Embeddings for Numerical Features Using *tanh* Activation

Bingyan Liu
University of Illinois at Urbana-Champaign
bingyan2@illinois.edu

Charles Elkan
University of California, San Diego
celkan@ucsd.edu

Anil N. Hirani
University of Illinois at Urbana-Champaign
hirani@illinois.edu

Abstract

Recent advances in tabular deep learning have demonstrated the importance of embeddings for numerical features, where scalar values are mapped to high-dimensional spaces before being processed by the main model. Here, we propose an embedding method using the hyperbolic tangent (tanh) activation function that allows neural networks to achieve better accuracy on tabular data via an inductive bias similar to that of decision trees. To make training with the new embedding method reliable and efficient, we additionally propose a principled initialization method. Experiments demonstrate that the new approach improves upon or matches accuracy results from previously proposed embedding methods across multiple tabular datasets and model architectures.

1 Introduction

Deep learning has achieved success in various domains, from computer vision to natural language processing. However, its application to tabular data has been challenging, with gradient-boosted decision trees (GBDTs) typically outperforming neural networks. This has led researchers to investigate how neural networks can better capture the inductive bias that makes tree-based models effective on tabular data.

Work by Gorishniy et al. (2022) has demonstrated that proper embedding of numerical features is beneficial for achieving performance competitive with that of GBDTs. Recent developments have introduced additional approaches to tabular embeddings: Li et al. (2024) use tree ensembles to transform numerical variables into binarized embeddings, while Wu et al. (2024) suggest a two-step feature expansion and deep transformation technique.

We propose here an approach to numerical feature embeddings based on properties of the hyperbolic tangent (tanh) function. The tanh function exhibits a dual nature that aligns well with the structure of tabular data: with large weight w, $\tanh(wx + b)$ captures a tree-like inductive bias by creating a sharp decision boundary, while with small w it approximates a linear function; see Figure 1.

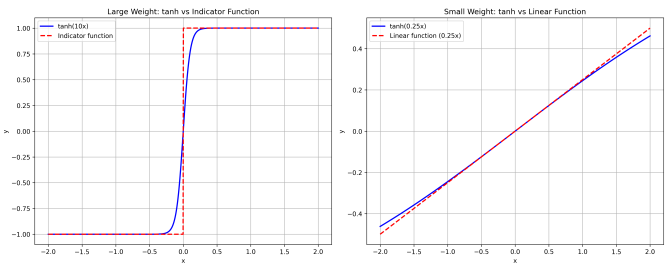

Figure 1: With a large weight w, $\tanh(wx)$ approximates an indicator function, enabling tree-like decision boundaries, while with small w, it allows a smooth feature transformation.

However, with poor initialization, neural network training using tanh can lead to vanishing gradients and unstable learning. To overcome this, we introduce an initialization method based on a simple probability argument. The new method ensures that the embedding parameters w and b start in a region that facilitates both tree-like and linear representations. Empirically, the new initialization method does achieve the desired benefits of more stable training and better accuracy.

Experiments demonstrate the effectiveness of our approach in two scenarios. In the first scenario, we compare embeddings using fixed dimensions, where the model hyperparameters are tuned without considering the embedding layer. In this case, the new tanh-based approach consistently outperforms previous embedding methods across various datasets and model architectures. In the second scenario, we compare against previous ReLU embeddings, where both the model parameters and the embedding dimensions were tuned for the use of

ReLU. Even in this challenging comparison, tanh-based embeddings lead to accuracy improvements. Overall, the new approach can achieve competitive or superior performance with minimal tuning overhead, making it particularly practical for scenarios where extensive hyperparameter search is not feasible.

2 Related Work

The application of deep learning to tabular data has historically been challenging, with GBDTs often achieving superior performance (Ke et al., 2017). Recent studies provide insights into this performance gap: Grinsztajn et al. (2022) demonstrated that tree-based models' success stems from their inherent ability to learn effective decision boundaries and handle heterogeneous features, while McElfresh et al. (2023) identified specific data characteristics where neural networks can potentially outperform GBDTs.

Traditional neural networks treat numerical features as direct inputs without specialized processing. This approach has limitations in capturing complex feature interactions and non-linear relationships. Recent work by Gorishniy et al. (2022) has studied simple differentiable embeddings, which apply a linear transformation followed by an activation function, and piecewise linear embeddings, which creates disjoint learnable bins for feature values. Their experiments demonstrated that these embeddings can significantly improve neural network performance on tabular data.

