HYPERBOLIC MANIFOLD CONSTRAINED TABULAR NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular prediction is central to a wide range of real-world applications. Tabular data typically contain heterogeneous features as well as rich and complex relational information that can imply a latent structural manifold. Hyperbolic geometry can help capture complex structural relations in data. However, most existing tabular prediction models are constructed and optimized in Euclidean space. How to incorporate hyperbolic geometry into supervised tabular learning remains underexplored. We propose **HTNN**, a supervised hyperbolic manifold constrained tabular neural network for tabular prediction. HTNN consists of a hyperbolic feature-value representation layer for heterogeneous categorical and numerical features, followed by a conventional MLP predictor. HTNN employs a *geometry-aware training* and *geometry-free inference* optimization framework. The *geometry-aware training* allows hyperbolic geometry to shape the latent representation learning of heterogeneous feature values. After training, the latent hyperbolic representations can be converted into ordinary Euclidean space for efficient *geometry-free inference*. We conducted extensive experiments on the TALENT benchmark. HTNN ranks first among 36 methods on 200 classification datasets and third among 34 methods on 100 regression datasets. Experimental results show that the proposed hyperbolic manifold constrained tabular neural network is effective.

1 INTRODUCTION

Tabular prediction supports decision making across healthcare, finance, energy, and many other real-world domains (Borisov et al., 2024). Tabular data typically contain heterogeneous categorical (e.g., gender) and numerical (e.g., age) features, together with rich and complex relational information that can imply latent manifold structure (Choi, 2026; He et al., 2025). Unlike graphs or other explicitly structured inputs, tabular data provide no predefined connectivity or known hierarchy for these relationships, leaving them to be inferred from the data. Existing studies have sought to model these implicit relationships through advances in feature encoding, interaction modeling, and training strategies (Gorishniy et al., 2022; 2021; Holzmüller et al., 2025; Gorishniy et al., 2025). However, how to better learn latent representations of heterogeneous features and incorporate their implied manifold structure remains an open research question for tabular prediction models.

Hyperbolic geometry can help capture complex structural relationships in data. Under negative curvature, hyperbolic space expands rapidly with distance from the origin, providing a distinctive geometry for organizing such relationships. This property has motivated successful hyperbolic representation learning across hierarchies, graphs, vision, and multimodal learning (Nickel & Kiela, 2017; Chami et al., 2019; Dai et al., 2021; Desai et al., 2023). However, most existing tabular prediction models focus on Euclidean space (Holzmüller et al., 2025; Ye et al., 2025; Beaglehole et al., 2026; Grinsztajn et al., 2026). The use of hyperbolic geometry in supervised tabular learning remains underexplored. In particular, it remains unclear how heterogeneous features should be represented within a unified hyperbolic space and how the resulting manifold geometry shapes their learning.

In this paper, we propose **HTNN**, a supervised hyperbolic manifold constrained tabular neural network for tabular prediction. HTNN introduces a hyperbolic feature-value (HFV) representation layer with dedicated categorical and numerical pathways. Each categorical value is assigned an in-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

dividual learnable HFV representation. Numerical features, however, do not provide a fixed finite vocabulary for direct representation lookup. Each numerical value is therefore represented through an input-dependent composition of a small set of feature-specific HFV representations. Both pathways use tangent readouts of their HFV representations, which are concatenated and processed by a conventional MLP predictor. During *geometry-aware training*, Riemannian optimization allows negative curvature to shape representation updates, producing radial adaptability with transverse stabilization. After training, categorical tangent readouts are stored as ordinary lookup vectors, while the numerical pathway collapses into affine maps. This preserves the original predictions while removing all manifold operations, enabling *geometry-free inference*.

We evaluate HTNN at two complementary scales. The full 300-dataset TALENT benchmark (Ye et al., 2024) provides the main predictive comparison, while the 45-dataset TALENT-tiny suite is used for controlled studies of optimization behavior, model design, and computational efficiency. Our contributions are summarized as follows:

- We propose **HTNN**, a supervised hyperbolic manifold constrained tabular neural network that combines an HFV representation layer for heterogeneous categorical and numerical features with a conventional MLP predictor.
- We introduce a *geometry-aware training and geometry-free inference* framework: HFV representations are optimized on a hyperbolic manifold during training, and the trained HFV representation layer is converted into ordinary Euclidean lookup and affine operations for efficient inference.
- We conducted extensive comparison experiments. On the full TALENT benchmark, HTNN ranks first among 36 methods on 200 classification datasets and third among 34 methods on 100 regression datasets.

2 Related Work

Feature representations in tabular neural networks. Neural tabular models have explored a wide range of architectures and input representations. MLP-style and feature-attentive models learn from embedded feature values, while TabTransformer and FT-Transformer use attention to model feature interactions (Kadra et al., 2021; Arik & Pfister, 2020; Huang et al., 2020; Gorishniy et al., 2021). Numerical embedding methods enrich scalar inputs through piecewise-linear or periodic representations, and RealMLP and TabM improve training strategies, default configurations, and parameter-efficient ensembling (Gorishniy et al., 2022; Holzmüller et al., 2025; Gorishniy et al., 2025). Other approaches incorporate sample neighborhoods through retrieval or exploit cross-dataset context through pretrained foundation models (Gorishniy et al., 2024; Ye et al., 2025; 2026; QU et al., 2025). Despite this architectural diversity, most existing tabular models are constructed and optimized in Euclidean space. Consequently, the role of non-Euclidean geometry in supervised tabular learning remains underexplored.

Hyperbolic representation learning. Hyperbolic geometry has been widely used to represent hierarchical and relational information. Poincaré embeddings exploit negative curvature to embed explicit hierarchies with low distortion (Nickel & Kiela, 2017), while subsequent work extends hyperbolic learning to graph neural networks and visual or multimodal representations (Chami et al., 2019; Dai et al., 2021; Lensink et al., 2022; Desai et al., 2023). HyperDT constructs decision trees for samples that are already represented in hyperbolic space (Chlenski et al., 2024). These methods generally begin with an explicit relational structure or an existing sample-level representation. Without either, how hyperbolic geometry should be incorporated into tabular neural networks remains an open question.

Training parameterization and Riemannian optimization. Equivalent predictors can follow different optimization trajectories under different trainable parameterizations (Gunasekar et al., 2017; Arora et al., 2018). Riemannian optimization enables manifold-valued parameters to be trained while respecting their geometric constraints, with RiemannianAdam extending adaptive optimization to such parameters (Becigneul & Ganea, 2019). However, the effects of manifold constraints on feature-value learning and supervised tabular prediction remain insufficiently characterized.

3 METHODOLOGY

HTNN is a supervised tabular prediction model that uses hyperbolic representation learning to construct feature-value representations for heterogeneous inputs. An input $\mathbf{x}$ consists of categorical features $\{x_s^{\mathrm{c}} \in \mathcal{C}_s\}_{s\in\mathcal{J}_{\mathrm{c}}}$ and numerical features $\{x_j^{\mathrm{n}} \in \mathbb{R}\}_{j\in\mathcal{J}_{\mathrm{n}}}$, where $\mathcal{J}_{\mathrm{c}}$ and $\mathcal{J}_{\mathrm{n}}$ denote the two feature types and $\mathcal{C}_s$ is the value set of categorical feature s. HTNN learns hyperbolic feature-value (HFV) representations for both feature types. Each categorical value maps to an HFV representation, whereas each numerical value determines an input-dependent combination of the tangent readouts of a small set of feature-specific HFV representations. Both pathways produce vectors in a shared tangent space for a conventional MLP predictor. Figure 1 summarizes the complete pipeline: the HFV representation construction, *geometry-aware training*, and *geometry-free inference*.

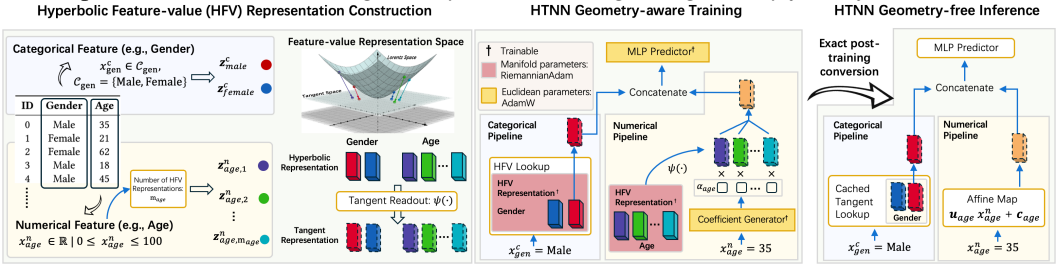

Figure 1: Overview of HTNN. Categorical values select individual HFV representations, whereas the feature-wise capacity m_j determines how many HFV representations are assigned to numerical feature j. Input-dependent coefficients combine their tangent readouts (left). HTNN jointly optimizes HFV representations and Euclidean parameters during *geometry-aware training* (center). Exact post-training conversion caches categorical tangent readouts and collapses each numerical composition into an affine map for *geometry-free inference* without changing predictions (right).

3.1 CATEGORICAL HFV REPRESENTATIONS

Both feature pathways use learnable HFV representations on a common Lorentz manifold. The Lorentz model represents a d-dimensional hyperbolic space as a hyperboloid in the ambient space $\mathbb{R}^{d+1}$, with one time-like coordinate and d spatial coordinates. For $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{d+1}$, the Lorentz inner product is

$$\langle \mathbf{z}, \mathbf{z}' \rangle_{\mathrm{L}} = -z_0 z_0' + \sum_{\ell=1}^{d} z_\ell z_\ell'. \tag{1}$$

Given a scale parameter $\kappa > 0$, the Lorentz manifold is the upper sheet of this hyperboloid:

$$\mathbb{H}_\kappa^d = \left\{ \mathbf{z} \in \mathbb{R}^{d+1} : \langle \mathbf{z}, \mathbf{z} \rangle_{\mathrm{L}} = -\kappa,\ z_0 > 0 \right\}. \tag{2}$$

This manifold has sectional curvature $-1/\kappa$ and origin $\mathbf{o} = (\sqrt{\kappa}, 0, \ldots, 0)$. To interface its HFV representations with a conventional neural predictor, we identify the tangent space at $\mathbf{o}$ with $\mathbb{R}^d$ through its spatial coordinates and define the tangent readout

$$\psi(\mathbf{z}) = \left[\log_{\mathbf{o}}^{\mathbb{H}}(\mathbf{z}) \right]_{1:d} \in \mathbb{R}^d, \tag{3}$$

where $\log_{\mathbf{o}}^{\mathbb{H}}$ is the Lorentz logarithmic map and $[\cdot]_{1:d}$ selects the spatial coordinates. Thus, each HFV representation remains constrained to the manifold during training, while its tangent readout provides a d-dimensional Euclidean vector to the downstream predictor.

For each categorical feature $s \in \mathcal{J}_{\mathrm{c}}$, HTNN assigns every value $c \in \mathcal{C}_s$ an HFV representation:

$$\mathbf{z}_{s,c}^{\mathrm{c}} \in \mathbb{H}_\kappa^d, \qquad c \in \mathcal{C}_s. \tag{4}$$

Given an observed value x_s^{c}, categorical lookup followed by the tangent readout produces

$$\mathbf{h}_s^{\mathrm{c}}\left(x_s^{\mathrm{c}}\right) = \psi\left(\mathbf{z}_{s,x_s^{\mathrm{c}}}^{\mathrm{c}}\right) \in \mathbb{R}^d. \tag{5}$$

3.2 NUMERICAL HFV REPRESENTATIONS

Unlike a categorical feature, a numerical feature does not provide a fixed finite vocabulary for direct representation lookup. HTNN therefore assigns each numerical feature $j \in \mathcal{J}_{\mathrm{n}}$ a feature-specific set of m_j learnable HFV representations:

$$\mathbf{Z}_j^{\mathrm{n}} = \left\{ \mathbf{z}_{j,k}^{\mathrm{n}} \right\}_{k=1}^{m_j}, \qquad \mathbf{z}_{j,k}^{\mathrm{n}} \in \mathbb{H}_\kappa^d. \tag{6}$$

Here, m_j is the HFV capacity of feature j, namely, the number of feature-specific HFV representations assigned to that feature. Each input value determines how their tangent readouts are combined into a single vector in $\mathbb{R}^d$, matching the output space of the categorical pathway.

Feature-wise HFV capacity. Numerical features can differ substantially in empirical diversity and observation support. HTNN therefore determines m_j once from the finite, non-missing training values of feature j, without using labels. Let $\widehat{\mathcal{H}}_j$ denote their empirical entropy in bits, n_j the number of finite, non-missing training observations, and $m_{\min}$ and $m_{\max}$ the global capacity bounds. For $n_j > 0$, we compute

$$\begin{aligned} m_j^{\text{ent}} &= \max\left\{m_{\min}, \operatorname{round}\left(2^{\widehat{\mathcal{H}}_j}\right)\right\}, \\ m_j^{\text{obs}} &= \max\left\{m_{\min}, \lfloor \log_2 n_j \rfloor\right\}, \\ m_j &= \min\left\{m_{\max}, m_j^{\text{ent}}, m_j^{\text{obs}}\right\}. \end{aligned} \tag{7}$$

The entropy-based term translates empirical diversity into a feature-specific capacity proposal, while the observation-based term limits it according to the available sample support. Their minimum enforces both constraints within the global range $[m_{\min}, m_{\max}]$. This rule determines only the number of HFV representations assigned to each feature; it neither defines value bins nor associates individual representations with particular values or intervals. The entropy estimator and edge-case handling are specified in Appendix B.

