS WITH GBDT-GUIDED PIECEWISE-LINEAR EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular data remains central to many scientific and industrial applications. Recently, deep learning models are emerging as a powerful tool for tabular data prediction, outperforming traditional methods such as Gradient Boosted Decision Trees (GBDTs). Despite this success, the fundamental challenge of feature heterogeneity still remains. Unlike in image or text modalities where features are semantically homogeneous, each tabular feature often carries a distinct semantic meaning and distribution. A common strategy to address the heterogeneity is to project features into a shared high-dimensional vector space. Among the various feature types in tabular data, categorical features are effectively embedded via embedding bags, which assign a learnable vector to each unique category. In contrast, effective embeddings for numerical features remain underexplored. In this paper, we argue that piecewise-linear functions are well suited to modeling the irregular and high-frequency patterns often found in tabular data, provided that breakpoints are carefully chosen. To this end, we propose GBDT-Guided Piecewise-Linear (GGPL) embeddings, a method comprising breakpoints initialization using GBDT split thresholds, stable breakpoint optimization using reparameterization, and stochastic regularization via breakpoints deactivation. Thorough evaluation on 46 datasets shows that applying GGPL to a range of state-of-the-art tabular models consistently improves performance—especially on regression tasks—or at least matches their native numerical embeddings. Together with its negligible overhead, this suggests that GGPL can serve as a practical default numerical embedding for future tabular architectures. The code is available in the supplementary material.

1 INTRODUCTION

Tabular data, characterized by its structured format of rows and columns, remains the most common data modality in a vast array of scientific and industrial domains, from healthcare and finance to e-commerce and climate science (Shwartz-Ziv & Armon, 2022; Somvanshi et al., 2024). Its practical importance across numerous domains has established predicting a target column from a set of observed features as a central problem within the field of machine learning, attracting extensive research (Gorishniy et al., 2021; Lee et al., 2024; Eo et al., 2025; Hollmann et al., 2025; Lee et al., 2025; Ye et al., 2025).

Deep learning architectures for tabular data (Yan et al., 2023; Gorishniy et al., 2025; Hollmann et al., 2025; Ye et al., 2025) are increasingly outperforming Gradient Boosted Decision Trees (GBDTs; Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018) on many benchmarks, which have long been the dominant approach in the field. This shift is driven by the ability of deep models to capture complex, non-linear feature interactions, reducing the dependence on manual feature engineering. In real-world applications with large and diverse datasets, deep learning models stand out for their high prediction accuracy and efficient inference.

Nevertheless, effectively handling heterogeneous tabular features remains a key challenge for deep models (Gorishniy et al., 2021). Unlike images or text, where features (e.g., pixels or words) are semantically homogeneous, tabular columns often represent fundamentally different concepts whose scales and distributions can vary widely. A common strategy to address this heterogeneity is to map features into a shared high-dimensional vector space, allowing subsequent layers to function

effectively (e.g., matrix multiplication in an MLP or the attention mechanism in a Transformer). Moreover, this projection into a high-dimensional space empowers the network to learn complex, non-linear feature interactions. While there is a well-established practice for categorical features using embedding bags (Mikolov et al., 2013; Guo et al., 2017), a standard methodology for embedding numerical features has yet to emerge.

Embedding a numerical feature can be viewed as learning a continuous mapping from $\mathbb{R}$ to $\mathbb{R}^d$. A variety of approaches have been explored for this purpose, including Multi-Layer Perceptrons (MLPs; Gorishniy et al., 2022; Wu et al., 2024), Fourier features (Gorishniy et al., 2022; Sergazinov et al., 2025), and piecewise-linear functions (Gorishniy et al., 2022). Among these, we find piecewise-linear functions well suited for embedding tabular data. Real-world tabular datasets often exhibit irregular and non-smooth feature–target relationships. A key aspect of these relationships is the presence of high-frequency components, which are critical for accurate prediction (Grinsztajn et al., 2022). However, MLPs exhibit a spectral bias that prevents them from effectively capturing such high-frequency target functions (Rahaman et al., 2019). Although Fourier features have addressed this limitation in computer vision (Tancik et al., 2020; Mildenhall et al., 2021), their effectiveness is reduced in the tabular domain. Accurate modeling with Fourier features requires an appropriate choice of frequency components, but the heterogeneity of tabular data implies that a different set of frequencies may be optimal for each feature. While the expressiveness of piecewise-linear functions in modeling irregular and high-frequency patterns also depends critically on the appropriate placement of breakpoints (Hastie et al., 2009), our method addresses this challenge by effectively selecting suitable breakpoints for each numerical feature.

In this paper, we address the challenge of breakpoint positioning within the prior piecewise-linear embedding (Gorishniy et al., 2022). To this end, we propose the GBDT-Guided Piecewise-Linear (GGPL) embedding that focuses on three essential components: initialization, optimization, and regularization. The properties of tabular data and their target functions make each of these components important. Effectively modeling the irregular and non-smooth function requires precise breakpoint placement, which makes both effective initialization and stable optimization essential. Additionally, the task of tabular prediction exhibits an inherent tendency to overfit (Kadra et al., 2021) and lacks the inherent invariances (e.g., spatial invariance in images) that facilitate data augmentation. This makes robust regularization essential. To address these challenges, we propose the following contributions, which are described in detail in Section 3.

1. We initialize the breakpoints using the split thresholds of the largest gains from an XGBoost model, effectively leveraging the well-established strength of GBDT.
2. We reparameterize the optimization of breakpoints into a stable process by optimizing the ratios of piece lengths on a probability simplex, which guarantees valid breakpoint positions throughout training.
3. We propose a regularization technique that stochastically deactivates breakpoints during training, encouraging similarity between adjacent linear pieces to prevent overfitting.

In practice, we train a default XGBoost once to obtain the split thresholds, which minimizes training overhead and shows no statistically significant difference from a hyperparameter-tuned one. In addition, all proposed methods are training-only and not applied at inference, so there is no additional inference-time overhead compared with prior piecewise-linear embedding methods (Gorishniy et al., 2022).

We validate our proposed method through extensive experiments on the 46 datasets from Gorishniy et al. (2025), spanning a wide range of sizes and domains. Our embedding demonstrates statistically significant improvements in two settings: (i) when applied to state-of-the-art deep tabular models (Yan et al., 2023; Gorishniy et al., 2025; Ye et al., 2025) and (ii) when compared with existing numerical embedding methods (Gorishniy et al., 2022; Li et al., 2024). Notably, our best-performing model achieves the top average rank across all datasets. Moreover, on small-sized datasets where tabular foundation models are available, our GGPL-enhanced models perform competitively, sometimes surpassing TabPFN (Hollmann et al., 2025). The detailed experimental setup and results are presented in Section 4. A subsequent analysis, including ablation studies, statistical tests, and performance analysis across dataset characteristics, is provided in Section 5.

We validate our proposed method through extensive experiments on the 46 datasets from Gorishniy et al. (2025), spanning a wide range of sizes and domains. Our embedding demonstrates statis-

tically significant improvements in two settings: (i) when applied to state-of-the-art deep tabular models (Yan et al., 2023; Gorishniy et al., 2025; Ye et al., 2025) and (ii) when compared with existing numerical embedding methods (Gorishniy et al., 2022; Li et al., 2024). These gains are more consistent on regression tasks, and on classification they are relatively small but at least match the performance of the native numerical embeddings. Combined with the negligible training and inference overhead of GGPL, these results suggest that it can serve as a practical default numerical embedding for future tabular architectures. Notably, our best-performing model achieves the top average rank across all datasets. Moreover, on small-sized datasets where tabular foundation models are available, our GGPL-enhanced models perform competitively, sometimes surpassing TabPFN (Hollmann et al., 2025). The detailed experimental setup and results are presented in Section 4, and subsequent analysis, including ablation studies, statistical tests, and performance analysis across dataset characteristics, is provided in Section 5.

2 Related Work

2.1 Tabular Prediction Models

GBDTs like XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) have long been the state-of-the-art. They build an ensemble of weak decision trees sequentially and are known for their ability to handle sparse, heterogeneous data and capture complex feature interactions. However, their performance is being surpassed by deep learning architectures, which can be largely classified into two main categories: foundation models and task-specific models.

Foundation models like TabPFN (Hollmann et al., 2025) can be applied to various downstream tasks without further parameter tuning, demonstrating remarkable performance on small-scale problems. This is achieved by pre-training a large model on millions of synthetic datasets and leveraging in-context learning at inference-time. However, their quadratic time complexity with respect to the number of training samples restricts their applicability to large-scale datasets.

Task-specific models, while chronologically preceding foundation models, remain a crucial and practical type as they are free from the scalability issues of foundation models. To improve the performance of task-specific models, various approaches have been explored, such as improving backbone architectures, enhancing embedding methods, and incorporating the strengths of GBDTs.

2.2 Improvements in Task-Specific Models

One primary line of research is designing specialized backbone architectures. These include MLP-based models, which have shown that even simple architectures can achieve top-tier performance when combined with proper regularization and ensemble techniques (Kadra et al., 2021; Holzmüller et al., 2024; Gorishniy et al., 2025); Transformer-based models that adapt the self-attention mechanism to learn complex interactions among heterogeneous features (Gorishniy et al., 2021; Yan et al., 2023); and retrieval-based models that make predictions by retrieving similar instances from the training set (Gorishniy et al., 2024; Ye et al., 2025).

Another key line of research involves improving embedding methods for input features. MLP-based embedding methods apply a feature-specific MLP to map each scalar value to an embedding vector (Guo et al., 2017; Gorishniy et al., 2022; Wu et al., 2024). However, MLPs have a spectral bias towards learning smooth functions (Rahaman et al., 2019), which may not be optimal for the often irregular relationships in tabular data. Inspired by their success in computer vision (Mildenhall et al., 2021), Fourier embeddings have also been used for numerical embeddings (Gorishniy et al., 2022; Sergazinov et al., 2025), but their effectiveness is limited in the tabular domain due to feature heterogeneity, which makes it difficult to find suitable frequency components for all features. Piecewise-linear embedding methods partition a feature's range into a set of bins and learn a linear function within each bin. Gorishniy et al. (2022) introduce piecewise-linear embedding and propose two methods for initializing breakpoints: a quantile-based approach, which places breakpoints based on the input distribution, and a target-aware approach, which trains a decision tree for each input feature to predict the target and uses the resulting thresholds. However, these breakpoints are fixed and not optimized during training. Further, the feature-wise decision tree-based approach prevents considering complex feature interactions when placing breakpoints.

A third line of work incorporates the strengths of GBDTs into deep learning models. Some methods introduce modules to mimic the thresholding behavior of decision trees within a neural network (Popov et al., 2020; Katzir et al., 2021). Another method uses GBDTs to calculate feature frequencies to determine the selection ratio of feature gates (Li et al., 2024). Our approach, rather than mimicking GBDTs, leverages their efficiency and accuracy to initialize the breakpoints of the piecewise-linear embedding using split thresholds with the largest score gain.