More recently, Li et al. (2024) proposed a tree-regularized method that uses tree ensembles to transform numerical variables into binarized embeddings, and Wu et al. (2024) introduced a unified framework employing lightweight neural networks for both numerical and categorical features, utilizing two-step feature expansion and transformation. Importantly, neither of these methods is a standard single neural network that can be trained by backpropagation in a standard way, whereas the method that we suggest below can be.

Recent research has also made progress in closing the performance gap with GBDTs through other innovations in feature processing and model architecture. Transformer-based models such as TabTransformer (Huang et al., 2020) and FT-Transformer (Gorishniy et al., 2021) tokenize the features, using attention mechanisms to capture complex feature interactions. Hybrid approaches such as NODE (Popov et al., 2020) incorporate tree-like structures into neural architectures, while DCN V2 (Wang et al., 2021) uses cross networks to model feature interactions. However, these methods are also not a simple single neural network that is trainable in a standard way.

2.1 Activation functions and initialization

ReLU (Nair and Hinton, 2010) has become the default choice for activation function due primarily to its ability to mitigate the vanishing gradient problem. However, it has limitations, including that neurons can become inactive during training. Alternatives proposed to address these limitations include Leaky ReLU (Maas et al., 2013) and Parametric ReLU (He et al., 2015) which introduce a small negative slope, while ELU (Clevert et al., 2016) and GELU (Hendrycks and Gimpel, 2016) offer smoother gradients.

The hyperbolic tangent (tanh) function, although less commonly used in modern architectures, has useful properties. Its bounded output between -1 and 1 provides natural normalization, while its sigmoidal shape enables both smooth transformations and sharp transitions that are similar to decision boundaries in trees.

Proper initialization is crucial for training stability and convergence, particularly in the context of tabular data where feature scales and distributions can vary significantly. Glorot and Bengio (2010) introduced Xavier initialization, scaling weights based on layer dimensions to maintain variance. He et al. (2015) extended this for ReLU activations, accounting for the activation's non-linearity. For tanh activations, LeCun et al. (2012) proposed scaling weights by the square root of fan-in to maintain variance.

While these approaches provide solid foundations for neural network training, adapting them for tabular data embeddings presents challenges due to varying feature distributions and the need to balance linear and non-linear representations. Recent data-dependent methods such as LSUV initialization (Mishkin and Matas, 2016) are adaptive, but can be computationally intensive.

3 The tanh-based embedding method

Given a tabular dataset with numerical features, our goal is to develop an embedding method that can capture both linear and non-linear relationships in the data. Let $\mathbf{x} \in \mathbb{R}^p$ represent a numerical feature vector, where p is the number of features. The new embedding method maps each feature x_i to $\mathbb{R}^d$, where d is the embedding dimension.

Previous approaches using ReLU activation functions in the embedding layer can be expressed as $\mathbf{e}_i = \text{ReLU}(\mathbf{W}_i x_i + \mathbf{b}_i)$ where $\mathbf{W}_i \in \mathbb{R}^{d\times 1}$ and $\mathbf{b}_i \in \mathbb{R}^d$ are learnable parameters and $\mathbf{e}_i \in \mathbb{R}^d$ is the embedding of x_i (Gorishniy et al., 2022).

We propose replacing the ReLU activation with tanh, so $\mathbf{e}_i = \tanh(\mathbf{W}_i x_i + \mathbf{b}_i)$. The advantage is that with large embedding weights $W_{i,j} \gg 1$, each component of the embedding captures a tree-like inductive bias, by creating a sharp decision boundary. Conversely, with a small weight $W_{i,j} \ll 1$, a component approximates a linear transformation, because $\tanh(x) \approx x$ for small x. See Figure 1.

We also propose an enhanced embedding variant with a second transformation layer:

$$\mathbf{e}_i = \sigma(\mathbf{M}_i \tanh(\mathbf{W}_i x_i + \mathbf{b}_i) + \mathbf{c}_i)$$

where $\mathbf{M}_i \in \mathbb{R}^{d\times d}$ and $\mathbf{c}_i \in \mathbb{R}^d$ are additional learnable parameters, and σ is another activation function, possibly ReLU. This two-layer method allows for more complex feature transformations while keeping the benefits of the tanh approach.