Input-dependent composition. For an observed value x_j^{n}, HTNN generates one coefficient for each HFV representation:

$$\boldsymbol{\alpha}_j(x_j^{\text{n}}) = \mathbf{a}_j x_j^{\text{n}} + \mathbf{b}_j \in \mathbb{R}^{m_j}, \tag{8}$$

where $\mathbf{a}_j, \mathbf{b}_j \in \mathbb{R}^{m_j}$ are learnable Euclidean parameters. The coefficients are neither normalized nor bounded, allowing numerical inputs to induce signed, scale-sensitive compositions.

Let $\mathbf{v}_{j,k}^{\text{n}} = \psi\left(\mathbf{z}_{j,k}^{\text{n}}\right) \in \mathbb{R}^d$ denote the tangent readout of the k-th HFV representation. The resulting numerical tangent representation is

$$\mathbf{h}_j^{\text{n}}(x_j^{\text{n}}) = \sum_{k=1}^{m_j} \alpha_{j,k}(x_j^{\text{n}}) \mathbf{v}_{j,k}^{\text{n}} \in \mathbb{R}^d. \tag{9}$$

Affine inference form. Because the coefficients are affine in x_j^{n}, the numerical composition reduces exactly to an affine map. Defining $\mathbf{u}_j = \sum_{k=1}^{m_j} a_{j,k} \mathbf{v}_{j,k}^{\text{n}}$ and $\mathbf{c}_j = \sum_{k=1}^{m_j} b_{j,k} \mathbf{v}_{j,k}^{\text{n}}$, we obtain

$$\mathbf{h}_j^{\text{n}}(x_j^{\text{n}}) = \mathbf{u}_j x_j^{\text{n}} + \mathbf{c}_j. \tag{10}$$

This identity separates the affine numerical representation used for prediction from the HFV-based parameterization used during training. Its optimization consequences and exact post-training conversion are analyzed in Section 3.4.

3.3 Prediction and Hybrid Optimization

Prediction. Using the categorical tangent representations in equation 5 and the numerical tangent representations in equation 9, HTNN forms the input to its predictor as

$$\mathbf{h}(\mathbf{x}) = \operatorname{concat}\left(\left\{\mathbf{h}_s^{\text{c}}(x_s^{\text{c}})\right\}_{s \in \mathcal{J}_{\text{c}}}, \left\{\mathbf{h}_j^{\text{n}}(x_j^{\text{n}})\right\}_{j \in \mathcal{J}_{\text{n}}}\right) \in \mathbb{R}^{d(|\mathcal{J}_{\text{c}}| + |\mathcal{J}_{\text{n}}|)}. \tag{11}$$

The concatenated representation is passed to an MLP with a task-specific output layer, $\widehat{y} = f_{\Theta_{\text{p}}}(\mathbf{h}(\mathbf{x}))$, where Θ_{p} denotes the predictor parameters. All parameters are trained jointly by minimizing the corresponding classification or regression loss $\mathcal{L}$.

Hybrid optimization. HTNN separates its parameters according to their geometric domains:

$$\begin{aligned} \Theta_{\text{H}} &= \left\{\mathbf{z}_{s,c}^{\text{c}}\right\}_{s \in \mathcal{J}_{\text{c}},\, c \in \mathcal{C}_s} \cup \left\{\mathbf{z}_{j,k}^{\text{n}}\right\}_{j \in \mathcal{J}_{\text{n}},\, 1 \le k \le m_j}, \\ \Theta_{\text{E}} &= \left\{\mathbf{a}_j, \mathbf{b}_j\right\}_{j \in \mathcal{J}_{\text{n}}} \cup \Theta_{\text{p}}. \end{aligned} \tag{12}$$

As summarized in the center panel of Figure 1, the HFV representations in Θ_{H} are optimized with RiemannianAdam, which maintains the Lorentz constraints, while the unconstrained Euclidean parameters Θ_{E} are optimized with AdamW. The Lorentz scale κ remains fixed throughout training.

3.4 Geometry-aware Training and Geometry-Free Inference Framework

We now characterize how the HFV-based parameterization underlying equation 10 changes the effective affine update and how its geometric operations can be removed after training.

Induced affine dynamics. Let $\mathbf{B}_j = [\mathbf{u}_j \quad \mathbf{c}_j] \in \mathbb{R}^{d\times 2}$ collect the effective affine parameters of numerical feature j. Under continuous-time gradient flow, direct optimization of the affine map and HFV-based optimization in HTNN yield

$$\dot{\mathbf{B}}_j = \begin{cases} -\nabla_{\mathbf{B}_j}\mathcal{L}, & \text{direct affine optimization,} \\ -\mathcal{K}_j\left[\nabla_{\mathbf{B}_j}\mathcal{L}\right], & \text{HTNN.} \end{cases} \tag{13}$$

Here, $\mathcal{K}_j$ is a state-dependent operator determined by the current HFV tangent readouts, their composition coefficients, and the local manifold geometry. The two parameterizations can therefore realize the same affine map while following different optimization trajectories.

Curvature-dependent updates. The operator $\mathcal{K}_j$ combines two update paths: the Euclidean coefficients change how the current HFV tangent readouts form $\mathbf{u}_j$ and $\mathbf{c}_j$, while the HFV representations change the readouts themselves. Hyperbolic geometry enters through the latter path.

Consider an HFV representation $\mathbf{z}$ with tangent readout $\mathbf{v} = \psi(\mathbf{z}) \neq \mathbf{0}$, and write $\mathbf{v} = r\mathbf{q}$, where $r = \|\mathbf{v}\|_2$ and $\mathbf{q} = \mathbf{v}/r$. For curvature $-1/\kappa$, let $\tau = \kappa^{-1/2}$. In these normal coordinates, the transverse metric scale grows as $\sinh(\tau r)/(\tau r)$, so the inverse metric increasingly attenuates transverse gradient components as r grows. The inverse metric operator in these coordinates is

$$P_{\mathrm{H}}(\mathbf{v}) = \mathbf{q}\mathbf{q}^\top + \left(\frac{\tau r}{\sinh(\tau r)}\right)^2 \left(I_d - \mathbf{q}\mathbf{q}^\top\right), \tag{14}$$

with the continuous extension $P_{\mathrm{H}}(\mathbf{0}) = I_d$. Accordingly, the HFV tangent readout evolves as $\dot{\mathbf{v}} = -P_{\mathrm{H}}(\mathbf{v})\nabla_{\mathbf{v}}\mathcal{L}$. The radial component is unchanged, whereas the transverse component is scaled by $(\tau r/\sinh(\tau r))^2 < 1$ for $r > 0$. Negative curvature therefore preserves radial responsiveness while increasingly attenuating transverse updates away from the origin. We refer to this combined behavior as radial adaptability with transverse stabilization.

Categorical HFV representations are shaped directly by the same geometry-dependent update. For numerical features, P_{H} enters the HFV-update component of $\mathcal{K}_j$, while the coefficient-update component remains Euclidean. The exact derivations of P_{H} and $\mathcal{K}_j$ are provided in Appendix A. Appendix F further verifies the predicted radial and transverse responses and examines their manifestation in practical optimization trajectories.

Geometry-free inference. As shown in the right panel of Figure 1, after training, each categorical tangent readout can be stored as an ordinary Euclidean lookup vector. For each numerical feature, the HFV representations and composition coefficients can be replaced exactly by the effective parameters $\mathbf{u}_j$ and $\mathbf{c}_j$ in equation 10. Inference then requires only Euclidean lookup, affine transformation, and the downstream MLP; it contains no manifold operations or input-dependent HFV composition, and its cost is independent of m_j. This conversion leaves the model predictions unchanged. HTNN thus realizes *geometry-aware training and geometry-free inference*.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate HTNN within the unified TALENT framework (Ye et al., 2024). The full TALENT benchmark contains 300 datasets (200 classification and 100 regression tasks) and supports broad comparison across heterogeneous tabular problems. TALENT-tiny is a compact 45-dataset core (30 classification and 15 regression tasks) curated under stricter quality and balance criteria for rapid, reproducible, and controlled in-depth evaluation. Following these roles, we use the full benchmark for the main predictive comparison and TALENT-tiny for controlled analyses of optimization dynamics, architecture, feature-wise HFV capacity, and computational efficiency.

Compared methods. We compare HTNN with 35 baselines on classification and 33 baselines on regression, yielding fixed task-specific pools of 36 and 34 methods, respectively. The pools span classical and tree-based methods, supervised neural architectures, retrieval-based methods, kernel-based feature learners, and tabular foundation models. For the efficiency study, we evaluate the union of the ten highest-ranked methods in the full-benchmark classification and regression comparisons. Full-benchmark baseline results are obtained either from TALENT's released result records or by executing the corresponding TALENT evaluation pipeline when no released record is available. For efficiency evaluation, the selected methods are rerun under a common timing protocol using the same selected configurations or released pretrained checkpoints as in the predictive comparison, without repeating hyperparameter search. The complete comparison pools, result provenance, and efficiency configurations are provided in Appendix C.2.

Evaluation protocol. We follow the official TALENT train, validation, and test splits, using accuracy (higher is better) for classification and root mean squared error (RMSE; lower is better) for regression. For each model–dataset pair, we report the mean test score over 15 random runs, with all downstream model-selection decisions made exclusively on the validation split. To aggregate performance across datasets, we rank methods within each dataset, with rank one denoting the best result, and use the mean rank as our primary summary statistic. We accompany the mean rank with its across-dataset standard deviation and the full distribution of per-dataset ranks. Pairwise win/tie/loss counts and multiplicity-adjusted statistical tests against all baselines are reported in Appendix D.1, with the comparison protocol detailed in Appendix C.3. For the efficiency analysis, fitting and inference times are normalized per 1K training and test samples, respectively. Each reported model-level value is the median across dataset-level means, where each dataset mean is computed over 15 runs. Detailed timing definitions are provided in Appendix C.4.

Implementation. We implement HTNN in PyTorch, using Geoopt for manifold operations and Riemannian optimization. We fix the Lorentz scale to $\kappa = 1.0$ and use $m_{\min} = 2$ and $m_{\max} = 12$ for feature-wise HFV capacity. Following TALENT's default neural-model training protocol, all HTNN runs use a batch size of 1,024, train for at most 200 epochs, and apply early stopping with a patience of 20 epochs. For each dataset, we perform 100 hyperparameter-optimization trials using Optuna (Akiba et al., 2019), searching over MLP depth, hidden width, and dropout rate, the representation dimension d, the Euclidean and Riemannian learning rates, and weight decay. Following Geoopt's recommendation for numerical stability in manifold computations, HTNN is trained and evaluated in double precision.[1] The complete hyperparameter search space is provided in Appendix C.5.

4.2 Main Results on TALENT

Figure 2 summarizes the per-dataset rank distributions of HTNN and representative high-performing methods from the complete task-specific comparison pools.

Classification. HTNN ranks first among all 36 methods, with a mean rank of 4.49, more than five positions ahead of the second-ranked TabPFN-v3. Its rank distribution remains concentrated near the top of the comparison pool across 200 datasets, showing that the aggregate advantage is broad rather than driven by a small subset of tasks. HTNN also outperforms the strongest method from every other represented model family, including all three foundation models. Pairwise tests confirm this result: after Holm correction, HTNN significantly outperforms each of the 35 classification baselines. The complete W/T/L counts and Holm-adjusted p-values are reported in Appendix D.1.

Regression. HTNN ranks third among the 34 methods, behind TabPFN-v3 and TabICL-v2, and is the highest-ranked non-foundation method. Despite trailing these two pretrained models, it significantly outperforms each of the remaining 31 baselines, including TabDPT and the strongest alternatives from every non-foundation model family. The complete W/T/L counts and Holm-adjusted p-values are reported in Appendix D.1.

[1] https://geoopt.readthedocs.io/en/latest/extended/stereographic.html#stability

Taken together, HTNN delivers the strongest classification result and the strongest regression result among non-foundation methods, demonstrating its competitiveness across different tabular tasks. Complete task-specific rankings and per-dataset scores are reported in Appendix D.

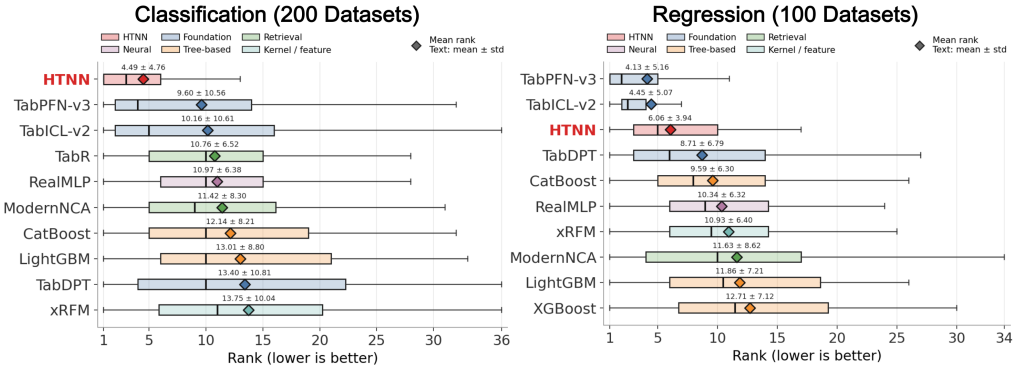

Figure 2: Per-dataset rank distributions on the full TALENT benchmark. Boxes summarize the rank distributions, and diamonds denote mean ranks. Annotations report the mean and across-dataset standard deviation. Lower ranks indicate better performance. Although only representative methods are displayed, ranks are computed using the complete task-specific comparison pools.