3 Proposed Method

3.1 Background: Piecewise-Linear Encoding

To enhance the representational capacity of tabular models, prior work has proposed embedding scalar numerical features into higher-dimensional vector spaces using piecewise-linear encoding (PLE; Gorishniy et al., 2022). Formally, for i-th numerical feature $x_i \in \mathbb{R}$, the input range is partitioned into K_i+1 disjoint intervals $[t_k^{(i)}, t_{k+1}^{(i)})$ for $k = 0, \ldots, K_i$, and the encoding is computed as $\mathrm{PLE}(x_i) = [e_0^{(i)}, \ldots, e_{K_i}^{(i)}] \in \mathbb{R}^{K_i+1}$:

$$e_k^{(i)} = \begin{cases} 0, & x_i < t_k^{(i)} \text{ and } k > 0 \\ 1, & x_i \geq t_{k+1}^{(i)} \text{ and } k < K_i \\ \frac{x_i - t_k^{(i)}}{t_{k+1}^{(i)} - t_k^{(i)}}, & \text{otherwise} \end{cases} \tag{1}$$

The conditions on k ($k > 0$ and $k < K_i$) handle linear extrapolation when input values x_i fall outside the range. This encoding produces a continuous and order-aware vector representation of scalar inputs, which can serve as an effective replacement for the original input value in downstream architectures. However, prior implementations of PLE use fixed breakpoints (e.g., quantile or target-aware), which remain static during training. We instead initialize breakpoints using GBDT splits and jointly optimize them via a differentiable reparameterization. This enables the embedding to adapt to high-frequency or irregular patterns. We further apply stochastic regularization to improve generalization.

3.2 GBDT-Guided Piecewise-Linear Embedding

The embedding function in a tabular neural network, $\phi_i : \mathbb{R} \to \mathbb{R}^d$, maps i-th feature to a shared high-dimensional space. In this paper, we use PLE to model $\phi_i(x_i)$ through a piecewise-linear curve as follows.

$$\phi_i(x_i) = \left[\mathbf{w}_0^{(i)}, \mathbf{w}_1^{(i)}, \ldots, \mathbf{w}_{K_i}^{(i)}\right] \begin{bmatrix} e_0^{(i)} \\ e_1^{(i)} \\ \vdots \\ e_{K_i}^{(i)} \end{bmatrix} + \mathbf{b}^{(i)} \tag{2}$$

$\phi_i(x_i)$ maps each scalar input to a point on a continuous piecewise-linear curve in $\mathbb{R}^d$. Specifically, when $x_i = t_k^{(i)}$, the embedding becomes a vertex of the curve given by $\mathbf{v}_k^{(i)} = \mathbf{b}^{(i)} + \sum_{j=0}^{k-1} \mathbf{w}_j^{(i)}$. When $x_i \in [t_k^{(i)}, t_{k+1}^{(i)})$, the embedding lies on the line segment connecting $\mathbf{v}_k^{(i)}$ and $\mathbf{v}_{k+1}^{(i)}$. When $x_i < t_0^{(i)}$ or $x_i \geq t_{K_i+1}^{(i)}$, the embedding extrapolates linearly based on the first or last segment, respectively. An example of $\phi_i(x_i)$ is illustrated in Figure 1.

The main challenge is to determine the optimal positions of $t_k^{(i)}$ and their corresponding $\mathbf{v}_k^{(i)}$, which are defined by $\mathbf{w}_k^{(i)}$ and $\mathbf{b}^{(i)}$. The GGPL embedding tackles this through three components:

1. GBDT-guided initialization which determines the initial values of $t_k^{(i)}$,
2. simplex-based reparameterization for stable optimization of $t_k^{(i)}$,
3. stochastic breakpoint regularization to mitigate overfitting.

All parameters including $\mathbf{w}_k^{(i)}$ and $\mathbf{b}^{(i)}$ are optimized via standard backpropagation.

3.3 GBDT-GUIDED BREAKPOINT INITIALIZATION

Our initialization method determines the initial values of $t_k^{(i)}$ using GBDTs trained on the data. Specifically, we adopt the feature threshold values used for splitting nodes in the GBDT as the initial locations for the breakpoints. By leveraging GBDTs' well-established strength, this approach identifies the most effective splits based on their gain scores. This process allocates fewer breakpoints to less important features, reducing the risk of overfitting by saving parameters. Algorithmic details are provided in Appendix A.1.

To keep the pipeline lightweight, we obtain split thresholds from a single XGBoost model with default hyperparameters trained on each dataset. Moreover, because the GBDT is used only to initialize breakpoints, it incurs no inference-time cost. Across the 46 datasets, the tuned XGBoost initializer does not yield statistically significant improvements over the default ($p \approx 0.07$), indicating that the default setting is adequate as a practical choice. Nonetheless, accuracy can still be pushed further through additional tuning of the initializer when desired. Further details are provided in Appendix A.2, including comparisons to tuned XGBoost and alternative GBDT initializers.

3.4 STABLE SIMPLEX-BASED OPTIMIZATION

Directly training $t_k^{(i)}$ may break ordering constraints ($t_{k-1}^{(i)} \leq t_k^{(i)}$) and is prone to division-by-zero errors. To optimize $t_k^{(i)}$ stably, we reparameterize the problem as the following.

We first normalize the position of each $t_k^{(i)}$ as $r_k^{(i)} = (t_k^{(i)} - \min(X_i))/(\max(X_i) - \min(X_i))$, where X_i is the set of values for i-th feature in the training set. Then, we define the k-th proportion as $\pi_k^{(i)} = r_k^{(i)} - r_{k-1}^{(i)}$. The vector of proportions $\boldsymbol{\pi}^{(i)} = [\pi_1^{(i)}, \ldots, \pi_{K_i+1}^{(i)}]$ forms a point on the K_i-dimensional probability simplex (Δ^{K_i}), which is the output of the softmax function.

$$\boldsymbol{\pi}^{(i)} = \text{softmax}(\mathbf{z}^{(i)}) \tag{3}$$

Optimizing unconstrained logits $\mathbf{z}^{(i)} \in \mathbb{R}^{K_i+1}$ instead of $t_k^{(i)}$ guarantees that the breakpoints remain ordered and prevents them from collapsing within the feature's range.

3.5 STOCHASTIC BREAKPOINT REGULARIZATION

Since there are no smoothness constraints between adjacent embedding vectors, the learned function exhibits high tortuosity, which increases the risk of overfitting. To mitigate this, we introduce a regularization technique analogous to dropout that encourages similarity between adjacent embedding vectors. During each training forward pass, we randomly deactivate a fraction of $t_k^{(i)}$ for $k = 1, \ldots, K_i$ with probability p.

Figure 2 provides a visual example of the stochastic regularization technique. When $t_k^{(i)}$ is deactivated, its corresponding $\mathbf{v}_k^{(i)}$ is ignored, and a new linear piece is formed between $\mathbf{v}_{k-1}^{(i)}$ and $\mathbf{v}_{k+1}^{(i)}$ (or the nearest active breakpoints if multiple consecutive breakpoints are deactivated). The embedding should remain consistent even if some breakpoints are deactivated. This encourages the model

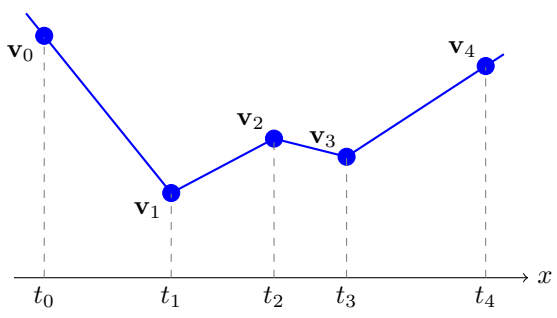


Figure 1: A piecewise-linear embedding that maps a scalar feature into a high-dimensional space, defined by a set of breakpoints ($t_k \in \mathbb{R}$) and their corresponding embedding vectors ($\mathbf{v}_k \in \mathbb{R}^d$).

to learn a smoother function, which is beneficial for regression but can be detrimental for classification tasks that rely on sharp decision boundaries. Consequently, we apply this regularization only to regression tasks. At inference-time, all breakpoints are activated ($p = 0$), and unlike dropout, no scaling of the embeddings is required. We analyze the effect of this regularization for regression and classification in Appendix A.3.

4 EXPERIMENTS

4.1 DATASETS

We conduct a comprehensive evaluation on the benchmark of 46 datasets previously used in Gorishniy et al. (2025). These datasets span a wide range of tabular tasks, with sample sizes from a few thousand to over a million and feature counts up to nearly a thousand—reflecting the scale and complexity of real-world applications. The characteristics of these datasets are summarized in Table 1, with further details available in Appendix B.

Table 1: Overview of the 46 benchmark datasets, categorized by task, sample size, and feature-to-sample ratio.

Category	**Criteria**	**Count**
Total		46
Task	Classification	18
	Regression	28
Sample size	Small ($\leq 30k$)	24
	Large ($> 30k$)	22
Feature-to-sample ratio	High (> 0.001)	21
	Low (≤ 0.001)	25

4.2 BASELINE MODELS

To evaluate the effectiveness and versatility of our GGPL embedding, we integrate it into three state-of-the-art deep tabular models and a baseline MLP, with each representing a different architectural paradigm. For each model, we then compare the performance of the original version against its GGPL-enhanced counterpart. The selected models are MLP, T2G-Former, ModernNCA, and TabM, and we provide detailed descriptions of these four models in Appendix C.1.1.

In addition, we include 10 additional models, including GBDTs (Chen & Guestrin, 2016; Ke et al., 2017; Klambauer et al., 2017; Prokhorenkova et al., 2018; Gorishniy et al., 2021; Somepalli et al., 2021; Wang et al., 2021; Chen et al., 2023; 2024; Gorishniy et al., 2024) for comparison purposes without incorporating our proposed method. TabPFN (Hollmann et al., 2025) is also included in

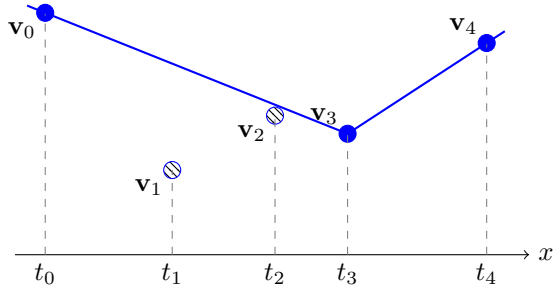


Figure 2: The effect of stochastic breakpoint regularization. When the middle breakpoints are deactivated (dashed circle), a new, linear piece is formed directly between its neighbors (solid circle).

Table 2: Average ranks across 46 datasets (lower is better). The numbers in parentheses indicate the rank improvement from applying GGPL.

Model	All Tasks (↓)	Regression (↓)	Classification (↓)	Num. Embedding
GBDT models				
LightGBM	9.48	8.86	10.44	-
XGBoost	9.00	8.93	9.11	-
CatBoost	8.04	7.64	8.67	-
Deep learning models (without numerical embedding)				
DCN2	16.04	15.93	16.22	-
SNN	15.20	15.61	14.56	-
Deep learning models (with numerical embedding)				
ExcelFormer	13.61	13.86	13.22	GLU
SAINT	12.46	13.07	11.50	MLP
FT-Transformer	11.70	12.25	10.83	Linear
Trompt	10.63	10.39	11.00	Linear
MLP	10.28	10.32	10.22	Periodic
T2G-Former	9.02	9.14	8.83	Linear
TabR	7.98	8.32	7.44	Periodic
ModernNCA	7.70	8.82	5.94	Periodic
TabM-mini	3.61	3.00	4.56	Piecewise-linear
Deep learning models (with GGPL embedding)				
MLP-GGPL	8.28 (-2.00)	7.64 (-2.68)	9.28 (-0.94)	GGPL (Ours)
T2G-Former-GGPL	7.80 (-1.22)	7.07 (-2.07)	8.94 (+0.11)	GGPL (Ours)
ModernNCA-GGPL	7.09 (-0.61)	8.07 (-0.75)	5.56 (-0.38)	GGPL (Ours)
TabM-mini-GGPL	**2.96** (-0.65)	**2.04** (-0.96)	**4.39** (-0.17)	GGPL (Ours)

the comparison only on small-scale datasets that meet its constraints. Detailed descriptions of these baseline models are in Appendix C.1.2.