3.1 Connection to decision trees

A decision tree can be expressed as a function as follows. First, each node in the tree is an indicator function of some feature. Next, a path from the root to a leaf node, which represents a sequence of decisions, is a product of these indicator functions or their negations along the path. Finally, the entire tree is a combination of decision path functions:

$$f(\mathbf{x}) = \sum_{p\in P} c_p \prod_{i\in p} \mathrm{D}_i(x_i, \theta_i)$$

where P is the set of all paths from root to leaves, c_p is the constant value assigned to the leaf node at the end of path p, and $\mathrm{D}_i(x_i, \theta_i)$ is either $\mathbb{1}_{x_i\ge\theta_i}$ or $\mathbb{1}_{x_i<\theta_i}$ for feature x_i with threshold θ_i, where the choice depends on the split direction and which half-domain the node represents. As a simple example, see Figure 2.

Each component of a tanh embedding can approximate a smoothed version of an indicator function as follows. Consider component j of the vector $\mathbf{e}_i$, as the weight $W_{i,j}$ approaches infinity. Given $b_{i,j} = -\theta_{i,j} W_{i,j}$ for a fixed $\theta_{i,j}$, the tanh function approaches an indicator function:

$$\lim_{W_{i,j}\to\infty} \tanh(W_{i,j} x_i + b_{i,j}) = \mathbb{1}_{x_i\ge\theta_{i,j}}.$$

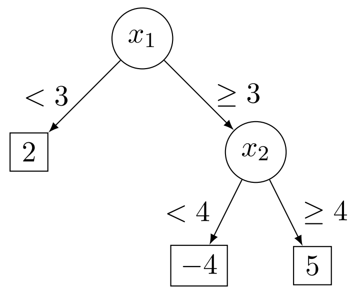

Figure 2: A example of decision tree of depth 2 operating on features x_1 and x_2, showing both branches at the root but only expanding the right subtree ($x_1 \ge 3$).

Each component j can capture a different decision boundary, allowing the model to learn a rich set of decision rules while maintaining differentiability, which is crucial for gradient-based optimization.

The piecewise linear embedding method in Gorishniy et al. (2022) partitions each feature's range into bins with predefined boundaries. In contrast, the tanh-based approach allows the model to adapt the location and the sensitivity of bin boundaries.

3.2 Principled initialization

Proper initialization of embedding weights is important for the success of tanh-based embeddings. We propose a method that takes advantage of the properties of tanh. We first preprocess all numerical features using min-max scaling to the range $[-1, 1]$. Our method uses the fact that $\tanh(t)$ behaves approximately linearly for t in $[-0.5, 0.5]$.

When initializing an embedding that maps a feature x_i into d dimensions, we aim to create uniformly distributed bins across the $[-1, 1]$ range, with each bin having a length of $2/d$. We initialize the embedding parameters as

$$W_{i,j} = d/2 \text{ and } b_{i,j} \sim \text{Uniform}(-d/2, d/2).$$

As training progresses, the model learns to adjust the weights and biases to capture both linear relationships (when $W_{i,j}x_i + b_{i,j}$ is within $[-1, 1]$) and non-linear relationships (otherwise).

In the appendix, we prove that this initialization strategy ensures that for any input value $x \in [-1, 1]$, the expected number of bins where preactivation $wx + b$ falls within $[-0.5, 0.5]$ is 1.

The effectiveness of the proposed initialization strategy is empirically validated through analysis of learned weight distributions in Section 5.3, which shows that the embeddings maintain good coverage of the feature space while adapting to local feature complexity.

Table 1: Dataset properties. MSE (↓: lower is better) denotes mean-square error, and AUC (↑: higher is better) denotes area under the ROC curve. Dataset abbreviations: GE (gesture), CH (churn), CA (california), HO (house), AD (adult), OT (otto), HI (higgs-small), FB (fb-comments), SA (santander), CO (covtype).

	Regression			Classification						
	CA	FB	HO	AD	CH	CO	GE	HI	OT	SA
#objects	20640	197080	22784	48842	10000	581012	9873	98049	61878	200000
#num. features	8	50	16	6	10	54	32	28	93	200
#cat. features	0	1	0	8	1	0	0	0	0	0
metric	MSE↓	MSE↓	MSE↓	AUC↑	AUC↑	AUC↑	AUC↑	AUC↑	AUC↑	AUC↑
#classes	–	–	–	2	2	7	5	2	9	2
majority class	–	–	–	76%	79%	48%	29%	52%	26%	89%

Table 2: Naming scheme for model variants. Each variant name consists of a prefix (experiment scenario), embedding variants, and initialization suffix. For example, 2B-LT-a is a model using optimized piecewise linear embedding parameters (2B-), with tanh-based embedding (T), and optimized initialization (-a).