4.3 What Makes HTNN Work?

To isolate the contributions of HTNN's design choices, we compare it with eight controlled variants on TALENT-tiny, ranking all nine methods within a fixed variant pool. Each variant is tuned independently using the same HPO budget, preprocessing pipeline, training protocol, and search space for shared hyperparameters.

Table 1: Controlled architecture ablation on TALENT-tiny. Values are the mean and across-dataset standard deviation of ranks computed within the fixed nine-variant pool over 30 classification and 15 regression datasets. Lower is better.

Variant	Categorical representation	Numerical module	Coefficient map	Mean rank ↓ Classification	Mean rank ↓ Regression
HTNN	Lorentz	Lorentz composition	Affine	$\mathbf{1.38 \pm 0.98}$	$\mathbf{1.20 \pm 0.41}$
HTNN-Sphere	Spherical	Spherical composition	Affine	5.12 ± 1.58	4.67 ± 1.23
HTNN-Euc	Euclidean	Euclidean composition	Affine	5.68 ± 1.68	5.53 ± 1.06
HTNN-CHNE	Lorentz	Euclidean composition	Affine	5.58 ± 1.91	5.60 ± 2.16
HTNN-CENH	Euclidean	Lorentz composition	Affine	4.12 ± 3.03	2.40 ± 1.18
HTNN-CHNL	Lorentz	Direct affine	—	4.38 ± 1.88	4.73 ± 2.05
HTNN-CENL	Euclidean	Direct affine	—	5.12 ± 1.70	4.73 ± 2.28
HTNN-Softmax	Lorentz	Lorentz composition	Softmax	6.65 ± 1.60	7.93 ± 1.10
HTNN-Sigmoid	Lorentz	Lorentz composition	Sigmoid	6.97 ± 1.83	8.20 ± 0.94

Controlled variants. HTNN-Sphere and HTNN-Euc replace the Lorentz geometry in both feature pathways with their matched spherical (curvature $+1$) and Euclidean counterparts, respectively, while HTNN uses Lorentz geometry with curvature -1. HTNN-CHNE retains hyperbolic categorical representations but uses a Euclidean numerical composition, whereas HTNN-CENH makes the complementary replacement. HTNN-CHNL and HTNN-CENL replace the numerical composition module with the direct affine map in equation 10, paired with hyperbolic and Euclidean categorical representations, respectively. Finally, HTNN-Softmax and HTNN-Sigmoid retain the Lorentz numerical composition but apply the corresponding transformation to the coefficient vector in equation 8. Table 1 summarizes the results; exact variant formulations and pairwise comparisons are provided in Appendix E.

Negative curvature. HTNN achieves the best mean rank on both tasks and significantly outperforms its matched spherical and Euclidean counterparts. The spherical result shows that the gain is not a generic consequence of using a curved manifold, while the Euclidean result shows that the

matched compositional parameterization alone is insufficient. Together, these comparisons identify negative curvature as the effective geometric choice within HTNN. A controlled optimization-dynamics analysis further tests the inverse-metric behavior predicted by equation 14: radial response remains unattenuated, whereas transverse response is increasingly attenuated with radius. The measured local responses follow this prediction, with a maximum transverse-response error of 7.2×10^{-10}. The same directional effect persists in practical training trajectories: relative to HTNN-Euc, HTNN exhibits a lower transverse-update fraction on all 30 classification and all 15 regression datasets, with mean reductions of 8.77 and 9.00 percentage points, respectively. Full results are provided in Appendix F.

Heterogeneous feature pathways. HTNN-CHNE and HTNN-CENH each retain one hyperbolic feature pathway while replacing the other with its Euclidean counterpart. Among the eight ablation variants, HTNN-CENH performs best on both tasks, indicating a particularly strong contribution from the numerical HFV pathway, especially on regression. Nevertheless, neither single-hyperbolic-pathway variant matches the complete model, supporting complementary contributions from the categorical and numerical HFV pathways.

HFV-based training parameterization. HTNN-CHNL retains the same hyperbolic categorical pathway as HTNN but replaces the numerical HFV composition with a directly optimized affine map. Although this variant has the same numerical inference function class as HTNN, it performs substantially worse on both tasks. The gap therefore supports the benefit of learning the affine numerical map through HFV representations, consistent with the induced dynamics in equation 13.

Coefficient constraints. HTNN-Softmax and HTNN-Sigmoid occupy the final two positions on both tasks. Their degradation supports the use of unconstrained affine coefficients, which allow numerical inputs to produce signed and scale-sensitive compositions.

Across both tasks, HTNN significantly outperforms all eight variants after Holm correction. The corresponding W/T/L counts and statistical results are reported in Appendix E.

4.4 Feature-Wise Adaptive HFV Capacity

We examine whether using equation 7 for choosing the number m_j of HFV representations assigned to each numerical feature is more effective than using a single globally fixed representation capacity. On TALENT-tiny, we compare adaptive HTNN, with $m_j \in \{2, \ldots, 12\}$, against hyperbolic fixed-m variants for $m \in \{1, \ldots, 12\}$ and the corresponding fixed-m HTNN-Euc variants. All 25 variants are ranked within the same fixed comparison pool and tuned independently using the same HPO budget and training protocol.

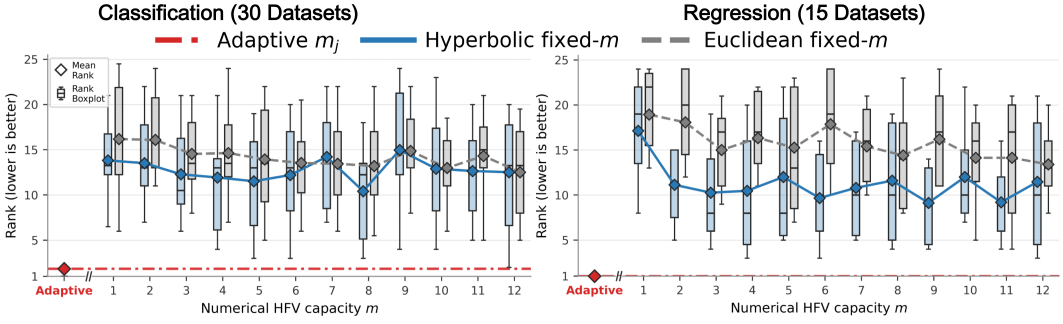

Figure 3: Feature-wise capacity analysis on TALENT-tiny. HTNN uses adaptive m_j, whereas each fixed-capacity variant uses the same m for all numerical features. Boxplots summarize per-dataset ranks, and diamonds denote mean ranks. All ranks are computed within the fixed 25-variant pool.

Feature-wise adaptive capacity. Adaptive HTNN outperforms every globally fixed alternative on both tasks, whereas fixed-m performance fluctuates rather than improving monotonically with m. The capacities computed by equation 7 are highly heterogeneous across numerical features (see Appendix G): no single m_j accounts for more than approximately one third of the features, while a substantial fraction of features receive the minimum capacity. Thus, the adaptive rule neither approximates a single global m nor uniformly enlarges the representation layer. Since every adaptive

m_j is bounded by the largest tested fixed m, these results support the motivation behind equation 7: matching each numerical feature's HFV capacity to its empirical information content yields better predictive performance than imposing a single global capacity.

Geometry under fixed capacity. At matched values of m, the hyperbolic variants outperform their Euclidean counterparts for $9/12$ settings on classification and $12/12$ on regression. This controlled pattern further supports the advantage of hyperbolic geometry observed in Section 4.3.

Complete ranking results and pairwise comparisons between adaptive HTNN and all 24 fixed-capacity variants are reported in Appendix G.

4.5 Efficiency: Training Cost and Geometry-Free Inference

Figure 4 summarizes the trade-off between predictive performance and computational efficiency on TALENT-tiny. For checkpoint-based foundation models, downstream fitting includes only their dataset-specific fitting stage and excludes pretraining. Per-dataset timing results are reported in Appendix H.

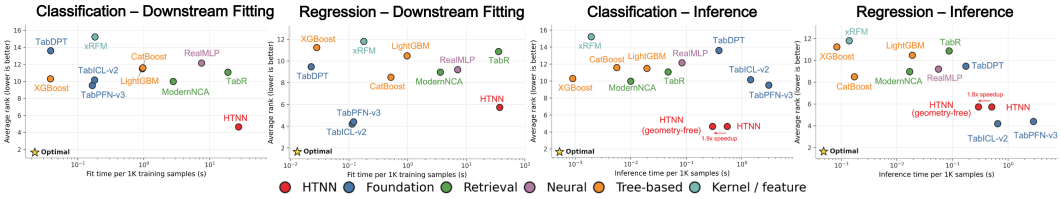

Figure 4: Performance versus downstream fitting and inference time on TALENT-tiny. Vertical axes show mean ranks over 30 classification or 15 regression datasets within the complete task-specific pools; horizontal axes show median time per 1K samples. The "Optimal" marker denotes the preferred lower-left region of low rank and runtime. HTNN (geometry-free) is the exact converted inference form of the same trained model.

Downstream fitting cost. The fitting panels make the computational trade-off explicit: HTNN achieves favorable ranks but incurs the largest median downstream fitting time among the displayed methods on both tasks. This cost reflects *geometry-aware training*, which combines Riemannian optimization of HFV representations with double-precision computation for numerical stability.

Geometry-free inference. We apply the exact conversion from Section 3.4 without retraining the model or changing its numerical precision. The converted model matches the original HTNN's outputs up to a maximum absolute difference of 5.47×10^{-17} and retains the same predictive rank, while accelerating inference on all 45 TALENT-tiny datasets. The median per-dataset speedup over the original HTNN is $1.87\times$ for classification and $1.84\times$ for regression. Despite continuing to operate entirely in double precision, HTNN (geometry-free) reaches an inference time comparable to TabDPT on classification and remains clearly faster than TabICL-v2 and TabPFN-v3 on both tasks. Thus, HTNN confines manifold-specific computation to *geometry-aware training*, while exact conversion enables *geometry-free inference* without changing predictions.

5 Conclusion

We introduced HTNN, a hyperbolic manifold constrained tabular neural network that learns categorical and numerical feature-value representations through *geometry-aware training* and converts exactly to ordinary Euclidean operations for *geometry-free inference*. Across the full 300-dataset TALENT benchmark, HTNN ranks first on classification and third on regression. Controlled studies support the roles of negative curvature, the complementary categorical and numerical pathways, and feature-wise HFV capacity, while the exact post-training conversion preserves predictions and accelerates inference. Together, these results demonstrate the effectiveness of hyperbolic geometry for supervised tabular learning.

HTNN requires additional training cost due to Riemannian optimization and double-precision manifold computation, and the present study focuses on supervised learning with a conventional MLP predictor. Future work will explore more efficient manifold optimization and extend hyperbolic feature-value representations to broader tabular architectures and learning settings.

AI USE STATEMENT

We used ChatGPT (Sol High) solely to assist with language polishing, review mathematical derivations for correctness, and provide suggestions for the design of ablation experiments. All suggestions were critically reviewed by the authors, who made the final methodological decisions, conducted the experiments, and verified the results, and who take full responsibility for the content of this work.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Sercan O. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning, 2020. URL https://openreview.net/forum?id=BylRkAEKDH.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International conference on machine learning*, pp. 244–253. PMLR, 2018.

Daniel Beaglehole, David Holzmüller, Adityanarayanan Radhakrishnan, and Mikhail Belkin. xRFM: Accurate, scalable, and interpretable feature learning models for tabular data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wHuVdpnUFp.

Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1eiqi09K7.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, 2024. doi: 10.1109/TNNLS.2022.3229161.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Philippe Chlenski, Ethan Turok, Antonio Khalil Moretti, and Itsik Pe'er. Fast hyperboloid decision tree algorithms. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TTonmgTT9X.

Jong In Choi. Tabgeoflow: A geometric flow matching model for tabular data synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 20562–20569, 2026.

Jindou Dai, Yuwei Wu, Zhi Gao, and Yunde Jia. A hyperbolic-to-hyperbolic graph convolutional network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 154–163, 2021.

Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Shanmukha Ramakrishna Vedantam. Hyperbolic image-text representations. In *International Conference on Machine Learning*, pp. 7694–7731. PMLR, 2023.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=pfI7u0eJAIr.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rhgIgTSSxW.

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, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. URL https://arxiv.org/abs/2605.13986.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. *Advances in neural information processing systems*, 30, 2017.

Zhipeng He, Alexander Stevens, Chun Ouyang, Johannes De Smedt, Alistair Barros, and Catarina Moreira. Crafting imperceptible on-manifold adversarial attacks for tabular data. *Applied Soft Computing*, pp. 114286, 2025.