4.3 Implementation Details

For TabM, we use the official implementation from Gorishniy et al. (2025), while our implementations of T2G-Former and ModernNCA are based on the code from Liu et al. (2024). To ensure a fair comparison, we follow the training protocol of Gorishniy et al. (2025).

With some exceptions, we apply a slightly modified version of the quantile transform from scikit-learn (Pedregosa et al., 2011) to numerical features. We use cross-entropy loss for classification and mean squared error loss for regression. Hyperparameters are tuned using Optuna (Akiba et al., 2019) over 100 trials (50 for large datasets). All baselines except TabPFN are also tuned using the hyperparameter search spaces from Gorishniy et al. (2025), while TabPFN is evaluated with default configuration. Further details are provided in Appendix C.2.

4.4 Results

For each dataset, we evaluate model performance using accuracy for classification and root mean squared error (RMSE) for regression, averaging the results over 15 random seeds. To aggregate performance across datasets, we compute the average rank of each model based on these scores. Detailed results for each dataset can be found in Appendix D.

Main Results

We present the main results in Table 2, which shows the average ranks of our GGPL-enhanced models against various baselines across all 46 datasets, as well as separate ranks for regression and classification tasks. The results consistently show that applying our GGPL embedding leads to

Table 3: Comparison with TabPFN on 23 small-scale datasets.

Model	All (↓)	Reg. (↓)	Cls. (↓)
LightGBM	10.91	10.40	11.88
XGBoost	10.70	10.80	10.50
MLP	10.96	11.60	9.75
T2G-Former	10.22	9.87	10.88
ModernNCA	9.39	10.73	6.88
MLP-GGPL	9.04	8.53	10.00
T2G-Former-GGPL	8.57	7.20	11.13
CatBoost	8.00	7.93	8.13
ModernNCA-GGPL	7.52	8.13	6.38
TabPFN	5.96	6.93	**4.13**
TabM-mini	4.57	3.47	6.63
TabM-mini-GGPL	**3.30**	**2.47**	4.88

Table 4: Ablation study on the components of GGPL. Performance is measured by the average rank on all 46 datasets using the MLP backbone.

Embedding Method	Average Rank
Base	9.76
Base+I	8.91 (-0.85)
Base+I+O	8.52 (-0.39)
Base+I+O+R (GGPL)	8.28 (-0.24)

performance improvements across all four backbone architectures with diverse design paradigms. For instance, GGPL provides a substantial boost to MLP on regression tasks, improving its average rank by 2.68. Overall, TabM-mini-GGPL emerges as the best-performing model, achieving the top average rank of 2.96 across all tasks. This demonstrates that our proposed embedding is not only effective but also versatile, enhancing the capabilities of diverse model architectures.

Comparison on Small-Scale Datasets

To motivate our focus on task-specific models, we compare GGPL-enhanced variants against the foundation model TabPFN (Hollmann et al., 2025). This comparison uses 23 small-scale datasets from the full benchmark of 46 datasets that satisfy TabPFN's constraints ($\leq$ 10 classes, $\leq$ 500 features, and $\leq$ 10000 samples). As shown in Table 3, our GGPL-enhanced models are competitive with both TabPFN and CatBoost, even on small-scale datasets where these models are presumed to excel. Notably, TabM-mini-GGPL surpasses TabPFN to achieve the top overall rank. These results highlight that task-specific models can attain state-of-the-art performance without the scalability constraints of foundation models.

5 Analysis

5.1 Ablation Study

To isolate the contribution of each of our proposed components—Initialization (I), Optimization (O), and Regularization (R)—we conduct an ablation study on the MLP backbone. Starting from a piecewise-linear embedding where breakpoints are initialized uniformly (Base), we incrementally add I, O, and R. The results in Table 4 demonstrate that each component consistently improves performance, validating our design choices.

5.2 Statistical Testing of GGPL against Other Embeddings

To evaluate the effectiveness of our proposed GGPL embedding, we conduct two comparisons: (i) against existing numerical embeddings on the MLP backbone and (ii) as a drop-in replacement within other architectures (e.g., T2G, MNCA, TabM), replacing their native numerical embeddings. The baseline embeddings on the MLP backbone include no embedding, periodic encoding, and piecewise-linear embeddings with both initialization methods (quantile-based and target-aware; Gorishniy et al., 2022); additionally, we compare against the tree-based T2V method (Li et al., 2024) on the 16 binary classification tasks.

We apply a stratified Wilcoxon signed-rank test to assess statistical significance across datasets with multiple random seeds: each dataset is treated as a stratum, seed-level paired differences are computed within each dataset, and the stratum-specific statistics are aggregated into a single p-value. As shown in Tables 5 and 6, GGPL demonstrates consistent and statistically significant improvements—over all numerical embedding baselines on MLP and over the native numerical embeddings on other architectures.

Table 5: GGPL vs other numerical embeddings on the MLP backbone using stratified Wilcoxon signed-rank test.

Baseline Method	Z-statistics	p-value
No Embedding	17.0	$< 10^{-10}$
T2V	10.6	$< 10^{-10}$
Periodic	3.38	7.15×10^{-4}
Piecewise-linear (quantile-based)	4.22	2.43×10^{-5}
Piecewise-linear (target-aware)	4.32	1.58×10^{-5}

Table 6: GGPL vs native numerical embeddings across models using stratified Wilcoxon signed-rank test.

Model	Z-statistics	p-value
MLP	3.38	7.15×10^{-4}
T2G	4.90	9.74×10^{-7}
MNCA	2.37	1.79×10^{-2}
TabM	4.47	7.66×10^{-6}

We further quantify the effect size of GGPL when used as a drop-in replacement for native numerical embeddings by computing Elo ratings (Elo, 1967), which is recently adopted in TabArena (Erickson et al., 2025). Following TabArena, we estimate 95% confidence intervals via 200-round bootstrap resampling (2.5–97.5% quantiles). Unlike TabArena, we calibrate the Elo scale by fixing the MLP backbone with its native numerical embedding to 1000 and report all other Elo scores relative to this anchor.

As shown in Figure 3, GGPL increases the Elo rating of every backbone. For MLP, the 95% confidence intervals of the native and GGPL variants do not overlap. For T2G-Former and TabM-mini, the intervals partially overlap, and the mean Elo of the GGPL variant lies near the upper end of the native variant's confidence interval. For ModernNCA, the intervals overlap more substantially, indicating a smaller yet positive shift. This pattern is consistent with our stratified Wilcoxon signed-rank tests, where ModernNCA also exhibited the largest (yet still significant) p-value of 1.79×10^{-2}. Taken together, the Elo and Wilcoxon analyses confirm that replacing native numerical embeddings with GGPL yields consistent and statistically meaningful improvements across architectures. Further details of the Elo rating and plots of all models are provided in Appendix E.1.

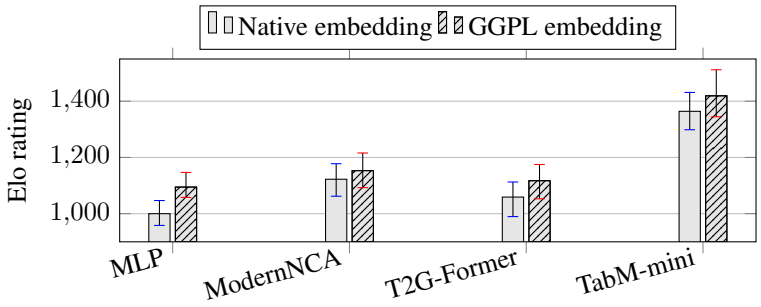


Figure 3: Elo ratings for each backbone with its native numerical embedding (solid bars) and with GGPL (hatched bars). MLP with native embedding is used as baseline (1000). Error bars indicate 95% confidence intervals.

5.3 Performance Analysis by Dataset Characteristics

To better understand where GGPL provides the most significant benefits, we analyze its performance improvement across different dataset characteristics in Table 7. A few general trends emerge from the data. First, we observe that the performance gains are consistently more pronounced in regression tasks than in classification. In contrast to classification tasks, where precise modeling near decision boundaries is important, regression tasks require global accuracy. This result suggests that our approach is particularly effective for regression tasks, as it helps model the entire feature–target relationship with high fidelity. Second, with the exception of MLP on feature-to-sample ratio, our method tends to yield greater improvements on datasets with small sample sizes and high feature-to-sample ratios. This indicates that GGPL provides a valuable inductive bias that is effective in preventing overfitting, where training data is limited or feature dimensionality is high.

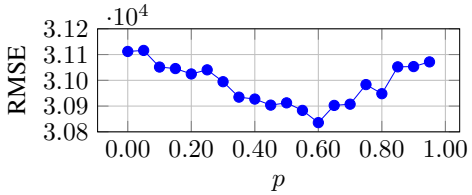
Table 7: Average rank improvement of GGPL across different dataset characteristics.

Characteristic	MLP	T2G-Former	ModernNCA	TabM-mini
Regression task	10.32 → 7.64 (**-2.68**)	9.14 → 7.07 (**-2.07**)	8.82 → 8.07 (**-0.75**)	3.00 → 2.04 (**-0.96**)
Classification task	10.22 → 9.28 (-0.94)	8.83 → 8.94 (+0.11)	5.94 → 5.56 (-0.38)	4.56 → 4.39 (-0.17)
Small sample size	10.58 → 8.21 (**-2.37**)	9.38 → 8.04 (**-1.34**)	8.25 → 6.75 (**-1.50**)	3.75 → 2.92 (**-0.83**)
Large sample size	9.95 → 8.36 (-1.59)	8.64 → 7.55 (-1.09)	7.09 → 7.45 (+0.36)	3.45 → 3.00 (-0.45)
High feature-to-sample ratio	10.19 → 8.52 (-1.67)	9.05 → 7.67 (**-1.38**)	9.10 → 7.62 (**-1.48**)	3.90 → 3.24 (**-0.66**)
Low feature-to-sample ratio	10.36 → 8.08 (**-2.28**)	9.00 → 7.92 (-1.08)	6.52 → 6.64 (+0.12)	3.36 → 2.72 (-0.64)

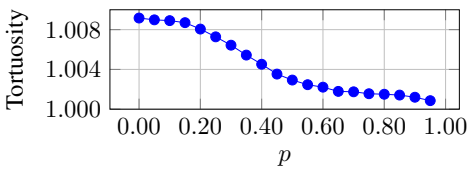
5.4 Analysis of Stochastic Breakpoint Regularization

We investigate the effect of stochastic breakpoint regularization by varying its deactivation ratio (p) using the MLP-GGPL model on the House 16H dataset (Gorishniy et al., 2024). While holding all other hyperparameters constant, we vary p from 0.0 to 0.95 in increments of 0.05, averaging results over 100 random seeds. We evaluate model performance and embedding complexity, where the latter is measured by tortuosity. Further details about tortuosity are provided in Appendix E.2.

Figure 4a shows that RMSE is minimized at $p = 0.60$, indicating that an appropriate level of regularization is essential for the best performance. Figure 4b shows that tortuosity decreases monotonically with p, confirming that the regularization smooths the embedding function as intended.



(a) RMSE: Model performance (RMSE) as a function of the deactivation ratio (p).



(b) Tortuosity: Embedding function tortuosity as a function of the deactivation ratio (p).

Figure 4: The effect of the deactivation ratio (p) in stochastic breakpoint regularization.

5.5 Effect Size Analysis

Finally, we aggregate effect sizes across datasets by normalizing each dataset so that the best-performing model scores 1 and the worst scores 0. We then use these normalized scores to compute, for each architecture and task, the mean and standard deviation of the gap between the baseline and its GGPL-enhanced variant. In addition, we plot histograms of the baseline and GGPL-normalized scores to visualize the distributional shifts; detailed statistics and figures are provided in Appendix E.3. On regression tasks, the GGPL distributions tend to shift toward 1 and the mean gains are consistently positive across architectures, whereas on classification tasks the two histograms almost overlap. These results indicate that GGPL improves (on regression) or at least matches (on classification) the native numerical embeddings of several state-of-the-art tabular architectures, supporting our goal of providing a practical default numerical embedding that can be plugged into diverse models with minimal overhead.