(Abbr.)	(Embedding Variants)
Base model	MLP, ResNet, FT-Transformer
FR	control group with ReLU activation
FT	control group with Tanh activation
LR	embedding with ReLU activation
LT	embedding with Tanh activation
LRLR	LR + linear layer + ReLU
LTLR	LT + linear layer + ReLU
(Abbr. suffix)	(Initialization)
-s	standard initialization
-a	principled initialization
(Abbr. prefix)	(Scenarios)
(#)-	preassigned embedding dim (#)
2A-	see the main text
2B-	see the main text

4 Design of Experiments

The embedding dimension is a hyperparameter that has to be chosen to balance model expressiveness with computational efficiency. We conduct experiments under two scenarios that differ in how base model parameters (e.g., hidden dimensions of MLP) and embedding dimensions are selected.

Scenario 1 - Preassigned Dimensions: In this scenario, we adopt the optimized base model parameters (hidden dimensions and dropout rates for MLP, number of blocks etc. for ResNet and Transformer) obtained from hyperparameter search without considering embeddings, and use preassigned embedding dimensions. This allows us to evaluate the impact of replacing ReLU with tanh activations while keeping all architectural choices fixed.

Scenario 2 - ReLU-Optimized Dimensions: Here, we adopt both the base model parameters and embedding dimensions that are obtained from hyperparameter search for ReLU embeddings variants. This scenario is split into two cases. Scenario 2A uses tuned hyperparameter for linear embedding, and Scenario 2B uses tuned hyperparameter for piecewise linear embedding, as reported in Gorishniy et al. (2022). This creates a challenging comparison where we replace ReLU with tanh in settings optimized for ReLU, demonstrating the robustness of our approach.

Importantly, we do not perform additional hyperparameter search for model parameters or embedding dimensions for the tanh-based approach. This evaluation strategy demonstrates that the benefits of our method are inherent rather than the result of hyperparameter search, making it a drop-in replacement for ReLU-based embeddings in applications where extensive tuning may be infeasible.

As baselines for comparison, we consider two control groups that, before feeding the input data to the base model, process it through an extra single fully-connected layer for all features, with either ReLU or tanh activation functions. Thus $\mathbf{e} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{W} \in \mathbb{R}^{d \times p}$, $\mathbf{b} \in \mathbb{R}^{d}$ and σ is either ReLU or tanh. These control groups separate the impact of our feature-wise embedding approach from the increase in dimensionality by comparing against a baseline fully-connected layer.

We evaluate three base model architectures, following the implementations in Gorishniy et al. (2022).

MLP: A multi-layer perceptron with multiple hidden layers.

ResNet: A residual network adapted for tabular data, incorporating skip connections to facilitate training of deeper architectures.

FT-Transformer: A Feature Tokenizer Transformer architecture that treats tabular features as a sequence, enabling feature interactions through the attention mechanism.

For each base architecture, we evaluate several vari-

Table 3: Performance of MLP variants on multiple datasets in Scenario 1. All rows are variations of MLP.

	MSE↓			AUC↑						
MLP-variants	CA	FB	HO	AD	CH	CO	GE	HI	OT	SA
MLP	.0682	.0115	**.0483**	.8972	.8568	.9953	.7954	.7392	.9541	.8575
30-FR-s	.0575	.0115	**.0483**	.9027	.8544	.9957	.7932	.7296	.9608	.8520
30-FT-s	.0578	.0117	.0486	.9029	.8590	.9925	.7947	.7584	.9632	.8549
30-LR-s	.0577	.0108	.0495	.9101	**.8640**	.9925	.7746	.5554	.9663	.8927
30-LT-s	.0591	.0108	.0508	.9085	.8615	.9956	.7827	.5000	**.9686**	.8601
30-LT-a	**.0497**	.0098	.0526	.9110	.8490	.9839	.8052	.7619	.9589	.8926
30-LRLR-s	.0584	.0099	.0500	.9096	.8611	.9962	.6181	.5000	.9678	.8932
30-LTLR-s	.0566	.0100	.0493	.9097	.8574	.9960	.7955	.6681	.9658	.8958
30-LTLR-a	.0525	**.0096**	.0492	**.9125**	.8538	**.9969**	**.8099**	**.7990**	.9664	**.8958**

Table 4: Performance of ResNet variants on multiple datasets in Scenario 1. All rows are variations of ResNet.