David Holzmüller, Leo Grinsztajn, and Ingo Steinwart. RealMLP: Advancing MLPs and default parameters for tabular data. In *ELLIS workshop on Representation Learning and Generative Models for Structured Data*, 2025. URL https://openreview.net/forum?id=fwajDrDy89.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Keegan Lensink, Bas Peters, and Eldad Haber. Fully hyperbolic convolutional neural networks. *Research in the Mathematical Sciences*, 9(4):60, 2022.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pIZxEOZCId.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

Jingang QU, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0VvD1PmNzM.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.11139.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024.

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.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at tabPFN v2: Understanding its strengths and extending its capabilities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=lQYNmlTwtc.

A Optimization Analysis of the HFV-Based Training Parameterization

We derive the normal-coordinate operator $P_{\mathrm{H}}(\mathbf{v})$ and the induced affine dynamics $\mathcal{K}_j$ used in Section 3.4.

A.1 Normal-Coordinate Update

Consider a tangent readout $\mathbf{v} = \psi(\mathbf{z}) \neq \mathbf{0}$ and write $\mathbf{v} = r\mathbf{q}$, where $r = \|\mathbf{v}\|_2$ is its geodesic distance from the origin and $\mathbf{q} = \mathbf{v}/r$ is its radial direction. For curvature $-1/\kappa = -\tau^2$, where $\tau = \kappa^{-1/2}$, the pullback metric in these normal coordinates is

$$g_{\mathrm{H}}(\mathbf{v}) = \mathbf{q}\mathbf{q}^\top + \left(\frac{\sinh(\tau r)}{\tau r}\right)^2 \left(I_d - \mathbf{q}\mathbf{q}^\top\right). \tag{15}$$

The radial and transverse projectors are orthogonal. Inverting their corresponding eigenvalues gives $g_{\mathrm{H}}(\mathbf{v})^{-1} = P_{\mathrm{H}}(\mathbf{v})$, recovering equation 14. At the origin, the metric factors converge to one, so $g_{\mathrm{H}}(\mathbf{0}) = P_{\mathrm{H}}(\mathbf{0}) = I_d$. Hence, for a loss expressed in the normal coordinate $\mathbf{v}$, the corresponding Riemannian gradient flow is $\dot{\mathbf{v}} = -P_{\mathrm{H}}(\mathbf{v})\nabla_{\mathbf{v}}\mathcal{L}$.

A.2 Induced Affine Dynamics

Fix a numerical feature j. For the k-th HFV representation, define its tangent readout and coefficient pair as $\mathbf{v}^{\mathrm{n}}_{j,k} = \psi\left(\mathbf{z}^{\mathrm{n}}_{j,k}\right) \in \mathbb{R}^d, \boldsymbol{\beta}_{j,k} = \begin{bmatrix} a_{j,k} \\ b_{j,k} \end{bmatrix} \in \mathbb{R}^2$. The effective affine parameters in equation 10 can then be written as

$$\mathbf{B}_j = [\mathbf{u}_j \quad \mathbf{c}_j] = \sum_{k=1}^{m_j} \mathbf{v}^{\mathrm{n}}_{j,k} \boldsymbol{\beta}_{j,k}^\top \in \mathbb{R}^{d\times 2}. \tag{16}$$

Let $\mathbf{G}_j = \nabla_{\mathbf{B}_j}\mathcal{L} \in \mathbb{R}^{d\times 2}$, with the remaining model parameters held fixed. The chain rule gives

$$\nabla_{\mathbf{v}^{\mathrm{n}}_{j,k}}\mathcal{L} = \mathbf{G}_j\boldsymbol{\beta}_{j,k}, \qquad \nabla_{\boldsymbol{\beta}_{j,k}}\mathcal{L} = \mathbf{G}_j^\top \mathbf{v}^{\mathrm{n}}_{j,k}. \tag{17}$$

Accordingly, the geometry-dependent tangent-readout flow and Euclidean coefficient flow are

$$\dot{\mathbf{v}}^{\mathrm{n}}_{j,k} = -P_{\mathrm{H}}\left(\mathbf{v}^{\mathrm{n}}_{j,k}\right)\mathbf{G}_j\boldsymbol{\beta}_{j,k}, \qquad \dot{\boldsymbol{\beta}}_{j,k} = -\mathbf{G}_j^\top \mathbf{v}^{\mathrm{n}}_{j,k}. \tag{18}$$

Differentiating the HFV-based parameterization of $\mathbf{B}_j$ and substituting these updates yields $\dot{\mathbf{B}}_j = -\mathcal{K}_j[\mathbf{G}_j]$, where, for any $\mathbf{G} \in \mathbb{R}^{d\times 2}$,

$$\begin{aligned}\mathcal{K}_j[\mathbf{G}] = &\left(\sum_{k=1}^{m_j} \mathbf{v}^{\mathrm{n}}_{j,k}\left(\mathbf{v}^{\mathrm{n}}_{j,k}\right)^\top\right)\mathbf{G} \\ &+ \sum_{k=1}^{m_j} P_{\mathrm{H}}\left(\mathbf{v}^{\mathrm{n}}_{j,k}\right)\mathbf{G}\boldsymbol{\beta}_{j,k}\boldsymbol{\beta}_{j,k}^\top.\end{aligned} \tag{19}$$

The first term arises from coefficient updates and depends on the current HFV tangent readouts. The second arises from HFV updates and additionally incorporates the geometry-dependent operator

$P_{\mathrm{H}}(\mathbf{v}^{\mathrm{n}}_{j,k})$. Consequently, $\mathcal{K}_j$ evolves with both the HFV tangent readouts and their composition coefficients throughout training.

Role of HFV capacity. When $m_j = 1$, this parameterization constrains $\mathbf{B}_j$ to have rank at most one. For $m_j \geq 2$, any $\mathbf{B}_j = [\mathbf{u}_j, \mathbf{c}_j] \in \mathbb{R}^{d\times 2}$ can be represented by choosing

$$\mathbf{v}^{\mathrm{n}}_{j,1} = \mathbf{u}_j, \qquad \boldsymbol{\beta}_{j,1} = (1,0)^\top, \qquad \mathbf{v}^{\mathrm{n}}_{j,2} = \mathbf{c}_j, \qquad \boldsymbol{\beta}_{j,2} = (0,1)^\top,$$

with any remaining contributions set to zero. Additional HFV representations therefore provide alternative HFV-based parameterizations and modify $\mathcal{K}_j$, without changing the affine inference form.

Scope. The derivation uses unit-rate, unregularized continuous-time gradient flow to isolate the effects of the HFV-based training parameterization and manifold metric. RiemannianAdam and AdamW additionally introduce adaptive rescaling, momentum, weight decay, and discrete updates, so the equations above do not describe their complete trajectories. The analysis establishes that the two parameterizations induce different optimization dynamics.

B Feature-Wise HFV Capacity

For each numerical feature j, HTNN converts its training entries to numerical values and removes missing and non-finite observations. Let n_j denote the number of remaining valid observations. The resulting HFV capacity m_j is computed once before model optimization, remains fixed throughout training, and uses neither labels nor validation or test data. If $n_j = 0$, we set $m_j = m_{\min}$.

Entropy estimation. Let $p_{j,b}$ denote the empirical probability of state or histogram bin b for numerical feature j. We estimate its entropy in bits as

$$\widehat{\mathcal{H}}_j = -\sum_{b:p_{j,b}>0} p_{j,b}\log_2 p_{j,b}. \tag{20}$$

If the feature contains at most 30 distinct valid values, we treat them as discrete states and estimate $p_{j,b}$ from their empirical frequencies. Otherwise, we construct a histogram from the valid observations. Its bin width is selected using the Freedman–Diaconis rule,

$$h = 2\ \mathrm{IQR}\ n_j^{-1/3}. \tag{21}$$

The histogram frequencies are then normalized to obtain $p_{j,b}$. Accordingly, $2^{\widehat{\mathcal{H}}_j}$ represents the entropy-based effective number of empirical states rather than the raw number of distinct values.

Numerical safeguards. A constant feature is handled by the discrete branch above and yields an entropy estimate of zero. In the histogram branch, if the Freedman–Diaconis width is zero, as can occur when the interquartile range is zero, NumPy constructs a single-bin histogram, also yielding an entropy estimate of zero. If the estimated Freedman–Diaconis histogram would contain more than 10,240 bins, we instead use 10,240 equal-width bins. If histogram construction raises an exception, we fall back to 20 equal-width bins. The resulting entropy estimate and n_j are substituted into equation 7 to determine m_j.

C Experimental Details

C.1 Data Preprocessing

All preprocessing statistics and encoders are fitted exclusively on the training split and then applied unchanged to the validation and test splits. Missing numerical values are replaced with 0.0, after which numerical features are standardized using a `StandardScaler` fitted on the imputed training data. Missing categorical values are represented by the dedicated token `__missing__`; the categorical encoder is fitted on the training split, and previously unseen categories are mapped to a dedicated unknown index. The adaptive HFV capacity m_j is computed before these transformations using only the original finite, non-missing training values, without labels. Classification label mappings and regression-target standardization statistics are likewise determined solely from the training split.

Classification (36 methods)	
Family	**Methods**
Classical	Dummy, Logistic Regression, NCM, Naive Bayes, KNN, SVM
Tree-based	XGBoost, CatBoost, Random Forest, LightGBM
Supervised neural	MLP, ResNet, NODE, SwitchTab, TabNet, TabCaps, TANGOS, DANets, FT-Transformer, AutoInt, DCNv2, SNN, TabTransformer, PTaRL, GrowNet, MLP-PLR, RealMLP, ExcelFormer, TabM
Retrieval-based	TabR, ModernNCA
Kernel-based feature learning	xRFM
Foundation	TabPFN-v3, TabICL-v2, TabDPT
Proposed	HTNN
Regression (34 methods)	
Family	**Methods**
Classical	Dummy, Linear Regression, KNN, SVM, DNNR
Tree-based	XGBoost, CatBoost, Random Forest, LightGBM
Supervised neural	MLP, ResNet, NODE, SwitchTab, TabNet, TANGOS, DANets, FT-Transformer, AutoInt, DCNv2, SNN, TabTransformer, PTaRL, GrowNet, MLP-PLR, RealMLP, ExcelFormer, TabM
Retrieval-based	TabR, ModernNCA
Kernel-based feature learning	xRFM
Foundation	TabPFN-v3, TabICL-v2, TabDPT
Proposed	HTNN

Table 2: Complete task-specific comparison pools used for the full-benchmark rankings.

C.2 Compared Methods and Result Provenance

Full-benchmark comparison pools. Table 2 lists the complete task-specific comparison pools. The classification pool contains 35 baselines and HTNN, while the regression pool contains 33 baselines and HTNN.

Result provenance. For baselines with complete prediction records released by TALENT, we use the corresponding scores directly. For xRFM (Beaglehole et al., 2026) and TabM (Gorishniy et al., 2025), whose required full-benchmark scores were not available in the released result records, we execute their complete TALENT pipelines, including model-specific hyperparameter search, training, model selection, and test prediction. For TabPFN-v3, TabICL-v2, and TabDPT (Grinsztajn et al., 2026; Qu et al., 2026; Ma et al., 2025), we evaluate the released pretrained checkpoints through their corresponding TALENT interfaces. Under this evaluation protocol, these foundation models require neither downstream hyperparameter search nor parameter training.

Efficiency configurations. For every trainable method, we reuse the configuration selected during its predictive evaluation. For the foundation models, we reuse the same released checkpoint and inference configuration. No additional hyperparameter search is performed or included in the efficiency measurements.

C.3 Pairwise Comparison Protocol

For each task, we compare HTNN separately with every baseline over the corresponding datasets. Win/tie/loss counts are determined from the model–dataset test scores averaged over 15 runs and rounded to four decimal places. HTNN records a win when it achieves higher accuracy for classification or lower RMSE for regression, a loss under the opposite condition, and a tie when the

rounded scores are equal. If only one method has a valid score, the method with the valid score is counted as the winner; if both scores are missing, the comparison is counted as a tie.

For statistical testing, we use the same per-dataset ranks as in the full-benchmark summaries rather than raw score differences. Raw scores do not provide a uniform scale across different datasets: in classification, the same five-percentage-point accuracy gain can represent substantially different error reductions at different performance levels (e.g., 0.55 to 0.60 versus 0.95 to 1.00), while regression RMSE values and their absolute differences can vary by orders of magnitude with the target scale. Per-dataset ranks remove these scale differences and align the statistical analysis with our primary mean-rank summary. The ranks are computed from the unrounded mean scores using the complete task-specific comparison pool. Rank one denotes the best result, tied scores receive average ranks, and missing results receive tied worst ranks. We then apply a two-sided paired Wilcoxon signed-rank test to the per-dataset ranks of HTNN and each baseline, using the Pratt convention for zero differences. To control the family-wise error rate, the resulting p-values are corrected with the Holm procedure independently over the 35 classification and 33 regression comparisons. A comparison is considered significant when its Holm-adjusted p-value is below 0.05.

C.4 Timing Protocol

All efficiency measurements are collected on a single machine equipped with an NVIDIA GeForce RTX 4080 GPU and an Intel Xeon Platinum 8481C CPU. For GPU-based methods, the device is synchronized immediately before and after each timed region to account for asynchronous execution. Tree-based methods are executed on the CPU, while the remaining methods are executed on the GPU. Fitting and inference are timed separately over $R = 15$ runs, and all runs are retained.