6 Conclusion

In this paper, we addressed the challenge of numerical feature embedding for deep tabular models. To this end, we proposed GGPL, a piecewise-linear embedding method built on three components: GBDT-guided initialization, stable optimization on a probability simplex, and stochastic breakpoint regularization. Our analysis confirms that all three components are essential for the method's effectiveness. With their synergy, GGPL significantly boosts various state-of-the-art models to achieve the top average rank in extensive experiments. Our method demonstrates particular strength in data-scarce settings, indicating it provides a valuable inductive bias to promote better generalization for deep tabular models.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Jintai Chen, Jiahuan Yan, Qiyuan Chen, Danny Z Chen, Jian Wu, and Jimeng Sun. Can a deep learning model be a sure bet for tabular prediction? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 288–296, 2024.

Kuan-Yu Chen, Ping-Han Chiang, Hsin-Rung Chou, Ting-Wei Chen, and Tien-Hao Chang. Trompt: Towards a better deep neural network for tabular data. In *International Conference on Machine Learning*, pp. 4392–4434. PMLR, 2023.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Arpad E Elo. The proposed uscf rating system, its development, theory, and applications. *Chess life*, 22(8):242–247, 1967.

Moonjung Eo, Kyungeun Lee, Hye-Seung Cho, Dongmin Kim, Ye Seul Sim, and Woohyung Lim. Representation space augmentation for effective self-supervised learning on tabular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 11625–11633, 2025.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025. URL https://arxiv.org/abs/2506.16791.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004, 2022.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors. In *ICLR*, 2024.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520, 2022.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1725–1731, 2017.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37:26577–26658, 2024.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34:23928–23941, 2021.

Liran Katzir, Gal Elidan, and Ran El-Yaniv. Net-{dnf}: Effective deep modeling of tabular data. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=73WTGs96kho.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Kyungeun Lee, Ye Seul Sim, Hye-Seung Cho, Moonjung Eo, Suhee Yoon, Sanghyu Yoon, and Woohyung Lim. Binning as a pretext task: Improving self-supervised learning in tabular domains. *arXiv preprint arXiv:2405.07414*, 2024.

Kyungeun Lee, Moonjung Eo, Hye-Seung Cho, Dongmin Kim, Ye Seul Sim, Seoyoon Kim, Min-Kook Suh, and Woohyung Lim. Multitab: A comprehensive benchmark suite for multi-dimensional evaluation in tabular domains. *arXiv preprint arXiv:2505.14312*, 2025.

Xuan Li, Yun Wang, and Bo Li. Tree-regularized tabular embeddings. *arXiv preprint arXiv:2403.00963*, 2024.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *arXiv preprint arXiv:2407.04057*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1eiu2VtwH.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. Tabred: Analyzing pitfalls and filling the gaps in tabular deep learning benchmarks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=L14sqcrUC3.

Renat Sergazinov, Jing Wu, and Shao-An Yin. Random at first, fast at last: Ntk-guided fourier pre-processing for tabular dl. *arXiv preprint arXiv:2506.02406*, 2025.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Shriyank Somvanshi, Subasish Das, Syed Aaqib Javed, Gian Antariksa, and Ahmed Hossain. A survey on deep tabular learning. *arXiv preprint arXiv:2410.12034*, 2024.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021.

Yuqian Wu, Hengyi Luo, and Raymond ST Lee. Deep feature embedding for tabular data. *arXiv preprint arXiv:2408.17162*, 2024.

Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z Chen, and Jian Wu. T2g-former: organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10720–10728, 2023.

Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JytL2MrlLT.

Disclosure of LLM Assistance

This paper benefited from assistance by a large language model (LLM) to improve its grammar.

A Algorithmic Details

A.1 GBDT-Guided Breakpoint Initialization

Given a trained GBDT, we collect node split thresholds and aggregate their gains to find a small set of informative breakpoints for each numerical feature, as described in Algorithm 1.

A.2 Sensitivity to the Choice of GBDT Initializer

In practice, we initialize breakpoints by training a default XGBoost model once per dataset. This choice keeps the pipeline simple with negligible computational overhead while already yielding competitive performance. To examine the sensitivity to the choice of GBDT initializer, we compare this default XGBoost against several alternatives on the MLP-GGPL backbone: a tuned XGBoost obtained via a 100-trial hyperparameter search using Optuna (Akiba et al., 2019), a default LightGBM model, and XGBoost variants with fewer trees ($n_{\text{estimators}}$). All other training settings are kept identical except for the GBDT initializer.

As summarized in Table 8, the tuned XGBoost tends to perform better than the default, but the difference is not statistically significant at the conventional $\alpha = 0.05$ level. Likewise, default LightGBM and a moderately shallower XGBoost yield performance that is statistically indistinguishable from the default XGBoost initializer, whereas only an extremely small XGBoost (with very few trees) shows a clear degradation. Overall, these results suggest that our default XGBoost initializer is a reasonable trade-off between simplicity and performance, while more aggressive tuning or alternative GBDTs can be used when additional performance gains are desired.

Algorithm 1 Breakpoint Selection from GBDT

Input: $S = [(i, t, g)]$: A list of node split information from the trained GBDT model, representing feature index, threshold, and gain for each node.
X: The training data, to extract min/max values.
K: Total number of internal breakpoints.
N_{num}: Set of numerical feature indices.
Output: T: A dictionary mapping numerical features to their sorted list of breakpoints.

$G \leftarrow$ defaultdict(float)
$T \leftarrow$ defaultdict(list)
for $(i, t, g) \in S$:
    $G[(i, t)] \mathrel{+}= g$ ▷ *Aggregate gain of thresholds*
$G \leftarrow \text{top\_k}(G, K)$ ▷ *Select top K splits*
for $(i, t) \in \text{keys}(G)$:
    $T[i].\text{append}(t)$ ▷ *Map thresholds to features*
for $i \in N_{num}$:
    $T[i].\text{extend}([\min(X[i]), \max(X[i])])$
    $T[i].\text{sort}()$ ▷ *Add boundaries and sort thresholds*
return T

Table 8: Comparison of the default XGBoost initializer with alternative GBDT initializers.

Comparison	Z-statistic	p-value
vs XGBoost (tuned, 100 trials)	-1.82	0.067
vs LightGBM (default)	0.88	0.375
vs XGBoost ($n_{\text{estimators}} = 10$)	0.56	0.571
vs XGBoost ($n_{\text{estimators}} = 1$)	4.14	3.34×10^{-5}

A.3 Effect of Stochastic Breakpoint Regularization

We quantify the effect of stochastic breakpoint regularization by comparing MLP-GGPL models trained with and without the regularizer, separately for regression and classification tasks. For each dataset, we use identical hyperparameters except for the regularization switch ($p > 0$ vs. $p = 0$), evaluate 15 random seeds, and apply a stratified Wilcoxon signed-rank test over seed-level paired differences.

Table 9: Stratified Wilcoxon signed-rank tests for stochastic breakpoint regularization (on vs. off) on MLP-GGPL across 46 datasets.

Task	Z-statistic	p-value
Regression	1.36	0.17
Classification	-0.17	0.87

For regression, the positive Z-statistic with a moderate p-value suggests a weak trend that the regularized variant tends to perform better than its unregularized counterpart. For classification, the test statistic is slightly negative with a large p-value ($p \approx 0.87$), indicating that the regularization does not help for classification. Consistent with these observations, we enable stochastic breakpoint regularization only for regression tasks and disable it for classification.

B Dataset Details

B.1 Preprocessing

To ensure a fair comparison and reproducibility, we follow the preprocessing used in Gorishniy et al. (2025), from which we adopt the benchmark datasets. Our preprocessing follows their methodology without any modifications. The key procedures are summarized below.

- Numerical features: By default, a slightly modified version of the quantile transform from scikit-learn (Pedregosa et al., 2011) is applied, which adds a small Gaussian noise (mean: 0, std: 1e-5) before calculating the distribution. For exceptions where quantile transform is detrimental, standard normalization or identical mapping is used.
- Categorical features: All categorical features are processed using one-hot encoding.
- Binary features: Features with only two distinct values are mapped to $\{0, 1\}$.

Please refer to the config files on the source code in the supplementary materials for dataset-specific details.

B.2 Dataset Characteristics

We provide a detailed overview of the 46 datasets used in our evaluation in Table 10. This benchmark, originally used by Gorishniy et al. (2025), is composed of datasets from three sources: 28 from Grinsztajn et al. (2022), 10 from Gorishniy et al. (2024), and 8 from Rubachev et al. (2025). The table summarizes key characteristics for each dataset, including its size, feature composition (numerical, binary, and categorical), task type, and its corresponding reference within the benchmark.

C Experimental Details

C.1 Baseline Model Details

C.1.1 Backbone Models for GGPL

We integrated our proposed GGPL embedding into four backbone models. For each model, the original component for processing numerical features was replaced by GGPL.

- **MLP:** A standard multi-layer perceptron, often used as a deep learning baseline due to its small and lightweight architecture. Gorishniy et al. (2022) compare various numerical embeddings on MLPs and find that piecewise-linear and periodic embeddings yield substantial performance improvements. We use a standard MLP as a primary backbone for evaluating GGPL and the base model for our in-depth analyses.
- **T2G-Former:** T2G-Former (Yan et al., 2023) is a Transformer-based architecture for tabular data that uses a T2G module to model feature interactions. We replace its original linear embedding layer for numerical features with GGPL.
- **ModernNCA:** ModernNCA (Ye et al., 2025) is a retrieval-augmented model that learns a distance metric for nearest-neighbor-based prediction. Its original numerical embedding, based on periodic functions, is replaced with GGPL.
- **TabM:** TabM (Gorishniy et al., 2025) is an MLP-based model with parameter-efficient ensembling. Among its variants, TabM-mini with a piecewise-linear embedding achieves the best performance. In our experiments, we employ the TabM-mini and replace the original quantile-based embedding with fixed breakpoints with GGPL.

C.1.2 Other Baseline Models for Comparison

To establish a comprehensive performance benchmark, we compare our GGPL-enhanced models against the following groups of baseline models.

- **GBDT models:** These models represent the traditional machine learning methods for tabular data.

  XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), CatBoost (Prokhorenkova et al., 2018)
- **Other Deep Learning Models:** These models represent diverse advancements in deep learning architectures for tabular data.

Table 10: Detailed characteristics of the 46 benchmark datasets.