	MSE↓			AUC↑						
ResNet variants	CA	FB	HO	AD	CH	CO	GE	HI	OT	SA
ResNet	.0662	.0107	.0420	.9106	.8610	.9978	**.8273**	.8129	.9695	.8658
30-FR-s	.0598	.0103	.0447	.9093	.8546	.9976	.8153	.7940	.9689	.8713
30-FT-s	.0633	.0107	.0426	.9058	.8630	.9975	.8162	.7907	.9694	.8609
30-LR-s	.0752	.0092	**.0416**	.9119	.8623	.9980	.8092	.8160	**.9698**	.8786
30-LT-s	.0648	.0093	.0420	.9120	.8627	.9977	.8152	.8175	.9696	.8628
30-LT-a	.0463	.0095	.0506	.9117	.8484	.9974	.8180	.7997	.9638	.8894
30-LRLR-s	.0699	.0090	.0463	.9111	.8644	.9978	.7239	**.8178**	**.9698**	.8819
30-LTLR-s	.0610	.0091	.0419	.9121	**.8659**	.9978	.7520	**.8179**	.9696	.8699
30-LTLR-a	**.0456**	**.0087**	.0516	**.9158**	.8622	**.9983**	.8146	.8146	.9656	**.8901**

ants to assess the impact of different embedding approaches. The base model is the architecture without any specialized embedding layer, serving as our primary baseline. The variants are:

- **ReLU-based**: Simple differentiable embeddings with ReLU activation
- **Tanh-based**: Our approach using tanh activation
- **Enhanced**: Additional linear transformation layer after the activation function for both ReLU and tanh variants
- **Control**: A fully-connected layer with specified activation function. (For the FT-transformer, we do not test control group variants as they are not applicable.)

Standard initialization follows Kaiming for ReLU-based models and Xavier for tanh-based models. For the FT-transformer, we use the initialization method from Gorishniy et al. (2022). “Principled" refers to our initialization method described above. Names for the model variants and hyperparameter settings are in Table 2.

4.1 Datasets and metrics

We evaluate our approach on the nine tabular datasets used in Gorishniy et al. (2022), which represent a range of real-world scenarios with varying mixtures of numerical and categorical features, both regression and classification problems, and sizes. For categorical features, we employ label encoding without additional preprocessing in the MLP and ResNet models, and tokenization in the FT-Transformer model. Table 1 provides statistics for each dataset, including the number of numerical and categorical features, sample sizes, and task types.

For evaluation of model performance, we employ task-specific metrics as follows.

Classification: We use the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) as our primary metric. AUC-ROC provides a measure of classification performance that is independent of the chosen decision threshold and handles class imbalance. For multi-class classification tasks, we report the average one-vs-rest AUC-ROC across all classes.

Regression: We evaluate using Mean Squared Error (MSE), which measures the average squared difference between predicted and actual values. Lower MSE values indicate better accuracy.

Training Efficiency: We record the training time for each initialization method to compare convergence speed and training efficiency.

Table 5: Performance of MLP variants on datasets in Scenario 2A and 2B.

	MSE$\downarrow$			AUC$\uparrow$						
MLP variants	CA	FB	HO	AD	CH	CO	GE	HI	OT	SA
MLP	.0682	.0115	.0483	.8972	.8568	.9953	.7954	.7392	.9541	.8575
2A-LR-s	.0530	.0104	.0715	.8923	.8546	.9666	.7833	.5494	.9704	.8940
2A-LT-s	.0530	.0106	.0566	.8888	.8521	.9783	.7798	.6638	.8988	.8618
2A-LT-a	.0496	.0099	.0442	.8717	.8495	.9201	.8274	.7955	.9066	.8955
2A-LRLR-s	.0575	.0093	.0651	.9105	.8582	.9692	.7897	.7990	.8970	.8770
2A-LTLR-s	.0574	.0101	.0444	.9091	.8520	.9896	.8414	.7885	.6885	.8966
2A-LTLR-a	**.0470**	**.0073**	**.0360**	**.9174**	.8520	.9963	**.8545**	.8013	.9649	.9028
2B-LR-s	.0832	.0104	.0509	.9020	.8572	.9965	.8058	.7377	.9652	.8946
2B-LT-s	.0858	.0111	.0513	.8971	.8502	**.9979**	.8033	.6883	.9639	.8790
2B-LT-a	.0528	.0092	.0464	.8856	.8482	**.9979**	.8121	.7881	.9611	.8946
2B-LRLR-s	.0511	.0106	.0457	.9088	.8598	.9914	.7987	.7874	.9553	.8909
2B-LTLR-s	.0524	.0100	.0466	.9095	**.8614**	.9946	.7998	.7999	.9644	.8906
2B-LTLR-a	.0510	.0087	.0434	.9143	.8488	.9915	.8181	**.8112**	**.9737**	**.9054**