Downstream fitting time is the elapsed time of the dataset-specific `fit` stage. For trainable methods, this stage includes model optimization under the selected configuration. For checkpoint-based foundation models, it corresponds to their dataset-specific `fit` call and excludes foundation-model pretraining. Inference time is the elapsed time required to generate predictions for the complete test partition.

Let $t^{\mathrm{fit}}_{i,d,r}$ and $t^{\mathrm{inf}}_{i,d,r}$ denote the fitting and inference times of model i on dataset d in run r. Their dataset-level mean times, normalized to 1K rows, are

$$\bar{t}^{\mathrm{fit}}_{i,d} = \frac{10^3}{RN^{\mathrm{train}}_d} \sum_{r=1}^{R} t^{\mathrm{fit}}_{i,d,r}, \tag{22}$$

$$\bar{t}^{\mathrm{inf}}_{i,d} = \frac{10^3}{RN^{\mathrm{test}}_d} \sum_{r=1}^{R} t^{\mathrm{inf}}_{i,d,r}. \tag{23}$$

The model-level values in Figure 4 are the medians of these dataset-level means over the corresponding classification or regression datasets. The per-dataset tables additionally report the standard deviation across the 15 normalized run-level times.

C.5 HTNN Hyperparameter Search Space

Hyperparameter	Search space	Hyperparameter	Search space
MLP depth	Integer in $[1, 8]$	Hidden width per layer	Integer in $[64, 512]$
Dropout rate per layer	0 or $\mathrm{Uniform}(0, 0.5)$	Representation dimension d	Integer in $[4, 32]$
Euclidean learning rate	$\mathrm{LogUniform}(10^{-5}, 10^{-2})$	Riemannian learning rate	$\mathrm{LogUniform}(10^{-5}, 10^{-2})$
Weight decay	0 or $\mathrm{LogUniform}(10^{-6}, 10^{-3})$		

Table 3: Hyperparameter search space for HTNN.

D Full-Benchmark Results

The full-benchmark result matrices are nearly complete: 17 of the 7,000 baseline–dataset entries are missing for classification, and 5 of the 3,300 entries are missing for regression. HTNN has valid

results on all 300 datasets. We retain the fixed task-specific comparison pools without removing any method or dataset; missing entries are handled according to the protocol in Appendix C.3.

D.1 Pairwise Statistical Comparisons

Table 4 reports the complete pairwise comparisons between HTNN and every baseline in the task-specific pools. On classification, all 35 comparisons favor HTNN and remain significant after Holm correction. On regression, TabPFN-v3 and TabICL-v2 significantly outperform HTNN, whereas HTNN significantly outperforms each of the remaining 31 baselines.

Table 4: Complete pairwise comparisons on the full TALENT benchmark. W/T/L counts are reported from the perspective of HTNN. Holm correction is applied independently over all 35 classification and 33 regression comparisons.

Classification (200 datasets)			Regression (100 datasets)		
Baseline	**W/T/L**	p_{Holm}	**Baseline**	**W/T/L**	p_{Holm}
TabPFN-v3	112/2/86	1.40×10^{-4}	TabPFN-v3	30/2/68	3.56×10^{-3}
TabICL-v2	115/1/84	2.44×10^{-5}	TabICL-v2	28/2/70	1.77×10^{-3}
TabR	159/2/39	1.21×10^{-16}	TabDPT	52/4/44	1.49×10^{-2}
RealMLP	158/2/40	1.11×10^{-18}	CatBoost	59/2/39	1.65×10^{-3}
ModernNCA	153/1/46	3.54×10^{-15}	RealMLP	62/5/33	1.93×10^{-5}
CatBoost	158/1/41	1.38×10^{-17}	xRFM	76/2/22	8.56×10^{-9}
LightGBM	162/1/37	1.07×10^{-18}	ModernNCA	65/2/33	2.00×10^{-5}
TabDPT	143/4/53	2.56×10^{-16}	LightGBM	68/2/30	7.75×10^{-6}
xRFM	153/4/43	5.08×10^{-18}	XGBoost	66/2/32	2.75×10^{-7}
XGBoost	161/3/36	1.38×10^{-19}	TabR	71/3/26	3.79×10^{-7}
FT-Transformer	186/0/14	7.05×10^{-29}	FT-Transformer	85/3/12	8.40×10^{-13}
MLP-PLR	179/1/20	1.75×10^{-27}	MLP-PLR	82/3/15	8.40×10^{-13}
ResNet	189/2/9	2.09×10^{-30}	ExcelFormer	88/2/10	6.48×10^{-15}
DCNv2	191/1/8	1.19×10^{-30}	Random Forest	76/2/22	5.39×10^{-11}
MLP	192/0/8	4.10×10^{-31}	PTaRL	95/3/2	3.32×10^{-16}
AutoInt	195/1/4	2.10×10^{-32}	AutoInt	95/2/3	1.64×10^{-16}
SNN	193/1/6	1.12×10^{-31}	NODE	86/2/12	2.06×10^{-14}
Random Forest	179/2/19	6.15×10^{-28}	DCNv2	97/2/1	1.41×10^{-16}
ExcelFormer	187/0/13	2.33×10^{-30}	MLP	98/2/0	1.21×10^{-16}
TANGOS	196/1/3	2.10×10^{-32}	ResNet	96/2/2	1.41×10^{-16}
TabCaps	197/0/3	8.49×10^{-33}	TabM	88/2/10	3.11×10^{-15}
PTaRL	195/1/4	3.76×10^{-32}	TANGOS	97/2/1	1.21×10^{-16}
DANets	197/1/2	4.99×10^{-33}	SNN	96/2/2	1.83×10^{-16}
TabM	183/1/16	5.48×10^{-29}	KNN	92/2/6	4.78×10^{-16}
NODE	191/1/8	3.35×10^{-32}	TabNet	98/2/0	1.21×10^{-16}
Logistic Regression	191/3/6	1.12×10^{-31}	DNNR	99/1/0	1.21×10^{-16}
TabTransformer	194/0/6	6.51×10^{-32}	GrowNet	98/2/0	1.21×10^{-16}
KNN	188/2/10	1.40×10^{-31}	SVM	98/2/0	1.21×10^{-16}
SVM	191/1/8	2.99×10^{-32}	Linear Regression	98/2/0	1.21×10^{-16}
GrowNet	199/0/1	4.74×10^{-33}	TabTransformer	100/0/0	1.21×10^{-16}
SwitchTab	199/0/1	4.57×10^{-33}	Dummy	100/0/0	1.21×10^{-16}
TabNet	199/0/1	4.57×10^{-33}	SwitchTab	100/0/0	1.21×10^{-16}
Naive Bayes	200/0/0	4.57×10^{-33}	DANets	100/0/0	1.21×10^{-16}
Dummy	199/1/0	4.57×10^{-33}			
NCM	199/0/1	4.57×10^{-33}			

D.2 Complete Task-Specific Rankings

Table 5 reports the complete task-specific rankings used in the full-benchmark comparison.

Table 5: Complete rankings on the full TALENT benchmark. Values are the mean and standard deviation of per-dataset ranks. Lower mean ranks indicate better overall performance. The proposed method is highlighted in bold.

Classification (200 datasets)			Regression (100 datasets)		
Rank	**Method**	**Mean ± Std**	**Rank**	**Method**	**Mean ± Std**
1	**HTNN**	$\mathbf{4.4875 \pm 4.7568}$	1	TabPFN-v3	4.1300 ± 5.1594
2	TabPFN-v3	9.5950 ± 10.5644	2	TabICL-v2	4.4500 ± 5.0700
3	TabICL-v2	10.1575 ± 10.6116	3	**HTNN**	$\mathbf{6.0600 \pm 3.9359}$
4	TabR	10.7550 ± 6.5242	4	TabDPT	8.7100 ± 6.7917
5	RealMLP	10.9700 ± 6.3779	5	CatBoost	9.5900 ± 6.3047
6	ModernNCA	11.4175 ± 8.3045	6	RealMLP	10.3400 ± 6.3229
7	CatBoost	12.1450 ± 8.2052	7	xRFM	10.9300 ± 6.4012
8	LightGBM	13.0125 ± 8.7968	8	ModernNCA	11.6300 ± 8.6224
9	TabDPT	13.3975 ± 10.8054	9	LightGBM	11.8550 ± 7.2105
10	xRFM	13.7525 ± 10.0369	10	XGBoost	12.7150 ± 7.1238
11	XGBoost	13.8250 ± 8.7183	11	TabR	12.8800 ± 8.1829
12	FT-Transformer	14.7925 ± 7.3149	12	FT-Transformer	14.3650 ± 7.3649
13	MLP-PLR	15.0900 ± 6.8650	13	MLP-PLR	14.9300 ± 7.9035
14	ResNet	15.9900 ± 7.2203	14	ExcelFormer	15.3550 ± 6.5595
15	DCNv2	16.5650 ± 7.2228	15	Random Forest	15.4950 ± 8.1496
16	MLP	16.7725 ± 6.7343	16	PTaRL	15.8250 ± 5.5470
17	AutoInt	17.0850 ± 6.0929	17	AutoInt	16.9900 ± 5.9144
18	SNN	17.6700 ± 6.2441	18	NODE	17.1450 ± 8.1984
19	Random Forest	17.8925 ± 9.2485	19	DCNv2	17.8050 ± 7.8852
20	ExcelFormer	18.3575 ± 7.4587	20	MLP	17.9200 ± 5.6469
21	TANGOS	18.7550 ± 6.6708	21	ResNet	18.1850 ± 6.5016
22	TabCaps	18.8600 ± 6.4329	22	TabM	18.8700 ± 7.3741
23	PTaRL	19.5050 ± 6.8472	23	TANGOS	18.9600 ± 5.9952
24	DANets	19.8725 ± 6.7041	24	SNN	19.7550 ± 6.0784
25	TabM	21.5500 ± 10.7237	25	KNN	21.5450 ± 7.1852
26	NODE	21.9050 ± 9.7890	26	TabNet	24.2500 ± 4.6103
27	Logistic Regression	22.4025 ± 9.7989	27	DNNR	24.9800 ± 7.0061
28	TabTransformer	22.5725 ± 8.6433	28	GrowNet	25.6950 ± 4.4670
29	KNN	23.2075 ± 9.7304	29	SVM	25.8450 ± 5.1868
30	SVM	24.4775 ± 9.1442	30	Linear Regression	26.0850 ± 5.0071
31	GrowNet	25.9450 ± 7.1921	31	TabTransformer	29.9100 ± 5.8652
32	SwitchTab	27.1575 ± 5.6443	32	Dummy	30.3750 ± 4.9074
33	TabNet	28.0375 ± 5.2934	33	SwitchTab	30.4550 ± 4.3351
34	Naive Bayes	31.9400 ± 5.7296	34	DANets	30.9700 ± 5.8109
35	Dummy	32.8775 ± 7.1158			
36	NCM	33.2050 ± 4.2374			

D.3 Per-Dataset HTNN Results

Table 6 reports HTNN's per-dataset test results on the full TALENT benchmark. For compact presentation, we use the concise dataset display names adopted in Table 9 of Ye et al. (2026); this changes only their presentation.

Table 6: Per-dataset test results for HTNN on the full TALENT benchmark. Classification entries report accuracy (%), and regression entries report RMSE. Values are means ± standard deviations over 15 runs.

Dataset	Mean ± Std	Dataset	Mean ± Std
Classification (200 datasets)			
abalone	64.68 ± 0.21	KDD	79.61 ± 0.19
accelerometer	74.44 ± 0.07	kdd_ipums_la_97-small	86.45 ± 0.19
ada	86.71 ± 0.14	KDDCup09_upselling	81.41 ± 0.15
ada_agnostic	84.69 ± 0.15	kr-vs-k	89.64 ± 0.15
ada_prior	85.63 ± 0.13	kr-vs-kp	99.75 ± 0.13
adult	86.09 ± 0.02	kropt	91.80 ± 0.08
airlines_2000	67.25 ± 0.42	law-school-admission-binary	100.00 ± 0.00

Continued on the next page.

Table 6 continued.