Dataset	# Samples	# Feat.	# Num	# Bin	# Cat	Task	# Classes	Reference
Adult	48842	14	6	1	7	cls.	2	Gorishniy et al. (2024)
Black_Friday	166821	9	4	1	4	reg.	-	Gorishniy et al. (2024)
California_Housing	20640	8	8	0	0	reg.	-	Gorishniy et al. (2024)
Churn_Modelling	10000	11	7	3	1	cls.	2	Gorishniy et al. (2024)
Covertype	581012	15	10	4	1	cls.	7	Gorishniy et al. (2024)
Diamond	53940	9	6	0	3	reg.	-	Gorishniy et al. (2024)
Higgs_Small	98049	28	28	0	0	cls.	2	Gorishniy et al. (2024)
House_16H	22784	16	16	0	0	reg.	-	Gorishniy et al. (2024)
Microsoft	1200192	136	131	5	0	reg.	-	Gorishniy et al. (2024)
Otto_Group_Products	61878	93	93	0	0	cls.	9	Gorishniy et al. (2024)
Ailerons	13750	33	33	0	0	reg.	-	Grinsztajn et al. (2022)
analcatdata_supreme	4052	7	2	5	0	reg.	-	Grinsztajn et al. (2022)
bank-marketing	10578	7	7	0	0	cls.	2	Grinsztajn et al. (2022)
Brazilian_houses	10692	11	8	2	1	reg.	-	Grinsztajn et al. (2022)
cpu_act	8192	21	21	0	0	reg.	-	Grinsztajn et al. (2022)
credit	16714	10	10	0	0	cls.	2	Grinsztajn et al. (2022)
elevators	16599	16	16	0	0	reg.	-	Grinsztajn et al. (2022)
fifa	18063	5	5	0	0	reg.	-	Grinsztajn et al. (2022)
house_sales	21613	17	15	2	0	reg.	-	Grinsztajn et al. (2022)
isolet	7797	613	613	0	0	reg.	-	Grinsztajn et al. (2022)
jannis	57580	54	54	0	0	cls.	2	Grinsztajn et al. (2022)
kdd_ipums_la_97-small	5188	20	20	0	0	cls.	2	Grinsztajn et al. (2022)
KDDCup09_upselling	5032	49	34	1	14	cls.	2	Grinsztajn et al. (2022)
MagicTelescope	13376	10	10	0	0	cls.	2	Grinsztajn et al. (2022)
medical_charges	163065	3	3	0	0	reg.	-	Grinsztajn et al. (2022)
Mercedes_Benz	4209	359	0	356	3	reg.	-	Grinsztajn et al. (2022)
MiamiHousing2016	13932	13	13	0	0	reg.	-	Grinsztajn et al. (2022)
MiniBooNE	72998	50	50	0	0	cls.	2	Grinsztajn et al. (2022)
nyc-taxi-green	581835	16	9	3	4	reg.	-	Grinsztajn et al. (2022)
OnlineNewsPopularity	39644	59	45	14	0	reg.	-	Grinsztajn et al. (2022)
particulate-matter-ukair	394299	6	3	0	3	reg.	-	Grinsztajn et al. (2022)
phoneme	3172	5	5	0	0	cls.	2	Grinsztajn et al. (2022)
pol	15000	26	26	0	0	reg.	-	Grinsztajn et al. (2022)
road-safety	111762	32	29	0	3	cls.	2	Grinsztajn et al. (2022)
superconduct	21263	79	79	0	0	reg.	-	Grinsztajn et al. (2022)
wine	2554	11	11	0	0	cls.	2	Grinsztajn et al. (2022)
wine_quality	6497	11	11	0	0	reg.	-	Grinsztajn et al. (2022)
year	515345	90	90	0	0	reg.	-	Grinsztajn et al. (2022)
Cooking_Time	319986	192	186	3	3	reg.	-	Rubachev et al. (2025)
Delivery_ETA	350516	220	218	1	1	reg.	-	Rubachev et al. (2025)
Ecom_Offers	160057	110	104	6	0	cls.	2	Rubachev et al. (2025)
Homecredit_Default	381664	677	593	2	82	cls.	2	Rubachev et al. (2025)
Homesite_Insurance	260753	298	252	23	23	cls.	2	Rubachev et al. (2025)
Maps_Routing	279945	986	984	0	2	reg.	-	Rubachev et al. (2025)
Sberbank_Housing	28321	392	365	17	10	reg.	-	Rubachev et al. (2025)
Weather	189963	99	96	3	0	reg.	-	Rubachev et al. (2025)

SNN (Klambauer et al., 2017), FT-Transformer (Gorishniy et al., 2021), SAINT (Somepalli et al., 2021), DCN2 (Wang et al., 2021), Trompt (Chen et al., 2023), ExcelFormer (Chen et al., 2024), TabR (Gorishniy et al., 2024)

- **Foundation Model:** TabPFN is a pre-trained foundation model that can perform inference on unseen tasks without any parameter tuning, although its application is limited by dataset size constraints.

  TabPFN (Hollmann et al., 2025)

C.2 Implementation Details

C.2.1 Hardware Environment

Our experiments were conducted on servers equipped with Intel(R) Xeon(R) Gold 6240 CPUs @ 2.60GHz and NVIDIA RTX 3090 GPUs. While most experiments were run on a single GPU, training on large datasets required up to 8 GPUs to meet GPU memory demands, particularly for models with high memory consumption (T2G-Former Yan et al., 2023, ModernNCA Ye et al., 2025).

C.2.2 HYPERPARAMETER SEARCH SPACES

All models are trained using the AdamW optimizer (Loshchilov & Hutter, 2019) with an early stopping patience of 16 epochs, and hyperparameters were tuned using Optuna (Akiba et al., 2019) with the TPE sampler over 100 trials for most datasets, and 50 trials for large ones.

For our GGPL-enhanced backbone models (MLP, T2G-Former, ModernNCA, and TabM), the search spaces for all non-GGPL hyperparameters were kept identical to those in Liu et al. (2024) for T2G-Former and ModernNCA and those in Gorishniy et al. (2025) for MLP and TabM. We provide the detailed search spaces in Tables 11 to 14.

For all baseline models, including GBDTs and other deep learning methods, the hyperparameter search spaces were kept identical to those defined in Gorishniy et al. (2025). We refer to their original paper for the details.

D DETAILED EXPERIMENTAL RESULTS

Tables 15 and 16 provide the detailed performance metrics for all models across all 46 datasets, including the mean and standard deviation over 15 random seeds. For the baseline models, we report the performance scores directly from the original benchmark publication by Gorishniy et al. (2025), as our experimental setup is identical to theirs.

Table 11: Hyperparameter search spaces for MLP-GGPL.

Hyperparameter	**Search Space**
Learning rate	LogUniform: [3e-5, 0.001]
Weight decay	{0, LogUniform: [0.0001, 0.1]}
# layers	Int: [1, 5]
Width	Int: [64, 1024, 16]
Dropout	{0, Uniform: [0.0, 0.5]}
Embedding dim. (d)	Int: [8, 32, 4]
Average number of breakpoints (K)	Int: [2, 48]
Deactivation Prob. (p)	{0, Uniform: [0.0, 0.3]}

Table 12: Hyperparameter search spaces for T2G-Former-GGPL.

Hyperparameter	**Search Space**
Learning rate	LogUniform: [1e-5, 0.001]
Weight decay	LogUniform: [1e-6, 0.001]
# layers	Int: [1, 4]
Token width	Categorical: {8, 16, 32, 64, 128}
Residual dropout	{0, Uniform: [0.0, 0.2]}
Attention dropout	Uniform: [0.0, 0.5]
FFN dropout	Uniform: [0.0, 0.5]
FFN expansion rate	Uniform: [0.67, 2.67]
Frozen switch	Categorical: {true, false}
Activation	reglu
Num heads	8
Embedding dim. (d)	Identical to the token width
Average number of breakpoints (K)	Int: [2, 48]
Deactivation Prob. (p)	{0, Uniform: [0.0, 0.3]}

Table 13: Hyperparameter search spaces for ModernNCA-GGPL.

Hyperparameter	**Search Space**
Learning rate	LogUniform: [1e-5, 0.1]
Weight decay	{0, LogUniform: [1e-6, 0.001]}
# MLP layers	{0, Int: [0, 2]}
MLP width	Int: [64, 1024]
Projection Dim.	Int: [64, 1024]
Dropout	Uniform: [0.0, 0.5]
Sample rate	Uniform: [0.05, 0.6]
Temperature	1.0
Embedding dim. (d)	Int: [8, 32, 4]
Average number of breakpoints (K)	Int: [2, 48]
Deactivation Prob. (p)	{0, Uniform: [0.0, 0.3]}

Table 14: Hyperparameter search spaces for TabM-mini-GGPL.

Hyperparameter	**Search Space**
Learning rate	LogUniform: [0.0001, 0.003]
Weight decay	{0, LogUniform: [0.0001, 0.1]}
# layers	Int: [1, 4]
Width	Int: [64, 1024, 16]
Dropout	{0, Uniform: [0.0, 0.5]}
# ensembles	32
Embedding dim. (d)	Int: [8, 32, 4]
Average number of breakpoints (K)	Int: [2, 48]
Deactivation Prob. (p)	{0, Uniform: [0.0, 0.3]}

Table 15: Detailed classification results (Accuracy ↑). Scores for baseline models are from Gorishniy et al. (2025).