Table 6: Performance of ResNet variants on multiple datasets in Scenario 2A and 2B.

	MSE$\downarrow$			AUC$\uparrow$						
ResNet variants	CA	FB	HO	AD	CH	CO	GE	HI	OT	SA
ResNet	.0662	.0107	.0420	.9106	.8610	.9978	.8273	.8129	.9695	.8658
2A-LR-s	.0725	**.0082**	**.0365**	**.9176**	**.8644**	.9909	.8228	**.8237**	.9712	.8895
2A-LT-s	.0704	.0096	.0428	.9119	.8607	.9949	**.8815**	.8045	**.9736**	.8827
2A-LT-a	**.0427**	.0089	.0481	**.9170**	.8516	**.9985**	.8619	.7941	**.9736**	**.9077**
2B-LR-s	.0601	.0096	.0449	.9124	.8617	.9972	.7971	.8182	.9685	.8928
2B-LT-s	.0671	.0094	.0428	.9146	.8602	.9955	.8065	.8198	.9684	.8849
2B-LT-a	.0477	.0089	.0459	.9135	.8492	.9970	.8093	.7897	.9643	.8919

We employ 5-fold cross-validation. We split the data into five shares, and in each fold, pick one share as test set and split the rest as training and validation set. We report the mean metrics across all five folds, and we consider one result to be better than another if its mean score is better and its standard deviation is less than the difference between the best and the second best result. Unless otherwise specified, we use hyperparameters that were tuned on 80% of the original dataset by Gorishniy et al. (2022).

We adapt the training framework from TabSurvey (Borisov et al., 2024) and begin with the model implementations from Gorishniy et al. (2021). All models are implemented in PyTorch and trained using the Adam optimizer with hypertuned learning rate, batch size of 128, and at most 300 epochs with early stopping based on validation performance. All experiments are conducted on a single Nvidia A100 GPU. Our code is available at `https://github.com/liu-bingyan/numbed`.

5 Results and Analysis

In Tables 3 to 7 MSE ($\downarrow$: lower is better) denotes mean-square error and AUC ($\uparrow$: higher is better) denotes area under the ROC curve. The best results for each dataset are shown in **bold**. Multiple bold entries in the same column indicate results that are statistically equivalent. Table 2 explains the model variant abbreviations used in the results.

5.1 Scenario 1: Preassigned Dimensionality

We first evaluate the effectiveness of our method in Scenario 1 as defined above.

For MLP models (Table 3), the tanh-based embedding exhibits better performance compared to the ReLU-based embedding across almost all test cases. Moreover, our initialization method shows notable improvements in the enhanced variants.

For ResNet models (Table 4), we observe consistent improvements similar to those observed in the MLP architecture. The tanh-based enhanced embedding demonstrates superior performance compared to the ReLU-based embedding across the majority of test cases. The new initialization significantly improves performance for the CA and SA datasets.

Overall, the results from Scenario 1 demonstrate that given a preassigned embedding dimension, the new tanh-based method effectively outperforms the ReLU-based embedding, particularly in the enhanced variants.

Table 7: Performance evaluation of FT-Transformer variants on multiple datasets in Scenario 2A and 2B. All rows are variations of FT-Transformer.