Dataset	Mean ± Std	Dataset	Mean ± Std
allbp	98.03 ± 0.04	led24	75.00 ± 0.23
allrep	98.41 ± 0.10	led7	75.12 ± 0.09
Amazon_employee_access	94.64 ± 0.02	letter	98.19 ± 0.04
analcatdata_authorship	99.64 ± 0.29	Long	99.93 ± 0.05
artificial-characters	75.18 ± 0.44	madeline	65.41 ± 0.45
ASP-POTASSCO-classification	44.59 ± 0.40	MagicTelescope	88.51 ± 0.05
autoUniv-au4-2500	63.26 ± 0.66	mammography	99.09 ± 0.02
autoUniv-au7-1100	40.05 ± 0.43	Marketing_Campaign	89.96 ± 0.37
bank	91.04 ± 0.04	maternal_health_risk	83.60 ± 0.66
Bank_Customer_Churn_Dataset	87.90 ± 0.05	mfeat-factors	97.30 ± 0.10
banknote_authentication	57.13 ± 0.68	mfeat-fourier	89.75 ± 0.39
baseball	95.56 ± 0.20	mfeat-karhunen	96.77 ± 0.08
Basketball_c	73.28 ± 0.25	mfeat-morphological	79.00 ± 0.16
BLE_RSSI_localization	74.51 ± 0.11	mfeat-pixel	97.40 ± 0.17
BNG(breast-w)	98.63 ± 0.03	mfeat-zernike	87.88 ± 0.54
BNG(cmc)	58.62 ± 0.03	MIC	85.03 ± 0.77
BNG(tic-tac-toe)	82.06 ± 0.04	mice_protein_expression	100.00 ± 0.00
California-Housing-Classification	89.32 ± 0.06	microaggregation2	64.07 ± 0.06
car-evaluation	100.00 ± 0.00	mobile_c36_oversampling	98.51 ± 0.03
Cardiovascular-Disease-dataset	73.56 ± 0.04	Mobile_Price_Classification	97.68 ± 0.22
churn	96.28 ± 0.12	mozilla4	93.11 ± 0.08
Click_prediction_small	83.45 ± 0.00	naticusdroid+android+permissions	97.16 ± 0.02
cmc	61.46 ± 0.43	national-longitudinal-survey-binary	99.69 ± 0.00
company_bankruptcy_prediction	97.38 ± 0.03	Nutrition_Health_Survey	85.37 ± 0.28
compass	78.11 ± 0.24	okcupid_stem	75.52 ± 0.08
connect-4	86.08 ± 0.10	one-hundred-plants-margin	85.41 ± 0.31
Contaminant-10.0GHz	95.63 ± 0.09	one-hundred-plants-shape	74.34 ± 0.57
Contaminant-10.5GHz	95.83 ± 0.21	one-hundred-plants-texture	89.38 ± 0.40
Contaminant-11.0GHz	95.10 ± 0.21	online_shoppers	90.82 ± 0.13
Contaminant-9.0GHz	93.27 ± 0.28	optdigits	99.08 ± 0.08
Contaminant-9.5GHz	93.65 ± 0.14	ozone-level-8hr	96.00 ± 0.09
contraceptive_method_choice	63.69 ± 0.39	ozone_level	98.23 ± 0.00
credit	76.52 ± 0.06	page-blocks	97.48 ± 0.06
credit-g	81.05 ± 0.57	pc1	94.64 ± 0.14
Credit_c	77.25 ± 0.06	pc3	90.22 ± 0.46
Customer_Personality_Analysis	90.42 ± 0.35	pc4	91.71 ± 0.34
customer_satisfaction_in_airline	96.18 ± 0.02	pendigits	99.70 ± 0.02
dabetes_130-us_hospitals	64.52 ± 0.08	Performance-Prediction	76.04 ± 0.44
default_of_credit_card_clients	82.82 ± 0.04	philippine	73.29 ± 0.20
delta_ailerons	95.06 ± 0.04	PhishingWebsites	97.88 ± 0.06
Diabetic_Retinopathy_Debrecen	79.78 ± 0.48	phoneme	89.74 ± 0.21
dis	98.98 ± 0.06	PieChart3	89.40 ± 0.60
dna	96.30 ± 0.08	Pima_Indians_Diabetes_Database	81.82 ± 0.92
drug_consumption	40.72 ± 0.32	PizzaCutter3	90.24 ± 0.38
dry_bean_dataset	93.15 ± 0.03	pol	99.37 ± 0.04
E-CommereShippingData	67.40 ± 0.29	predict_students_dropout	76.89 ± 0.17
eeg-eye-state	97.21 ± 0.22	Pumpkin_Seeds	87.70 ± 0.13
electricity	87.21 ± 0.13	QSAR_biodegradation	91.28 ± 0.38
Employee	86.36 ± 0.11	Rain_in_Australia	86.36 ± 0.03
estimation_of_obesity_levels	98.56 ± 0.20	rice_cammeo_and_osmancik	93.01 ± 0.08
eucalyptus	77.03 ± 0.60	ringnorm	98.22 ± 0.03
eye_movements	65.58 ± 0.31	rl	73.04 ± 0.46
eye_movements_bin	60.74 ± 0.17	satimage	92.37 ± 0.19
FICO-HELOC-cleaned	74.41 ± 0.15	SDSS17	97.24 ± 0.02
Firm-Teacher-Direction	87.81 ± 0.13	segment	93.46 ± 0.19
first-order-theorem-proving	59.64 ± 0.20	seismic+bumps	93.44 ± 0.06
Fitness_Club_c	80.00 ± 0.39	semeion	94.11 ± 0.39
FOREX_audcad-day-High	76.19 ± 0.25	shill-bidding	91.15 ± 0.04
FOREX_audcad-hour-High	71.93 ± 0.07	Shipping	68.86 ± 0.16
FOREX_audchf-day-High	77.36 ± 0.19	shrutime	87.26 ± 0.11
FOREX_audjpy-day-High	78.39 ± 0.42	shuttle	99.99 ± 0.00
FOREX_audjpy-hour-High	71.18 ± 0.15	spambase	94.77 ± 0.08
FOREX_audsgd-hour-High	69.13 ± 0.18	splice	95.41 ± 0.23
FOREX_audusd-hour-High	67.76 ± 0.17	sports_articles	85.95 ± 0.35
FOREX_cadjpy-day-High	75.12 ± 0.59	statlog	75.55 ± 0.47
FOREX_cadjpy-hour-High	69.65 ± 0.20	steel_plates_faults	79.85 ± 0.49
GAMETES_Epistasis	66.06 ± 0.88	svmguide3	85.46 ± 0.30
GAMETES_Heterogeneity	70.03 ± 0.94	sylvine	95.38 ± 0.21
gas-drift	99.50 ± 0.02	taiwanese_bankruptcy_prediction	97.20 ± 0.06
Gender_Gap_in_Spanish_WP	61.89 ± 0.22	telco-customer-churn	80.59 ± 0.07
GesturePhaseSegmentationProcessed	68.28 ± 0.19	Telecom_Churn_Dataset	94.38 ± 0.12
golf_play_dataset_extended	95.80 ± 0.49	texture	99.91 ± 0.00
Heart-Disease-Dataset	92.98 ± 0.38	thyroid	99.39 ± 0.04
helena	38.90 ± 0.06	thyroid-ann	99.11 ± 0.06
heloc	73.13 ± 0.09	thyroid-dis	71.27 ± 0.35
hill-valley	79.59 ± 2.12	turiye_student_evaluation	53.21 ± 0.23
house_16H	88.34 ± 0.07	twonorm	98.20 ± 0.05
HR_Analytics	80.34 ± 0.09	UJI_Pen_Characters	56.30 ± 0.40

Continued on the next page.

Table 6 continued.

Dataset	Mean ± Std	Dataset	Mean ± Std
htru	98.09 ± 0.02	vehicle	85.88 ± 0.64
ibm-employee-performance	99.83 ± 0.17	VulNoneVul	98.96 ± 0.03
in_vehicle_coupon	77.54 ± 0.13	walking-activity	64.90 ± 0.11
Indian_pines	94.66 ± 0.14	wall-robot-navigation	96.79 ± 0.10
INNHotelsGroup	88.52 ± 0.14	water_quality	91.68 ± 0.15
Insurance	75.77 ± 0.03	Water_Quality_and_Potability	66.20 ± 0.27
internet_firewall	79.79 ± 0.00	Waterstress	72.18 ± 0.56
internet_usage	53.51 ± 0.17	waveform-5000	87.53 ± 0.20
Intersectional-Bias-Assessment	94.89 ± 0.05	waveform_v1	87.74 ± 0.15
Is-this-a-good-customer	89.97 ± 0.23	website_phishing	92.84 ± 0.24
JapaneseVowels	99.52 ± 0.03	Wilt	99.48 ± 0.00
jasmine	80.55 ± 0.17	wine	75.85 ± 0.35
jm1	82.05 ± 0.14	wine-quality-red	63.97 ± 0.52
jungle_chess_2pcs_raw_endgame	98.99 ± 0.07	wine-quality-white	63.65 ± 0.39
kc1	88.96 ± 0.12	yeast	61.85 ± 0.48
Regression (100 datasets)			
1000-Cameras-Dataset	462.8 ± 5.823	houses	$4.87\times10^{4} \pm 104.4$
2dplanes	$1.001 \pm 1.13\times10^{-4}$	housing_price_prediction	$9.91\times10^{5} \pm 6758$
Abalone_reg	2.076 ± 0.001796	IEEE80211aa-GATS	$0.01949 \pm 5.34\times10^{-4}$
Ailerons	$1.52\times10^{-4} \pm 3.26\times10^{-7}$	Job_Profitability	13 ± 0.0293
airfoil_self_noise	1.229 ± 0.01859	kin8nm	$0.06502 \pm 3.77\times10^{-4}$
analcatdata_supreme	0.07597 ± 0.001587	Laptop_Prices_Dataset	452.4 ± 2.397
archive2	357 ± 5.475	mauna-loa-atmospheric-co2	0.4715 ± 0.007237
archive_r56_Portuguese	2.732 ± 0.03814	MiamiHousing2016	$8.59\times10^{4} \pm 428.5$
auction_verification	1991 ± 102.3	MIP-2016-regression	$2.4\times10^{4} \pm 102.7$
avocado_sales	$0.1072 \pm 4.01\times10^{-4}$	Mobile_Phone_Market	741.9 ± 19.13
bank32nh	$0.07593 \pm 1.42\times10^{-4}$	Moneyball	20.16 ± 0.1555
bank8FM	$0.02776 \pm 7.13\times10^{-5}$	mv	$0.005557 \pm 3.63\times10^{-4}$
Bias_correction_r	0.7375 ± 0.004439	NASA_PHM2008	40.17 ± 0.01303
Bias_correction_r_2	0.5539 ± 0.003991	NHANES_age_prediction	15.36 ± 0.01155
bike_sharing_demand	70.58 ± 0.2054	OnlineNewsPopularity	8572 ± 1.313
BNG(echoMonths)	11.53 ± 0.006885	Parkinson_Sound_Record	14.91 ± 0.0507
BNG(lowbwt)	456.6 ± 0.0982	Parkinsons_Telemonitoring	1.704 ± 0.1426
BNG(mv)	4.603 ± 0.002507	Physicochemical_r	3.369 ± 0.005011
BNG(stock)	2.95 ± 0.003955	pol_reg	2.008 ± 0.03007
boston	3.862 ± 0.04451	pole	1.957 ± 0.03273
Brazilian_houses_reproduced	$0.003602 \pm 5.47\times10^{-4}$	puma32H	$0.005717 \pm 2.49\times10^{-5}$
chscase_foot	0.8221 ± 0.001891	puma8NH	3.229 ± 0.001481
colleges	$0.1557 \pm 5.71\times10^{-4}$	qsar_aquatic_toxicity	0.9467 ± 0.0103
combined_cycle_power_plant	3.709 ± 0.02767	qsar_fish_toxicity	0.8468 ± 0.005675
communities_and_crime	$0.1318 \pm 3.96\times10^{-4}$	RSSI_Estimation	0.005501 ± 0.001442
concrete_compressive_strength	4.441 ± 0.05308	RSSI_Estimation1	$0.002037 \pm 7.46\times10^{-4}$
CookbookReviews	$1.489 \pm 4.08\times10^{-4}$	SAT11-HAND-runtime	1062 ± 3.758
CPMP-2015-regression	491 ± 4.679	satellite_image	0.6004 ± 0.008006
CPS1988	363.6 ± 0.0927	sensory	0.6752 ± 0.006116
cpu_act	2.312 ± 0.01599	Shop_Customer_Data	28.42 ± 0.02303
cpu_small	2.845 ± 0.01152	socmob	15.93 ± 2.231
Data_Science_Salaries	$5.96\times10^{4} \pm 91.8$	space_ga	$0.09417 \pm 4.25\times10^{-4}$
dataset_sales	3.889 ± 0.006586	steel_industry_energy	0.386 ± 0.04383
debutanizer	0.06088 ± 0.001625	stock	0.704 ± 0.009581
delta_elevators	$0.001367 \pm 3.86\times10^{-7}$	stock_fardamento02	18.43 ± 0.05335
Diamonds	564.1 ± 6.122	sulfur	$0.01796 \pm 5.11\times10^{-4}$
elevators	$0.00184 \pm 3.04\times10^{-6}$	Superconductivty	10.16 ± 0.04896
Facebook_Comment_Volume	16.58 ± 0.1995	topo_2_1	$0.03468 \pm 4.39\times10^{-6}$
Fiat	748.6 ± 2.073	treasury	0.2015 ± 0.00453
fifa	$0.7925 \pm 9.99\times10^{-4}$	us_crime	$0.1351 \pm 3.62\times10^{-4}$
Food_Delivery_Time	7.836 ± 0.002023	volume	39.62 ± 0.3411
fried	$1.004 \pm 4\times10^{-4}$	Wave_Energy_Perth_100	$1.68\times10^{4} \pm 211.7$
garments_worker_productivity	0.1475 ± 0.001885	Wave_Energy_Sydney_100	$1.93\times10^{4} \pm 407.1$
gas_turbine_emission	0.4401 ± 0.003682	Wave_Energy_Sydney_49	4580 ± 48.73
Goodreads-Computer-Books	0.4812 ± 0.001904	weather_izmir	1.094 ± 0.007075
healthcare_insurance_expenses	4678 ± 18.25	wind	2.831 ± 0.00412
house_16H_reg	$2.92\times10^{4} \pm 86.74$	wine+quality	0.6947 ± 0.001761
house_8L	$2.8\times10^{4} \pm 130.4$	Wine_Quality_red	0.6297 ± 0.005276
house_prices_nominal	$2.34\times10^{4} \pm 536$	Wine_Quality_white	0.6595 ± 0.002086
house_sales_reduced	$1.17\times10^{5} \pm 424.6$	yprop_4_1	$0.02666 \pm 4.83\times10^{-5}$

E Architecture Ablation Details

Exact variant formulations. For a numerical representation geometry $G \in \{\mathbb{H}, \mathbb{S}, E\}$, define

$$\mathbf{v}_{j,k}^{G} = \begin{cases} \psi_{\mathbb{H}}\left(\mathbf{z}_{j,k}^{\mathbb{H}}\right), & G = \mathbb{H}, \\ \psi_{\mathbb{S}}\left(\mathbf{z}_{j,k}^{\mathbb{S}}\right), & G = \mathbb{S}, \\ \mathbf{v}_{j,k}^{E}, & G = E, \end{cases} \tag{24}$$

where $\mathbf{z}_{j,k}^{\mathbb{H}} \in \mathbb{H}_1^d, \mathbf{z}_{j,k}^{\mathbb{S}} \in \mathbb{S}^d, \mathbf{v}_{j,k}^{E} \in \mathbb{R}^d$ are learnable Lorentz, spherical, and Euclidean representations, respectively. Using the affine coefficients from equation 8, the corresponding composition-based numerical representation is

$$\mathbf{h}_j^{\mathrm{n},G}(x_j^{\mathrm{n}}) = \sum_{k=1}^{m_j} \left(a_{j,k} x_j^{\mathrm{n}} + b_{j,k}\right) \mathbf{v}_{j,k}^{G}. \tag{25}$$

All composition-based variants use the same adaptive m_j rule with $m_{\min} = 2$ and $m_{\max} = 12$.