Dataset	LightGBM	XGBoost	CatBoost	DCN2	SNN	Excel	SAINT	FT-T	Trompt	MLP	T2G	TabR	MNCA	TabM	TabPFN	MLP-GGPL	T2G-GGPL	MNCA-GGPL	TabM-GGPL
Adult	0.8720 ± 0.0006	0.8713 ± 0.0007	0.8714 ± 0.0012	0.8582 ± 0.0011	0.8582 ± 0.0009	0.8613 ± 0.0024	0.8601 ± 0.0019	0.8588 ± 0.0015	0.8590 ± nan	0.8693 ± 0.0007	0.8601 ± 0.0011	0.8699 ± 0.0011	0.8717 ± 0.0008	0.8700 ± 0.0007	N/A	0.8657 ± 0.0013	0.8696 ± 0.0012	0.8715 ± 0.0011	0.8701 ± 0.0017
Churn_Modelling	0.8605 ± 0.0022	0.8600 ± 0.0008	0.8582 ± 0.0017	0.8567 ± 0.0020	0.8506 ± 0.0051	0.8618 ± 0.0023	0.8603 ± 0.0029	0.8593 ± 0.0028	0.8600 ± nan	0.8624 ± 0.0010	0.8613 ± 0.0015	0.8625 ± 0.0021	0.8606 ± 0.0032	0.8606 ± 0.0023	0.8620 ± nan	0.8610 ± 0.0026	0.8585 ± 0.0033	0.8625 ± 0.0030	0.8625 ± 0.0022
Covertype	0.9710 ± 0.0002	0.9709 ± 0.0003	0.9670 ± 0.0003	0.9622 ± 0.0019	0.9636 ± 0.0010	0.9606 ± 0.0018	0.9669 ± 0.0010	0.9698 ± 0.0008	0.9286 ± nan	0.9690 ± 0.0008	0.9668 ± 0.0008	0.9752 ± 0.0003	0.9747 ± 0.0002	0.9755 ± 0.0003	N/A	0.9720 ± 0.0009	0.9742 ± 0.0005	0.9758 ± 0.0004	0.9759 ± 0.0004
Higgs_Small	0.7246 ± 0.0015	0.7256 ± 0.0009	0.7260 ± 0.0011	0.7164 ± 0.0030	0.7142 ± 0.0024	0.7262 ± 0.0017	0.7236 ± 0.0019	0.7281 ± 0.0016	0.7262 ± nan	0.7260 ± 0.0017	0.7352 ± 0.0037	0.7294 ± 0.0014	0.7300 ± 0.0020	0.7361 ± 0.0011	N/A	0.7250 ± 0.0011	0.7279 ± 0.0016	0.7275 ± 0.0014	0.7288 ± 0.0009
Otto_Group_Products	0.8297 ± 0.0011	0.8302 ± 0.0009	0.8250 ± 0.0013	0.8064 ± 0.0021	0.8087 ± 0.0020	0.8102 ± 0.0022	0.8119 ± 0.0018	0.8133 ± 0.0033	0.8093 ± nan	0.8190 ± 0.0021	0.8161 ± 0.0019	0.8246 ± 0.0018	0.8265 ± 0.0015	0.8342 ± 0.0012	N/A	0.8215 ± 0.0025	0.8142 ± 0.0023	0.8291 ± 0.0014	0.8330 ± 0.0012
Ecom_Offers	0.5763 ± 0.0072	0.5758 ± 0.0006	0.5596 ± 0.0068	0.5996 ± 0.0043	0.5912 ± 0.0056	0.5759 ± 0.0066	0.5812 ± 0.0098	0.5775 ± 0.0063	0.5803 ± nan	0.5800 ± 0.0029	0.5791 ± 0.0056	0.5762 ± 0.0052	0.5758 ± 0.0050	0.5910 ± 0.0012	N/A	0.5943 ± 0.0024	0.5807 ± 0.0094	0.5865 ± 0.0041	0.5885 ± 0.0020
Homecredit_Default	0.8670 ± 0.0005	0.8664 ± 0.0004	0.8627 ± nan	0.8471 ± 0.0019	0.8541 ± 0.0016	0.8513 ± 0.0024	0.8377 ± nan	0.8571 ± 0.0023	0.8355 ± nan	0.8598 ± 0.0009	0.8597 ± 0.0007	0.8547 ± 0.0021	0.8544 ± 0.0033	0.8635 ± 0.0008	N/A	0.8557 ± 0.0018	0.8586 ± 0.0064	0.8523 ± 0.0022	0.8523 ± 0.0024
Homesite_Insurance	0.9601 ± 0.0002	0.9603 ± 0.0002	0.9606 ± 0.0003	0.9398 ± 0.0053	0.9473 ± 0.0013	0.9622 ± 0.0004	0.9613 ± nan	0.9622 ± 0.0006	0.9588 ± nan	0.9609 ± 0.0009	0.9624 ± 0.0006	0.9556 ± 0.0021	0.9620 ± 0.0006	0.9631 ± 0.0003	N/A	0.9613 ± 0.0006	0.9644 ± 0.0010	0.9609 ± 0.0009	0.9638 ± 0.0005
bank-marketing	0.8013 ± 0.0081	0.8006 ± 0.0078	0.8026 ± 0.0068	0.7859 ± 0.0068	0.7836 ± 0.0074	0.7957 ± 0.0090	0.7953 ± 0.0058	0.7918 ± 0.0076	0.7975 ± 0.0080	0.7947 ± 0.0101	0.7918 ± 0.0058	0.8023 ± 0.0088	0.7977 ± 0.0081	0.7989 ± 0.0086	0.8002 ± 0.0088	0.8001 ± 0.0078	0.7994 ± 0.0085	0.7996 ± 0.0095	0.8028 ± 0.0089
credit	0.7698 ± 0.0027	0.7686 ± 0.0028	0.7734 ± 0.0035	0.7703 ± 0.0034	0.7712 ± 0.0045	0.7724 ± 0.0038	0.7739 ± 0.0052	0.7745 ± 0.0041	0.7740 ± 0.0006	0.7749 ± 0.0055	0.7744 ± 0.0046	0.7723 ± 0.0037	0.7734 ± 0.0045	0.7761 ± 0.0033	0.7739 ± 0.0065	0.7743 ± 0.0038	0.7724 ± 0.0033	0.7727 ± 0.0044	0.7726 ± 0.0026
jannis	0.7967 ± 0.0019	0.7956 ± 0.0017	0.7985 ± 0.0018	0.7712 ± 0.0029	0.7818 ± 0.0025	0.7954 ± 0.0015	0.7971 ± 0.0028	0.7940 ± 0.0028	0.8027 ± nan	0.7923 ± 0.0018	0.7998 ± 0.0024	0.8051 ± 0.0023	0.8068 ± 0.0021	0.8078 ± 0.0008	N/A	0.7899 ± 0.0013	0.8001 ± 0.0016	0.8076 ± 0.0016	0.8079 ± 0.0014
kdd_ipums_la_97-small	0.7930 ± 0.0108	0.7932 ± 0.0119	0.7992 ± 0.0117	0.7850 ± 0.0161	0.7884 ± 0.0122	0.7903 ± 0.0074	0.7942 ± 0.0112	0.7957 ± 0.0127	0.7994 ± 0.0055	0.7962 ± 0.0093	0.8037 ± 0.0100	0.7908 ± 0.0123	0.7960 ± 0.0131	0.8024 ± 0.0075	0.8053 ± 0.0159	0.7975 ± 0.0084	0.8019 ± 0.0105	0.7950 ± 0.0106	0.8061 ± 0.0100
KDDCup09_upselling	0.8825 ± 0.0089	0.8792 ± 0.0075	0.8793 ± 0.0088	0.8770 ± 0.0072	0.8722 ± 0.0093	0.8803 ± 0.0054	0.8837 ± 0.0055	0.8795 ± 0.0077	0.8847 ± 0.0070	0.8765 ± 0.0108	0.8833 ± 0.0054	0.8831 ± 0.0050	0.8837 ± 0.0062	0.8779 ± 0.0094	0.8807 ± 0.0051	0.8805 ± 0.0087	0.8827 ± 0.0057	0.8827 ± 0.0058	0.8834 ± 0.0076
MagicTelescope	0.8550 ± 0.0094	0.8547 ± 0.0085	0.8586 ± 0.0070	0.8432 ± 0.0074	0.8536 ± 0.0052	0.8480 ± 0.0090	0.8595 ± 0.0060	0.8588 ± 0.0046	0.8605 ± 0.0102	0.8591 ± 0.0061	0.8553 ± 0.0055	0.8641 ± 0.0052	0.8622 ± 0.0085	0.8644 ± 0.0088	0.8699 ± 0.0097	0.8572 ± 0.0060	0.8587 ± 0.0062	0.8618 ± 0.0039	0.8624 ± 0.0065
MiniBooNE	0.9436 ± 0.0006	0.9422 ± 0.0009	0.9453 ± 0.0008	0.9433 ± 0.0011	0.9476 ± 0.0013	0.9430 ± 0.0015	0.9471 ± 0.0009	0.9467 ± 0.0014	0.9473 ± nan	0.9466 ± 0.0009	0.9475 ± 0.0014	0.9475 ± 0.0007	0.9493 ± 0.0012	0.9490 ± 0.0004	N/A	0.9477 ± 0.0010	0.9470 ± 0.0010	0.9498 ± 0.0011	0.9494 ± 0.0007
phoneme	0.8682 ± 0.0174	0.8702 ± 0.0129	0.8827 ± 0.0117	0.8342 ± 0.0151	0.8596 ± 0.0124	0.8551 ± 0.0092	0.8657 ± 0.0130	0.8667 ± 0.0127	0.8465 ± 0.0205	0.8742 ± 0.0120	0.8672 ± 0.0166	0.8772 ± 0.0087	0.8828 ± 0.0082	0.8780 ± 0.0119	0.8846 ± 0.0068	0.8576 ± 0.0133	0.8669 ± 0.0102	0.8815 ± 0.0098	0.8705 ± 0.0132
road-safety	0.8101 ± 0.0017	0.7982 ± 0.0012	0.8012 ± 0.0009	0.7781 ± 0.0014	0.7847 ± 0.0010	0.7864 ± 0.0053	0.7584 ± 0.0584	0.7907 ± 0.0012	0.7804 ± nan	0.7867 ± 0.0018	0.7912 ± 0.0026	0.8374 ± 0.0013	0.8232 ± 0.0017	0.7999 ± 0.0023	N/A	0.7897 ± 0.0009	0.7939 ± 0.0009	0.8118 ± 0.0023	0.8016 ± 0.0031
wine	0.7949 ± 0.0178	0.7890 ± 0.0160	0.7994 ± 0.0131	0.7492 ± 0.0147	0.7818 ± 0.0143	0.7631 ± 0.0171	0.7684 ± 0.0144	0.7755 ± 0.0133	0.7818 ± 0.0081	0.7803 ± 0.0157	0.7733 ± 0.0118	0.7804 ± 0.0148	0.7867 ± 0.0113	0.7839 ± 0.0169	0.7970 ± 0.0060	0.7798 ± 0.0193	0.7708 ± 0.0120	0.7963 ± 0.0151	0.7861 ± 0.0200

Table 16: Detailed regression results (RMSE ↓). Scores for baseline models are from Gorishniy et al. (2025).