	MSE↓			AUC↑							
FT-Transformer variants	CA	FB	HO	AD	CH	CO	GE	HI	OT	SA	
2A-LR-s	.0555	.0098	.0604	.9205	**.8690**	**.9982**	**.8865**	**.8093**	.9672	**.8987**	
2A-LT-s	.0519	.0099	.0502	.9194	**.8689**	**.9981**	.8501	.8069	.9657	.8939	
2A-LT-a	**.0396**	**.0090**	.0476	.9224	.8562	.9967	.8446	.8015	.9665	.8954	
2B-LR-s	.0679	.0098	.0478	.9158	.8619	**.9981**	.8026	.8074	**.9677**	**.8988**	
2B-LT-s	.0685	.0098	.0466	.9144	.8650	.9975	.8470	.8082	.9661	.8943	
2B-LT-a	.0443	**.0091**	**.0313**	**.9259**	.8623	.9968	.8718	.8040	.9655	.8949	

5.2 Scenario 2: ReLU-Optimized Dimensions

Models with carefully tuned hyperparameters constitute Scenario 2. While the improvements are more modest, they remain consistent across architectures.

For MLP (Table 5), tanh-based enhanced embeddings demonstrate superior performance compared to ReLU-based embeddings across all test cases in both Scenarios 2A and 2B. This suggests that our method effectively combines the advantages of both linear embeddings and piecewise linear embeddings. Notably, this performance advantage holds even though the hyperparameters are tuned for the comparison model, demonstrating the generality and robustness of our approach.

For ResNet (Table 6), while the original ReLU-based embedding shows competitive performance, our tanh-based enhanced embedding maintains better performance in more than half of the test cases. This demonstrates that our method achieves comparable or better performance than hyperparameter-tuned models. Additionally, our initialization method improves performance in more than half of the test cases and does not significantly degrade performance in the remaining cases.

For FT-Transformer (Table 7), our method shows significant improvement in some datasets, reducing MSE to .0396 on dataset CA, while it stays competitive in other cases.

5.3 Further Analyses

Figure 3 visualizes the learned embeddings for the first feature from the California Housing dataset. Compared to standard initialization, principled initialization allows bins to concentrate in regions where the conditional expectation of the label changes rapidly with the feature.

A significant advantage of our method lies in its computational efficiency, as demonstrated in Table 8. The average number of epochs required for convergence is consistently improved for MLP models, with many cases converging twice as fast compared to the ReLU-based embedding.

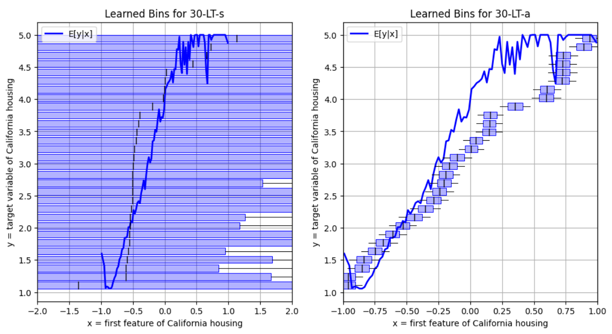

Figure 3: Comparison of embedding spaces learned with different initializations for the first feature x of the California Housing dataset (left standard, right principled). The center of each box is the embedding center $c = -b/w$ in the expression $w(x-c) = wx+b$, and the width is $1/w$. The horizontal line represents width $2/w$. Boxes are sorted by their centers; the vertical position of each box is for display only and carries no meaning.

Table 8: Number of epochs required for convergence across different MLP variants in Scenario 1. Lower values indicate faster convergence. The best values in each group are bolded.

MLP variants	AD	CA	CH	CO	FB	GE	HI	HO	OT	SA
MLP	62	94	38	100	113	31	54	63	89	20
30-FR	76	84	27	101	104	33	40	65	79	18
30-FT	57	99	28	55	74	29	74	62	47	18
30-LR-s	56	117	85	207	105	78	44	106	216	93
30-LT-s	51	89	79	124	116	109	32	78	240	41
30-LT-a	**38**	**74**	**22**	**46**	**72**	**61**	**30**	**35**	**94**	**22**
30-LRLR-s	50	90	67	129	126	41	**23**	141	90	87
30-LTLR-s	45	97	50	122	104	41	50	106	74	99
30-LTLR-a	**34**	**51**	**26**	**74**	**56**	**28**	63	**46**	**33**	**26**

6 Discussion

In summary, the experimental results above show that:

- In Scenario 1 (preassigned dimensions), the new method achieves better accuracy than the

ReLU-based method, particularly in enhanced variants.

- In Scenario 2 (ReLU-optimized dimensions), the new method maintains competitive performance against hyperparameter-tuned models, suggesting it captures a useful inductive bias.