HTNN, HTNN-Sphere, and HTNN-Euc use the matched module pairs $(\mathbb{H}, \mathbb{H})$, $(\mathbb{S}, \mathbb{S})$, and (E, E), respectively, where each pair specifies the categorical and numerical representation geometries. HTNN-CHNE uses $(\mathbb{H}, E)$, whereas HTNN-CENH uses $(E, \mathbb{H})$. HTNN-CHNL and HTNN-CENL replace the numerical composition module with the direct affine map in equation 10, paired with Lorentz and Euclidean categorical representations, respectively.

HTNN-Softmax and HTNN-Sigmoid retain Lorentz representations in both pathways but replace the unconstrained coefficients with

$$\mathbf{h}_j^{\mathrm{n,softmax}}(x_j^{\mathrm{n}}) = \sum_{k=1}^{m_j} \left[\mathrm{softmax}\left(\mathbf{a}_j x_j^{\mathrm{n}} + \mathbf{b}_j\right)\right]_k \mathbf{v}_{j,k}^{\mathbb{H}}, \tag{26}$$

$$\mathbf{h}_j^{\mathrm{n,sigmoid}}(x_j^{\mathrm{n}}) = \sum_{k=1}^{m_j} \left[\mathrm{sigmoid}\left(\mathbf{a}_j x_j^{\mathrm{n}} + \mathbf{b}_j\right)\right]_k \mathbf{v}_{j,k}^{\mathbb{H}}. \tag{27}$$

Lorentz and spherical representations are optimized with RiemannianAdam, while Euclidean representations, composition coefficients, and predictor parameters are optimized with AdamW.

Pairwise comparisons. We follow the score, ranking, and Wilcoxon definitions in Appendix C.3. Ranks are computed within the fixed nine-variant pool, and Holm correction is applied independently over the eight HTNN–variant comparisons for each task. All variants produce valid results on every TALENT-tiny dataset.

Table 7: Pairwise comparisons between HTNN and its controlled variants. W/T/L reports the number of datasets on which HTNN wins, ties, or loses against the listed variant. Holm p denotes the multiplicity-adjusted p-value. All comparisons remain significant at the 0.05 level after correction.

	Classification			Regression		
Variant	W/T/L	Raw p	Holm p	W/T/L	Raw p	Holm p
HTNN-Sphere	28/2/0	1.859×10^{-6}	1.487×10^{-5}	15/0/0	5.897×10^{-4}	4.371×10^{-3}
HTNN-Euc	28/2/0	1.917×10^{-6}	1.487×10^{-5}	15/0/0	5.882×10^{-4}	4.371×10^{-3}
HTNN-CHNE	28/2/0	1.928×10^{-6}	1.487×10^{-5}	15/0/0	6.354×10^{-4}	4.371×10^{-3}
HTNN-CENH	23/4/3	1.976×10^{-5}	1.976×10^{-5}	12/0/3	1.052×10^{-2}	1.052×10^{-2}
HTNN-CHNL	27/2/1	3.262×10^{-6}	1.487×10^{-5}	15/0/0	6.273×10^{-4}	4.371×10^{-3}
HTNN-CENL	28/2/0	1.885×10^{-6}	1.487×10^{-5}	15/0/0	6.241×10^{-4}	4.371×10^{-3}
HTNN-Softmax	28/2/0	1.875×10^{-6}	1.487×10^{-5}	15/0/0	5.464×10^{-4}	4.371×10^{-3}
HTNN-Sigmoid	28/2/0	1.920×10^{-6}	1.487×10^{-5}	15/0/0	5.655×10^{-4}	4.371×10^{-3}

F Controlled Optimization-Dynamics Analysis

Controlled setting. We examine whether the update behavior predicted by the hyperbolic metric appears during practical optimization. We compare HTNN with the matched HTNN-Euc counter-

part defined in Appendix E on all 45 TALENT-tiny datasets, comprising 30 classification and 15 regression tasks. For each dataset and seed, the two models use matched initial tangent-coordinate representations, numerical coefficients, predictor parameters, and batch orders, and are trained for the same prespecified dataset-specific horizon. Each model–dataset configuration is run with three seeds. To make update magnitudes directly comparable, the learning rates of HTNN's manifold representations, its Euclidean parameters, and all HTNN-Euc parameters are set to the same value. All other settings follow the selected dataset-specific HTNN configuration. Because manifold and Euclidean representations are optimized with RiemannianAdam and AdamW, respectively, the trajectory comparison evaluates the realized end-to-end updates of the two training procedures rather than geometry under an identical optimizer, while the JVP analysis separately tests the local metric response.

Measurements. For a tangent-coordinate representation $\mathbf{v} = r\mathbf{q}$, equation 14 predicts an unattenuated radial response and a transverse response of $R_\perp = (r/\sinh r)^2$ under HTNN's curvature of -1. We measure the realized local response using a Jacobian–vector product through the implemented manifold operations. Dataset-level predicted and measured responses use the same transverse-gradient-energy weighting rather than evaluating the response at an average radius.

For the practical trajectories, we record each optimizer update $\Delta\mathbf{v}_{t,i} = \mathbf{v}^+_{t,i} - \mathbf{v}^-_{t,i}$ for each active representation with a non-negligible pre-update norm, and decompose it relative to the pre-update radial direction $\mathbf{q}_{t,i} = \mathbf{v}^-_{t,i}/\|\mathbf{v}^-_{t,i}\|_2$. Its transverse component is

$$\Delta\mathbf{v}_{\perp,t,i} = \left(I_d - \mathbf{q}_{t,i}\mathbf{q}_{t,i}^\top\right)\Delta\mathbf{v}_{t,i}.$$

The transverse-update fraction is

$$F_\perp = \frac{\sum_{t,i}\|\Delta\mathbf{v}_{\perp,t,i}\|_2^2}{\sum_{t,i}\|\Delta\mathbf{v}_{t,i}\|_2^2}, \tag{28}$$

where i indexes feature-value representations with active gradients. Task-level means are computed from dataset-level results averaged over three seeds, and the 95% confidence intervals are obtained from 10,000 bootstrap resamples of datasets.

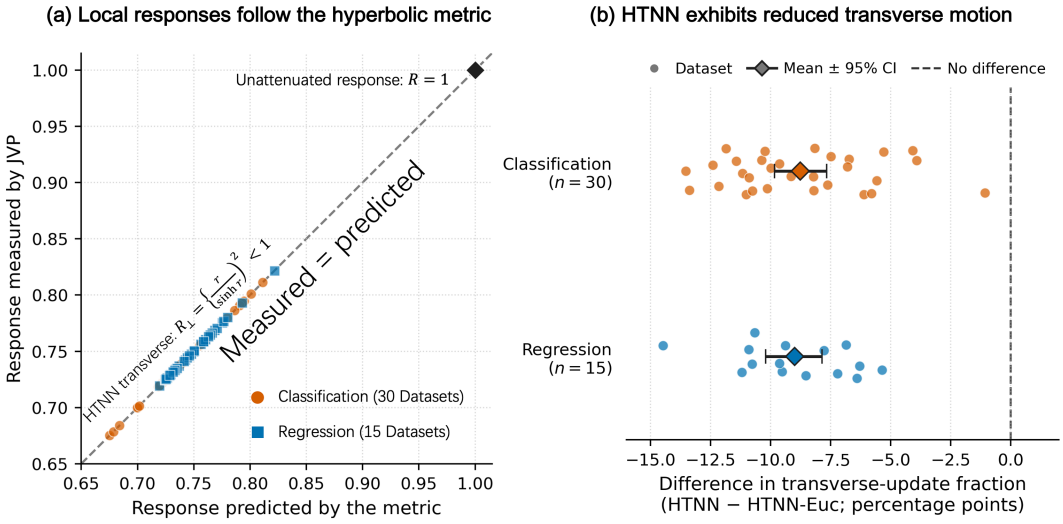

Figure 5: Controlled optimization-dynamics analysis on the 45 TALENT-tiny datasets. Each colored marker reports a dataset-level result averaged over three matched seeds. **(a)** JVP-measured local transverse responses closely follow the predictions of the hyperbolic metric. The black diamond marks the unattenuated response $R = 1$, predicted for HTNN's radial direction and for both directions under Euclidean geometry. **(b)** Difference in transverse-update fraction between HTNN and HTNN-Euc. Negative values indicate less transverse motion under HTNN. Diamonds and horizontal bars denote the task-level mean and its 95% bootstrap confidence interval.

Results. Figure 5(a) shows that the measured transverse responses follow the metric predictions across all datasets, with a maximum absolute error of 7.2×10^{-10}. HTNN's radial response and both HTNN-Euc responses agree with the unattenuated value $R = 1$, whereas HTNN's transverse responses are consistently below one. Figure 5(b) shows that this directional effect persists in practical optimizer trajectories: HTNN has a lower transverse-update fraction than HTNN-Euc on all 30 classification and all 15 regression datasets. The mean difference is -8.77 percentage points for classification, with a 95% confidence interval of $[-9.83, -7.67]$, and -9.00 percentage points for regression, with an interval of $[-10.21, -7.85]$. These observations connect the inverse-metric behavior presented in Section 3.4 and derived in Appendix A.1 to the trajectories observed during training. Together with HTNN's predictive advantage over its matched Euclidean counterpart, they support radial adaptability with transverse stabilization as the optimization behavior induced by HTNN's hyperbolic representation geometry.

G Capacity-Allocation Results

Here, HFV capacity denotes the number of feature-specific HFV representations assigned to each numerical feature: adaptive HTNN computes a feature-specific m_j, whereas each fixed-capacity variant uses the same m for all numerical features.

Figure 6 shows the capacities computed by equation 7. The resulting allocations are highly heterogeneous across benchmark scales and prediction tasks: no single m_j accounts for more than approximately one third of the numerical features, and even the two most frequent capacities together cover fewer than 60%, while a substantial fraction of features receive the minimum capacity $m_j = 2$. Thus, the adaptive rule neither collapses to a nearly fixed capacity nor uniformly enlarges the HFV representation layer. Together with adaptive HTNN's consistent advantage over all fixed-capacity variants, these distributions support feature-wise capacity allocation, rather than greater uniform capacity, as the explanation for the observed improvement.

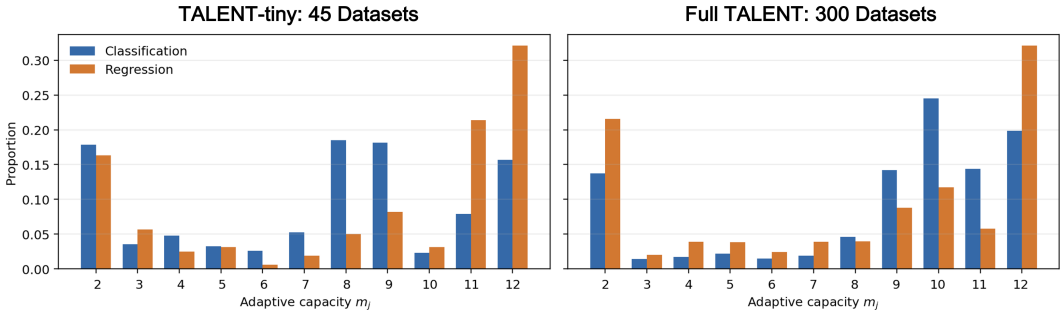

Figure 6: Distribution of the feature-wise HFV capacities m_j computed by equation 7. The left panel covers the 45 TALENT-tiny datasets, and the right panel covers all 300 TALENT datasets. For each task, bar height denotes the proportion of numerical features assigned the corresponding capacity.