Dataset	LightGBM	XGBoost	CatBoost	DCN2	SNN	Excel	SAINT	FT-T	Trompt	MLP	T2G	TabR	MNCA	TabM	TabPFN	MLP-GGPL	T2G-GGPL	MNCA-GGPL	TabM-GGPL
Black_Friday	0.6806 ± 0.0001	0.6799 ± 0.0003	0.6822 ± 0.0003	0.6968 ± 0.0013	0.6996 ± 0.0013	0.6947 ± 0.0016	0.6934 ± 0.0009	0.6987 ± 0.0192	0.6983 ± nan	0.6849 ± 0.0006	0.6887 ± 0.0046	0.6761 ± 0.0009	0.6885 ± 0.0007	0.6781 ± 0.0004	N/A	0.6849 ± 0.0009	0.6857 ± 0.0013	0.6875 ± 0.0007	0.6772 ± 0.0004
California_Housing	0.4327 ± 0.0016	0.4352 ± 0.0019	0.4294 ± 0.0012	0.4971 ± 0.0122	0.5033 ± 0.0075	0.4544 ± 0.0048	0.4680 ± 0.0048	0.4635 ± 0.0048	0.4579 ± nan	0.4652 ± 0.0045	0.4640 ± 0.0100	0.3998 ± 0.0033	0.4142 ± 0.0031	0.4275 ± 0.0024	N/A	0.4491 ± 0.0030	0.4438 ± 0.0038	0.4216 ± 0.0024	0.4157 ± 0.0021
Diamond	0.1368 ± 0.0004	0.1359 ± 0.0002	0.1335 ± 0.0006	0.1420 ± 0.0032	0.1473 ± 0.0057	0.1766 ± 0.0023	0.1369 ± 0.0019	0.1376 ± 0.0013	0.1391 ± nan	0.1342 ± 0.0008	0.1372 ± 0.0011	0.1333 ± 0.0013	0.1327 ± 0.0012	0.1315 ± 0.0006	N/A	0.1319 ± 0.0007	0.1359 ± 0.0011	0.1328 ± 0.0014	0.1308 ± 0.0008
House_16H (×1e-4)	3.1773 ± 0.0102	3.1774 ± 0.0087	3.1172 ± 0.0125	3.3327 ± 0.0878	3.2176 ± 0.0376	3.2460 ± 0.0685	3.2424 ± 0.0595	3.1823 ± 0.0460	3.0638 ± nan	3.0633 ± 0.0248	3.1613 ± 0.0320	3.1048 ± 0.0410	3.0704 ± 0.0388	2.9976 ± 0.0196	N/A	3.0974 ± 0.0365	3.1069 ± 0.0230	3.1072 ± 0.0360	2.9760 ± 0.0209
Microsoft	0.7413 ± 0.0001	0.7417 ± 0.0001	0.7412 ± 0.0001	0.7499 ± 0.0003	0.7488 ± 0.0004	0.7479 ± 0.0007	0.7625 ± 0.0066	0.7460 ± 0.0007	0.7476 ± nan	0.7446 ± 0.0002	0.7460 ± 0.0006	0.7501 ± 0.0005	0.7460 ± 0.0008	0.7423 ± 0.0002	N/A	0.7459 ± 0.0004	0.7448 ± 0.0006	0.7466 ± 0.0005	0.7414 ± 0.0001
Cooking_Time	0.4823 ± 0.0001	0.4826 ± 0.0001	0.4823 ± 0.0001	0.4834 ± 0.0003	0.4835 ± 0.0006	0.4821 ± 0.0005	0.4840 ± nan	0.4820 ± 0.0008	0.4809 ± nan	0.4811 ± 0.0004	0.4809 ± 0.0008	0.4818 ± 0.0006	0.4818 ± 0.0005	0.4804 ± 0.0001	N/A	0.4811 ± 0.0005	0.4813 ± 0.0008	0.4816 ± 0.0006	0.4802 ± 0.0003
Delivery_ETA	0.5468 ± 0.0002	0.5468 ± 0.0001	0.5465 ± 0.0001	0.5516 ± 0.0014	0.5495 ± 0.0008	0.5552 ± 0.0030	0.5528 ± nan	0.5542 ± 0.0026	0.5519 ± nan	0.5521 ± 0.0014	0.5527 ± 0.0016	0.5520 ± 0.0015	0.5507 ± 0.0013	0.5510 ± 0.0019	N/A	0.5520 ± 0.0014	0.5513 ± 0.0016	0.5545 ± 0.0012	0.5491 ± 0.0010
Maps_Routing	0.1616 ± 0.0001	0.1618 ± 0.0000	0.1619 ± 0.0001	0.1656 ± 0.0004	0.1634 ± 0.0002	0.1628 ± 0.0001	0.1634 ± nan	0.1625 ± 0.0003	0.1624 ± nan	0.1618 ± 0.0002	0.1616 ± 0.0001	0.1622 ± 0.0002	0.1627 ± 0.0002	0.1610 ± 0.0001	N/A	0.1619 ± 0.0002	0.1616 ± 0.0002	0.1639 ± 0.0003	0.1607 ± 0.0001
Sberbank_Housing	0.2419 ± 0.0012	0.2468 ± 0.0009	0.2482 ± 0.0034	0.2616 ± 0.0049	0.2671 ± 0.0140	0.2533 ± 0.0046	0.2467 ± 0.0019	0.2440 ± 0.0038	0.2509 ± nan	0.2528 ± 0.0055	0.2416 ± 0.0025	0.2542 ± 0.0101	0.2448 ± 0.0039	0.2334 ± 0.0018	N/A	0.2360 ± 0.0021	0.2391 ± 0.0051	0.2456 ± 0.0042	0.2325 ± 0.0013
Weather	1.4671 ± 0.0006	1.4625 ± 0.0008	1.4688 ± 0.0019	1.5606 ± 0.0057	1.5280 ± 0.0085	1.5131 ± 0.0022	1.5097 ± 0.0045	1.5104 ± 0.0097	1.5187 ± nan	1.5170 ± 0.0040	1.4849 ± 0.0087	1.4458 ± 0.0018	1.5008 ± 0.0034	1.4651 ± 0.0020	N/A	1.5056 ± 0.0040	1.4850 ± 0.0064	1.5077 ± 0.0045	1.4505 ± 0.0025
Ailerons (×1e3)	0.1571 ± 0.0041	0.1581 ± 0.0038	0.1533 ± 0.0034	0.1663 ± 0.0028	0.1628 ± 0.0022	0.1566 ± 0.0048	0.1588 ± 0.0033	0.1578 ± 0.0043	0.1564 ± 0.0038	0.1595 ± 0.0040	0.1560 ± 0.0027	0.1615 ± 0.0035	0.1612 ± 0.0043	0.1515 ± 0.0032	0.0002 ± 0.0000	0.1578 ± 0.0019	0.1548 ± 0.0047	0.1561 ± 0.0040	0.1503 ± 0.0030
analcatdata_supreme	0.0801 ± 0.0126	0.0778 ± 0.0115	0.0780 ± 0.0067	0.0811 ± 0.0137	0.0826 ± 0.0096	0.0796 ± 0.0101	0.0773 ± 0.0078	0.0787 ± 0.0086	0.0782 ± 0.0095	0.0798 ± 0.0088	0.0775 ± 0.0081	0.0807 ± 0.0088	0.0825 ± 0.0090	0.0764 ± 0.0071	0.0740 ± 0.0085	0.0757 ± 0.0061	0.0747 ± 0.0077	0.0782 ± 0.0082	0.0760 ± 0.0060
Brazilian_houses	0.0541 ± 0.0270	0.0603 ± 0.0249	0.0468 ± 0.0312	0.0477 ± 0.0172	0.0630 ± 0.0162	0.0450 ± 0.0156	0.0479 ± 0.0205	0.0438 ± 0.0181	0.0404 ± 0.0266	0.0426 ± 0.0180	0.0468 ± 0.0165	0.0451 ± 0.0163	0.0553 ± 0.0192	0.0416 ± 0.0215	0.0302 ± 0.0305	0.0408 ± 0.0191	0.0423 ± 0.0185	0.0527 ± 0.0156	0.0277 ± 0.0182
cpu_act	2.5237 ± 0.3530	2.2223 ± 0.0894	2.1239 ± 0.0489	2.7868 ± 0.1999	2.5811 ± 0.1480	2.3094 ± 0.2401	2.2781 ± 0.0630	2.2394 ± 0.0508	2.2133 ± 0.0221	2.2730 ± 0.0457	2.2111 ± 0.0413	2.1278 ± 0.0783	2.2105 ± 0.0483	2.1391 ± 0.0542	2.6891 ± 0.0591	2.3031 ± 0.0853	2.1663 ± 0.0743	2.1901 ± 0.0596	2.1117 ± 0.0595
elevators	0.0020 ± 0.0000	0.0020 ± 0.0000	0.0020 ± 0.0000	0.0019 ± 0.0000	0.0020 ± 0.0001	0.0019 ± 0.0000	0.0018 ± 0.0000	0.0019 ± 0.0000	0.0018 ± 0.0000	0.0019 ± 0.0000	0.0019 ± 0.0000	0.0019 ± 0.0001	0.0018 ± 0.0000	0.0018 ± 0.0000	0.0019 ± 0.0000	0.0018 ± 0.0000	0.0018 ± 0.0000	0.0018 ± 0.0000	0.0018 ± 0.0000
fifa	0.7800 ± 0.0108	0.7806 ± 0.0120	0.7835 ± 0.0116	0.8046 ± 0.0135	0.8074 ± 0.0140	0.7909 ± 0.0111	0.7901 ± 0.0118	0.7928 ± 0.0132	0.7880 ± 0.0180	0.7940 ± 0.0118	0.7928 ± 0.0139	0.7914 ± 0.0136	0.7909 ± 0.0107	0.7771 ± 0.0107	0.7820 ± 0.0137	0.7827 ± 0.0096	0.7809 ± 0.0099	0.7830 ± 0.0120	0.7775 ± 0.0111
house_sales	0.1694 ± 0.0003	0.1692 ± 0.0004	0.1669 ± 0.0001	0.1862 ± 0.0032	0.1800 ± 0.0008	0.1713 ± 0.0010	0.1713 ± 0.0015	0.1690 ± 0.0010	0.1667 ± nan	0.1699 ± 0.0008	0.1689 ± 0.0010	0.1636 ± 0.0009	0.1694 ± 0.0007	0.1652 ± 0.0003	0.1599 ± nan	0.1684 ± 0.0005	0.1666 ± 0.0008	0.1688 ± 0.0011	0.1636 ± 0.0003
isolet	2.7567 ± 0.0470	2.7005 ± 0.0296	2.8847 ± 0.0227	2.2449 ± 0.1579	2.4269 ± 0.2382	2.8691 ± 0.0882	2.7696 ± 0.0200	2.4879 ± 0.2524	2.6219 ± 0.0315	2.2719 ± 0.1006	2.2867 ± 0.2489	1.9919 ± 0.1813	1.8912 ± 0.1851	1.7799 ± 0.0859	N/A	2.1511 ± 0.1778	2.2857 ± 0.1721	1.8189 ± 0.1294	1.8515 ± 0.0766
medical_charges	0.0825 ± 0.0001	0.0820 ± 0.0000	0.0816 ± 0.0000	0.0818 ± 0.0003	0.0827 ± 0.0006	0.0817 ± 0.0004	0.0814 ± 0.0002	0.0814 ± 0.0002	0.0812 ± nan	0.0812 ± 0.0002	0.0813 ± 0.0002	0.0811 ± 0.0001	0.0809 ± 0.0000	0.0811 ± 0.0001	0.0813 ± nan	0.0813 ± 0.0000	0.0810 ± 0.0001	0.0811 ± 0.0001	0.0809 ± 0.0002
Mercedes_Benz_Greener_Manufacturing	8.2177 ± 0.8175	8.2078 ± 0.8231	8.1629 ± 0.8193	8.3540 ± 0.8314	8.2718 ± 0.8152	8.2244 ± 0.8514	8.3556 ± 0.9566	8.2252 ± 0.8617	8.3409 ± 0.9840	8.3045 ± 0.8708	8.2120 ± 0.8485	8.3187 ± 0.8186	8.2557 ± 0.8602	8.2075 ± 0.9185	8.1282 ± 0.9385	8.2939 ± 0.8668	8.2223 ± 0.8762	8.2644 ± 0.8182	8.2056 ± 0.9097
MiamiHousing2016	0.1440 ± 0.0029	0.1461 ± 0.0025	0.1417 ± 0.0021	0.1683 ± 0.0099	0.1618 ± 0.0029	0.1519 ± 0.0038	0.1507 ± 0.0022	0.1514 ± 0.0029	0.1478 ± 0.0028	0.1514 ± 0.0025	0.1523 ± 0.0023	0.1392 ± 0.0023	0.1475 ± 0.0031	0.1408 ± 0.0019	0.1360 ± 0.0027	0.1469 ± 0.0023	0.1471 ± 0.0030	0.1441 ± 0.0034	0.1397 ± 0.0021
nyc-taxi-green-dec-2016	0.3792 ± 0.0002	0.3688 ± 0.0002	0.3647 ± 0.0005	0.3919 ± 0.0009	0.3933 ± 0.0013	0.3969 ± 0.0036	0.3905 ± 0.0013	0.3937 ± 0.0064	0.3979 ± nan	0.3812 ± 0.0018	0.3908 ± 0.0045	0.3725 ± 0.0091	0.3536 ± 0.0052	0.3485 ± 0.0038	N/A	0.3556 ± 0.0013	0.3492 ± 0.0079	0.3370 ± 0.0039	0.3342 ± 0.0016
OnlineNewsPopularity	0.8545 ± 0.0002	0.8546 ± 0.0002	0.8532 ± 0.0003	0.8714 ± 0.0013	0.8692 ± 0.0015	0.8605 ± 0.0024	0.8600 ± 0.0007	0.8629 ± 0.0019	0.8623 ± nan	0.8604 ± 0.0009	0.8632 ± 0.0009	0.8624 ± 0.0011	0.8647 ± 0.0010	0.8563 ± 0.0004	0.8533 ± nan	0.8586 ± 0.0003	0.8591 ± 0.0010	0.8602 ± 0.0012	0.8540 ± 0.0003
particulate-matter-ukair-2017	0.3641 ± 0.0001	0.3637 ± 0.0001	0.3647 ± 0.0004	0.3759 ± 0.0012	0.3790 ± 0.0007	0.3699 ± 0.0014	0.3704 ± 0.0014	0.3735 ± 0.0012	0.3700 ± nan	0.3665 ± 0.0008	0.3676 ± 0.0024	0.3596 ± 0.0004	0.3646 ± 0.0001	0.3593 ± 0.0004	N/A	0.3643 ± 0.0008	0.3667 ± 0.0009	0.3642 ± 0.0004	0.3598 ± 0.0004
pol	4.2963 ± 0.0644	4.2320 ± 0.3369	3.6320 ± 0.1006	6.5374 ± 0.9479	6.1816 ± 0.7366	3.0682 ± 0.2389	2.7203 ± 0.1858	2.6974 ± 0.1666	3.2337 ± 0.0605	2.8239 ± 0.2173	2.9539 ± 0.1994	2.5770 ± 0.1689	2.9083 ± 0.1364	2.2808 ± 0.0343	4.4870 ± 0.4072	2.4938 ± 0.0860	2.4045 ± 0.0679	2.7848 ± 0.1148	2.3954 ± 0.0548
superconduct	10.1610 ± 0.0201	10.1634 ± 0.0118	10.2422 ± 0.0222	10.8108 ± 0.0957	10.8562 ± 0.1300	11.0879 ± 0.1571	10.7807 ± 0.1074	10.8256 ± 0.1692	10.4442 ± nan	10.5058 ± 0.0758	10.8310 ± 0.1406	10.3835 ± 0.0562	10.5651 ± 0.0616	10.1326 ± 0.0186	10.0041 ± nan	10.8142 ± 0.0732	10.8755 ± 0.0826	10.4330 ± 0.0536	10.2099 ± 0.0279
wine_quality	0.6039 ± 0.0134	0.6135 ± 0.0138	0.6088 ± 0.0132	0.7010 ± 0.0171	0.6604 ± 0.0174	0.6881 ± 0.0182	0.6797 ± 0.0161	0.6787 ± 0.0149	0.6605 ± 0.0153	0.6569 ± 0.0167	0.6783 ± 0.0170	0.6412 ± 0.0105	0.6099 ± 0.0144	0.6294 ± 0.0120	0.6935 ± 0.0264	0.6678 ± 0.0142	0.6808 ± 0.0175	0.6102 ± 0.0159	0.6288 ± 0.0144
year	9.0307 ± 0.0028	9.0200 ± 0.0025	9.0370 ± 0.0073	9.2761 ± 0.0401	9.0054 ± 0.0256	9.0395 ± 0.0266	9.0248 ± 0.0225	9.0005 ± 0.0215	8.9707 ± nan	8.9355 ± 0.0103	8.9775 ± 0.0138	8.9721 ± 0.0105	8.8973 ± 0.0082	8.8737 ± 0.0119	N/A	8.9244 ± 0.0109	8.9763 ± 0.0128	8.9437 ± 0.0149	8.8701 ± 0.0064