In both scenarios, our initialization's performance varies for different models, but it doesn't degrade performance and improves it in half of the cases. Moreover, the new method improves computational efficiency, reducing training time while maintaining or improving model accuracy.

Overall, tanh-based embeddings appear to constitute a practical and effective solution for using numerical features in tabular deep learning, offering both accuracy improvements and computational benefits, without the need for extensive hyperparameter tuning.

Limitations and Future Work

While our method demonstrates promising results across multiple architectures and datasets, there are several directions for future exploration.

Our current evaluation is based on the benchmark datasets from Gorishniy et al. (2022). Future work could extend this evaluation to more recent benchmarks, such as those proposed in Gorishniy et al. (2024a) and Holzmüller et al. (2024), to further validate the effectiveness of our approach.

In terms of model architectures, we have demonstrated the effectiveness of our method on the models presented in Gorishniy et al. (2022), which includes MLP, ResNet, and Transformer architectures. Future work could explore the integration of our method with more recent architectures, such as TabR (Gorishniy et al., 2024b), RealMLP (Holzmüller et al., 2024) and others.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable feedback and suggestions. The work of BL and ANH was supported in part by the University of Illinois Urbana-Champaign Campus Research Board Award RB24134. This work used NCSA Delta at National Center for Supercomputing Applications through allocation CIS230374 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

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

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. *arXiv preprint arXiv:2203.05556*.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2024a. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. 2024b. TabR: Tabular deep learning meets nearest neighbors. In *International Conference on Learning Representations*.

Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35, pages 507–520.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

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

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. TabTransformer: Tabular data modeling using contextual embeddings. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 627–635.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30:3146–3154.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Xuan Li, Yun Wang, and Bo Li. 2024. Tree-regularized tabular embeddings. *arXiv preprint arXiv:2403.00963*. Table Representation Learning Workshop at NeurIPS 2023.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1–8.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. 2023. When do neural nets outperform boosted trees on tabular data? In *Advances in Neural Information Processing Systems*, volume 36, pages 76336–76369.

Dmytro Mishkin and Jiri Matas. 2016. All you need is a good init. *International Conference on Learning Representations*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814.

Sergei Popov, Stanislav Morozov, and Artem Babenko. 2020. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2021. Deep & cross network for ad click predictions. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1–9.

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

A Analysis of Initialization Strategy

This appendix analyzes the proposed initialization strategy for the embedding parameters. Consider a numerical feature x normalized to the interval $[-1, 1]$ and its embedding $\tanh(wx + b)$. For optimal learning of decision boundaries, the pre-activation values $wx + b$ should lie predominantly within $[-0.5, 0.5]$, where the tanh function exhibits linear behavior.

Lemma 1. Let $\{c_i\}_{i=1}^{d}$ be independent and identically distributed random variables following a uniform distribution on $[-1, 1]$. For each i, let $I_i = [c_i - r, c_i + r]$ be the closed interval of radius r centered at c_i. Then, for any fixed point $x \in [-1, 1]$, the expected number of intervals containing x is more than $dr/2$.

Proof. For any fixed $x \in [-1, 1]$ and each interval I_i, we have

$$\begin{aligned}\mathbb{P}(x \in I_i) &= \mathbb{P}(c_i - r \leq x \leq c_i + r)\\ &= \begin{cases} r & \text{if } x \in [-1 + r, 1 - r],\\ \frac{r+1-|x|}{2} & \text{if } |x| > 1 - r, \end{cases}\end{aligned}$$

Thus the $\mathbb{P}(x \in I_i) \geq r/2$. By the linearity of expectation, the expected number of intervals containing x is

$$\mathbb{E}\left[\sum_{i=1}^{d} \mathbb{1}_{x \in I_i}\right] = \sum_{i=1}^{d} \mathbb{P}(x \in I_i) \geq dr/2$$

□

The proposed initialization strategy sets $w = d/2$, so $r = 1/d$, because $-0.5 < wx + b < 0.5$ is equivalent to $-0.5/w < (x + b/w) < 0.5/w$ and $r = 0.5/w = 1/d$. Therefore the expected number of bins where the tanh activation provides meaningful gradients for learning, for any given data point, is at least 1.

This property ensures effective gradient propagation during training while keeping the bins relatively small for discriminative learning. The theoretical justification for this choice stems from the trade-off between gradient propagation and feature discrimination: a higher coverage probability would lead to excessive activation and reduced discriminative capacity, while a lower probability would risk insufficient gradient flow during training.