Tables 8 and 9 report the complete fixed-capacity comparisons on TALENT-tiny. For each fixed-capacity variant, W/T/L denotes the number of datasets on which adaptive HTNN wins, ties, or loses against that variant. Ranks are computed within the fixed 25-variant pool, and Holm correction is applied independently over the 24 comparisons for each task.

Table 8: Complete capacity-allocation results on the 30 TALENT-tiny classification datasets. Rank entries report the mean and across-dataset standard deviation within the fixed 25-variant pool. Lower is better. Underlining marks the best fixed capacity within each geometry.

	Hyperbolic HTNN				HTNN-Euc			
Capacity	Mean rank	W/T/L	Raw p	Holm p	Mean rank	W/T/L	Raw p	Holm p
Adaptive m_j	$\mathbf{1.83 \pm 3.04}$	—	—	—			—	
$m = 1$	13.82 ± 5.57	27/2/1	2.667×10^{-6}	4.693×10^{-5}	16.18 ± 6.51	28/2/0	1.983×10^{-6}	4.693×10^{-5}
$m = 2$	13.52 ± 5.95	28/2/0	1.972×10^{-6}	4.693×10^{-5}	16.08 ± 5.86	28/2/0	1.995×10^{-6}	4.693×10^{-5}
$m = 3$	12.28 ± 5.95	28/2/0	1.956×10^{-6}	4.693×10^{-5}	14.53 ± 5.34	28/2/0	1.984×10^{-6}	4.693×10^{-5}
$m = 4$	11.93 ± 5.95	28/2/0	1.978×10^{-6}	4.693×10^{-5}	14.63 ± 5.95	28/2/0	1.990×10^{-6}	4.693×10^{-5}
$m = 5$	11.52 ± 6.27	28/2/0	1.988×10^{-6}	4.693×10^{-5}	13.93 ± 6.45	28/2/0	1.999×10^{-6}	4.693×10^{-5}
$m = 6$	12.20 ± 6.01	28/2/0	1.991×10^{-6}	4.693×10^{-5}	13.53 ± 5.21	28/2/0	1.991×10^{-6}	4.693×10^{-5}
$m = 7$	14.22 ± 5.98	28/2/0	1.983×10^{-6}	4.693×10^{-5}	13.45 ± 5.81	28/2/0	1.991×10^{-6}	4.693×10^{-5}
$m = 8$	$\underline{10.42 \pm 6.01}$	28/2/0	1.974×10^{-6}	4.693×10^{-5}	13.20 ± 6.18	28/2/0	1.983×10^{-6}	4.693×10^{-5}
$m = 9$	14.97 ± 7.15	28/2/0	1.989×10^{-6}	4.693×10^{-5}	14.87 ± 5.38	28/2/0	1.989×10^{-6}	4.693×10^{-5}
$m = 10$	12.90 ± 6.89	28/2/0	1.988×10^{-6}	4.693×10^{-5}	12.98 ± 5.06	28/2/0	1.993×10^{-6}	4.693×10^{-5}
$m = 11$	12.65 ± 6.01	28/2/0	1.994×10^{-6}	4.693×10^{-5}	14.32 ± 5.60	28/2/0	1.989×10^{-6}	4.693×10^{-5}
$m = 12$	12.52 ± 6.70	28/2/0	1.986×10^{-6}	4.693×10^{-5}	$\underline{12.52 \pm 5.87}$	28/2/0	1.991×10^{-6}	4.693×10^{-5}

Table 9: Complete capacity-allocation results on the 15 TALENT-tiny regression datasets. Rank entries report the mean and across-dataset standard deviation within the fixed 25-variant pool. Lower is better. Underlining marks the best fixed capacity within each geometry.

	Hyperbolic HTNN				HTNN-Euc			
Capacity	Mean rank	W/T/L	Raw p	Holm p	Mean rank	W/T/L	Raw p	Holm p
Adaptive m_j	$\mathbf{1.00 \pm 0.00}$	—	—	—			—	
$m = 1$	17.13 ± 7.04	15/0/0	6.517×10^{-4}	1.506×10^{-2}	18.93 ± 6.10	15/0/0	6.419×10^{-4}	1.506×10^{-2}
$m = 2$	11.13 ± 5.59	15/0/0	6.467×10^{-4}	1.506×10^{-2}	18.07 ± 6.83	15/0/0	6.402×10^{-4}	1.506×10^{-2}
$m = 3$	10.27 ± 6.28	15/0/0	6.484×10^{-4}	1.506×10^{-2}	15.00 ± 5.82	15/0/0	6.500×10^{-4}	1.506×10^{-2}
$m = 4$	10.47 ± 7.00	15/0/0	6.517×10^{-4}	1.506×10^{-2}	16.33 ± 6.11	15/0/0	6.500×10^{-4}	1.506×10^{-2}
$m = 5$	12.00 ± 7.91	15/0/0	6.451×10^{-4}	1.506×10^{-2}	15.27 ± 7.57	15/0/0	6.451×10^{-4}	1.506×10^{-2}
$m = 6$	9.67 ± 5.18	15/0/0	6.435×10^{-4}	1.506×10^{-2}	17.87 ± 6.42	15/0/0	6.386×10^{-4}	1.506×10^{-2}
$m = 7$	10.80 ± 6.32	15/0/0	6.451×10^{-4}	1.506×10^{-2}	15.40 ± 5.78	15/0/0	6.517×10^{-4}	1.506×10^{-2}
$m = 8$	11.60 ± 7.68	15/0/0	6.484×10^{-4}	1.506×10^{-2}	14.40 ± 6.37	15/0/0	6.435×10^{-4}	1.506×10^{-2}
$m = 9$	$\underline{9.13 \pm 5.41}$	15/0/0	6.435×10^{-4}	1.506×10^{-2}	16.20 ± 5.85	15/0/0	6.273×10^{-4}	1.506×10^{-2}
$m = 10$	12.00 ± 6.92	15/0/0	6.500×10^{-4}	1.506×10^{-2}	14.13 ± 6.24	15/0/0	6.467×10^{-4}	1.506×10^{-2}
$m = 11$	9.20 ± 5.31	15/0/0	6.354×10^{-4}	1.506×10^{-2}	14.13 ± 7.25	15/0/0	6.517×10^{-4}	1.506×10^{-2}
$m = 12$	11.47 ± 7.69	15/0/0	6.500×10^{-4}	1.506×10^{-2}	$\underline{13.40 \pm 6.05}$	15/0/0	6.451×10^{-4}	1.506×10^{-2}

H PER-DATASET EFFICIENCY RESULTS

Tables 10 and 11 report the per-dataset timing results for HTNN and its geometry-free inference form. Fitting times are reported in seconds per 1K training rows, while inference times are reported in seconds per 1K test rows. Each entry gives the mean and standard deviation over 15 runs. Because the geometry-free form is obtained through exact post-training conversion, the fitting-time column corresponds to the original HTNN training process.

For dataset d, inference speedup is computed from the normalized mean times defined in Appendix C.4:

$$s_d = \frac{\bar{t}^{\,\mathrm{inf}}_{\mathrm{HTNN},d}}{\bar{t}^{\,\mathrm{inf}}_{\mathrm{GF},d}}, \tag{29}$$

where GF denotes the geometry-free inference form. Thus, $s_d > 1$ indicates faster inference after conversion. The task-level speedup is the median of s_d over the corresponding datasets.

Table 10: Per-dataset normalized timing results on the 30 TALENT-tiny classification datasets. The speedup column reports original HTNN inference time divided by geometry-free inference time.

Dataset	Fitting time	HTNN inference	Geometry-free inference	Speedup
ada	48.979 ± 4.255	0.943 ± 0.079	0.590 ± 0.069	$1.60\times$
airlines_2000	4.383 ± 1.133	0.330 ± 0.047	0.177 ± 0.073	$1.87\times$
allbp	46.807 ± 17.763	0.582 ± 0.096	0.294 ± 0.013	$1.98\times$
ASP–POTASSCO	453.495 ± 144.814	0.985 ± 0.054	0.608 ± 0.066	$1.62\times$
autoUniv-au7-1100	70.000 ± 46.418	0.314 ± 0.026	0.146 ± 0.012	$2.15\times$
company_bankruptcy_prediction	66.614 ± 41.057	1.901 ± 0.084	1.178 ± 0.100	$1.61\times$
eucalyptus	114.876 ± 38.221	0.286 ± 0.017	0.121 ± 0.011	$2.37\times$
Gender_Gap_in_Spanish_WP	36.192 ± 17.235	0.440 ± 0.103	0.280 ± 0.075	$1.57\times$
hill-valley	657.055 ± 51.084	0.747 ± 0.148	0.471 ± 0.079	$1.59\times$
house_16H	28.802 ± 6.356	0.548 ± 0.098	0.267 ± 0.060	$2.05\times$
ibm-employee-performance	86.889 ± 28.854	0.431 ± 0.063	0.179 ± 0.011	$2.40\times$
INNHotelsGroup	15.809 ± 5.254	0.877 ± 0.100	0.625 ± 0.022	$1.40\times$
internet_firewall	1.271 ± 0.818	0.772 ± 0.103	0.382 ± 0.016	$2.02\times$
jasmine	54.978 ± 34.532	1.717 ± 0.102	1.255 ± 0.171	$1.37\times$
jungle_chess_2pcs_raw_endgame	13.481 ± 1.297	0.514 ± 0.098	0.309 ± 0.024	$1.66\times$
law-school-admission-binary	0.268 ± 0.055	0.476 ± 0.071	0.258 ± 0.018	$1.85\times$
microaggregation2	25.463 ± 7.677	0.607 ± 0.106	0.311 ± 0.015	$1.95\times$
national-longitudinal-survey-binary	19.642 ± 7.996	0.508 ± 0.104	0.228 ± 0.012	$2.23\times$
okcupid_stem	2.920 ± 1.657	1.117 ± 0.585	0.345 ± 0.028	$3.24\times$
online_shoppers	3.395 ± 0.661	0.519 ± 0.103	0.244 ± 0.018	$2.12\times$
ozone_level	0.647 ± 0.539	0.558 ± 0.107	0.413 ± 0.014	$1.35\times$
pc4	131.507 ± 36.235	0.470 ± 0.085	0.252 ± 0.067	$1.86\times$
PhishingWebsites	37.007 ± 9.148	0.771 ± 0.104	0.548 ± 0.024	$1.41\times$
rice_cammeo_and_osmancik	9.203 ± 6.055	0.360 ± 0.068	0.175 ± 0.014	$2.05\times$
shill-bidding	7.787 ± 2.359	0.249 ± 0.012	0.137 ± 0.007	$1.82\times$
Shipping	11.502 ± 5.456	0.543 ± 0.098	0.368 ± 0.015	$1.47\times$
statlog	45.063 ± 30.596	0.330 ± 0.016	0.139 ± 0.009	$2.38\times$
thyroid	26.130 ± 12.894	0.670 ± 0.099	0.299 ± 0.018	$2.24\times$
waveform_v1	15.266 ± 3.867	0.561 ± 0.101	0.423 ± 0.015	$1.33\times$
wine-quality-red	71.199 ± 5.311	0.254 ± 0.016	0.104 ± 0.007	$2.44\times$
			Median per-dataset speedup	$\mathbf{1.87}\times$

Table 11: Per-dataset normalized timing results on the 15 TALENT-tiny regression datasets. The speedup column reports original HTNN inference time divided by geometry-free inference time.

Dataset	Fitting time	HTNN inference	Geometry-free inference	Speedup
2dplanes	16.136 ± 2.076	0.640 ± 0.104	0.448 ± 0.072	$1.43\times$
Abalone_reg	52.300 ± 3.676	0.333 ± 0.013	0.316 ± 0.040	$1.05\times$
analcatdata_supreme	42.386 ± 16.687	0.429 ± 0.090	0.272 ± 0.071	$1.58\times$
BNG(echoMonths)	10.705 ± 1.914	0.451 ± 0.078	0.352 ± 0.068	$1.28\times$
BNG(mv)	7.509 ± 1.095	1.085 ± 0.096	0.688 ± 0.050	$1.58\times$
concrete_compressive_strength	230.254 ± 106.320	0.425 ± 0.323	0.137 ± 0.026	$3.10\times$
cpu_act	51.010 ± 8.633	0.538 ± 0.075	0.293 ± 0.044	$1.84\times$
fifa	13.012 ± 2.706	0.433 ± 0.108	0.189 ± 0.022	$2.29\times$
Goodreads-Computer-Books	44.731 ± 38.870	0.845 ± 0.843	0.131 ± 0.014	$6.46\times$
house_8L	6.947 ± 1.462	0.508 ± 0.099	0.267 ± 0.051	$1.91\times$
IEEE80211aa-GATS	109.513 ± 37.563	0.954 ± 0.697	0.443 ± 0.014	$2.15\times$
Physicochemical_r	16.205 ± 1.484	0.720 ± 0.096	0.421 ± 0.073	$1.71\times$
pole	17.056 ± 5.825	0.740 ± 0.100	0.396 ± 0.071	$1.87\times$
qsar_aquatic_toxicity	183.522 ± 38.294	0.256 ± 0.015	0.122 ± 0.016	$2.09\times$
RSSI_Estimation	37.095 ± 9.084	0.365 ± 0.070	0.223 ± 0.062	$1.63\times$
			Median per-dataset speedup	$\mathbf{1.84}\times$