E Detailed Analysis

E.1 Elo Evaluation Details

We use an Elo-based evaluation Elo (1967) to summarize the relative performance of multiple architectures across heterogeneous datasets and tasks that differ in both evaluation metrics and difficulty. We include all 18 models considered in our main experiments (Table 2). These consist of each backbone (MLP, T2G-Former, ModernNCA, TabM-mini) with its native numerical embedding and with the proposed GGPL embedding, together with additional baselines such as GBDTs. The Elo evaluation is computed jointly over all 46 datasets (classification and regression), using the same metrics as in the main experiments (accuracy for classification and RMSE for regression).

Following TabArena (Erickson et al., 2025), we use a stable Bradley-Terry implementation to compute Elo scores and 200 bootstrapping rounds to approximate 2.5%-97.5% confidence interval. We also adopt the 400-point Elo gap. The expected win rate of i-th model with Elo ratings of R_i against R_j is

$$E_i = \frac{1}{1 + 10^{(R_j - R_i)/400}},$$

Finally, unlike TabArena, we calibrate 1000 Elo to the performance of native MLP model.

Figure 5 summarizes the resulting Elo ratings and confidence intervals for all 18 models. GGPL-enhanced variants consistently achieve higher Elo scores than their native counterparts, confirming the improvements reported in the main text.

E.2 Definition of Embedding Tortuosity

We quantify the complexity of a learned numerical embedding (ϕ) by measuring its tortuosity. For a given numerical feature i, let $t_1^{(i)} < t_2^{(i)} < \cdots < t_{K_i}^{(i)}$ denote the breakpoints that are initialized from GBDT (e.g., XGBoost) splits and subsequently optimized by backpropagation, and let $\mathbf{v}_1^{(i)}, \ldots, \mathbf{v}_{K_i}^{(i)} \in \mathbb{R}^d$ be the corresponding embedding vectors. As formulated in Eq. 4, we define the tortuosity of ϕ_i as the ratio of the piecewise-linear curve length to the Euclidean distance between its start and end points. A higher value indicates a more complicated function, while a value closer to 1 signifies a smoother path.

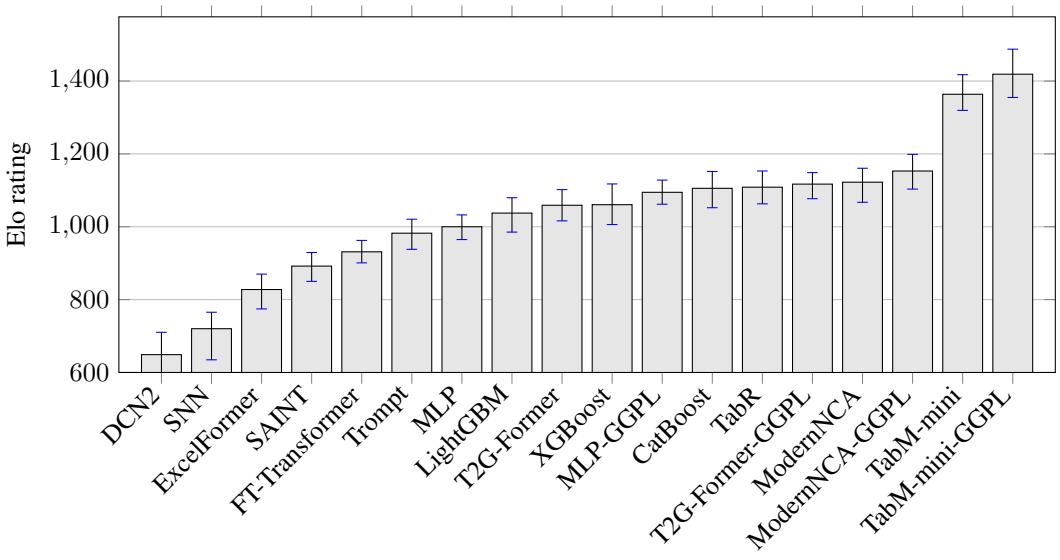
$$\text{Tortuosity}(\phi_i) = \frac{\sum_{k=0}^{K_i} \left\| [\mathbf{v}_{k+1}^{(i)}; t_{k+1}^{(i)}] - [\mathbf{v}_k^{(i)}; t_k^{(i)}] \right\|_2}{\left\| [\mathbf{v}_{K_i+1}^{(i)}; t_{K_i+1}^{(i)}] - [\mathbf{v}_0^{(i)}; t_0^{(i)}] \right\|_2} \tag{4}$$

E.3 Effect Size Details

We aggregate effect sizes across datasets by normalizing each dataset so that the best-performing model attains a score of 1 and the worst-performing model a score of 0. For each backbone and task, we then define the normalized-score gain Δ as the difference between the GGPL-enhanced variant and its baseline (native) numerical embedding, and summarize this gain via its mean and standard deviation over all datasets and seeds.

Figure 6 shows histograms of these normalized scores for regression (left) and classification (right) across all datasets and seeds for the four backbones (TabM-mini, ModernNCA, T2G-Former, MLP). For each architecture, we compare the native numerical embedding (Baseline) and GGPL, along with the mean $\mu(\Delta)$ and standard deviation $\sigma(\Delta)$ of Δ. On regression tasks, all backbones exhibit positive mean gains, and the GGPL histograms are shifted toward 1 relative to their baselines, indicating consistent improvements in performance. On classification tasks, the two histograms overlap. Table 2 and Figure 6 confirm that GGPL consistently improves regression performance and yields slightly improved, at least comparable performance on classification.

Figure 5: Elo ratings with 95% confidence intervals for all 18 models considered in Table 2.



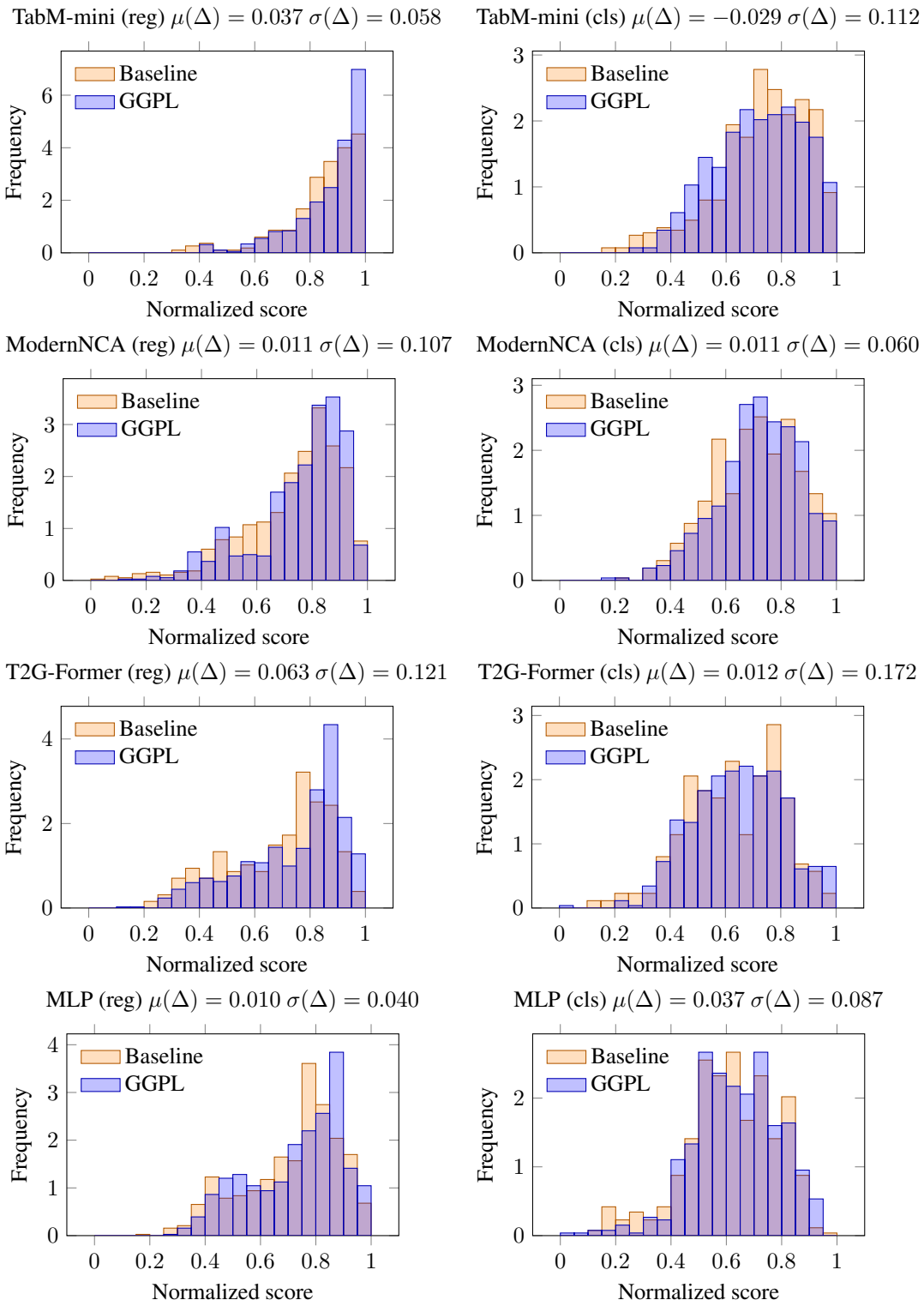


Figure 6: Histograms of normalized scores for regression (left) and classification (right) across all datasets and seeds for four backbones (TabM-mini, ModernNCA, T2G-Former, MLP). For each architecture we compare the native numerical embedding (Baseline) and GGPL, and report the mean $\mu(\Delta)$ and standard deviation $\sigma(\Delta)$ of the normalized-score gain Δ.