On the Efficiency of NLP-Inspired Methods for Tabular Deep Learning

Anton Frederik Thielmann [*] and **Soheila Samiee**[2]

[1]BASF SE, Germany
[2]BASF Canada Inc., Canada
{firstname.lastname}@basf.com

Abstract

Recent advancements in tabular deep learning (DL) have led to substantial performance improvements, surpassing the capabilities of traditional models. With the adoption of techniques from natural language processing (NLP), such as language model-based approaches, DL models for tabular data have also grown in complexity and size. Although tabular datasets do not typically pose scalability issues, the escalating size of these models has raised efficiency concerns. Despite its importance, efficiency has been relatively underexplored in tabular DL research. This paper critically examines the latest innovations in tabular DL, with a dual focus on performance and computational efficiency. The source code is available at `https://github.com/basf/mamba-tabular`.

1 Introduction

Deep Neural Networks have emerged as a powerful tool for a wide range of tasks, particularly those involving unstructured data. In areas such as image classification (Maurício et al., 2023), video processing (Apostolidis et al., 2021), point cloud analysis (Zhang et al., 2023), protein structure prediction (Baek et al., 2021; Jumper et al., 2021), or natural language understanding (e.g. (Brown, 2020)), the best-performing models today are all based on neural networks.

Particularly for textual problems, larger models consistently outperform their smaller counterparts (Muennighoff et al., 2022). Therefore, efficiency is of critical importance, not only for inference but also for training. As many models, especially large language models, rely on transformer architectures, the quadratic time complexity of attention mechanisms with respect to sequence length can be especially challenging for longer sequences (Beltagy et al., 2020).

Recently, the introduction of the Mamba architecture (Gu and Dao, 2023), based on state-space models (SSMs) (Gu et al., 2021), has gained significant attention due to its promise of a linear increase in memory consumption compared to the quadratic increase observed in transformer architectures. Mamba is designed to be both memory-efficient and high-performing, as demonstrated by the variety of extensions built upon it. For instance, the MoE (JAMBA) architecture (Lieber et al., 2024; Pióro et al., 2024), Mamba for point clouds (Liu et al., 2024; Zhang et al., 2024), for time series (Ahamed and Cheng, 2024; Liang et al., 2024), and for graphs (Behrouz and Hashemi, 2024) all show the versatility and efficiency of Mamba-based models.

Tabular data tasks, on the other hand, remain one of the last frontiers not yet fully dominated by deep neural networks. Gradient Boosted Decision Trees (GBDTs) still lead the way for tasks involving tabular data (McElfresh et al., 2024; Grinsztajn et al., 2022). However, tabular deep learning has started to close the performance gap. Advances from natural language processing have shown

---

[*]Correspondence to `anton.thielmann@basf.com`

promise for tabular tasks. Models such as TabTransformer (Huang et al., 2020), FT-Transformer (Gorishniy et al., 2021) and Mambular (Thielmann et al., 2024) have proven to be competitive models, occasionally outperforming GBDTs. More recently, Prior-Fitted Networks (PFNs) have introduced a new and promising direction for tabular models (Hollmann et al., 2022). Despite this, models like TabPFN (Hollmann et al., 2022) are not yet scalable for larger datasets.

Since tabular tasks rarely involve massive datasets, efficiency has traditionally not been a significant concern. However, to truly close the gap between GBDTs and tabular deep learning models, both, memory-efficiency and speed will play a crucial role, since GBDTs are not only performant but also highly efficient compared to tabular deep learning models (Gu et al., 2021; Nori et al., 2019). While architectures like transformers and Mamba, originally designed for NLP, have been adapted for tabular tasks, a thorough analysis of their efficiency for tabular data is still missing. This paper addresses that gap by evaluating and comparing the performance and efficiency of several tabular deep learning models, complementing existing benchmark studies that focus solely on performance (e.g., (McElfresh et al., 2024; Gorishniy et al., 2021; Grinsztajn et al., 2022; Borisov et al., 2022)).

In summary, the contributions of this paper are as follows:

I. We evaluate LLM-inspired deep learning methods for tabular data, focusing on their efficiency and examining how effectively efficient NLP techniques can be applied to tabular tasks

II. We demonstrate the effectiveness of recurrent neural networks (RNNs) and show that simple sequential architectures can be powerful tools for solving tabular problems.

2 Methodology

We follow the experimental design from Thielmann et al. (2024).The exact same pre-processing, data splits and evaluation is used. The notation is aligned with Thielmann et al. (2024) and Gorishniy et al. (2021) such that $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ is the training dataset of size n and y denotes the target variable that can be arbitrarily distributed. Each input $\boldsymbol{x} = (x_1, x_2, \ldots, x_J)$ contains J features (variables). The categorical and numerical features are denoted as $\boldsymbol{x} \equiv (\boldsymbol{x}_{cat}, \boldsymbol{x}_{num})$. For each feature, embedding friendly periodic linear encodings as presented by Gorishniy et al. (2022) are used.

$$z_{j(\text{num})}^{t} = \begin{cases} 0 & \text{if } x < b_{t-1}, \\ 1 & \text{if } x \geq b_t, \\ \frac{x - b_{t-1}}{b_t - b_{t-1}} & \text{else.} \end{cases} \tag{1}$$

The tested models can be grouped into Transformer-based models, such as FT-Transformer (Gorishniy et al., 2021); Recurrent models, including Mambular (Thielmann et al., 2024) and a classical RNN; hybrid architectures, similar to the Jamba architecture (Lieber et al., 2024); and MLP and ResNet architectures.

For all models except the MLP and ResNet, the features are fed through embedding layers with identical dimensions for all datasets. For categorical features, traditional embedding layers are used after integer encoding. For numerical features, simple linear layers with the same dimensions as the embedding layers are employed. After the embedding layer, in the transformer and recurrent models, the feature matrix has the shape $\tilde{\mathbf{x}} \in \mathbb{R}^{N \times J \times d}$, where N is the batch size, J is the number of variables, and d is the embedding dimension.

This paper introduces two new architectures for tabular deep learning that are described in the following:

TabulaRNN TabulaRNN is taking advantage of a RNN architecture. Instead of token embeddings, the feature embeddings, similar to those in Mambular (Thielmann et al., 2024) are passed to the RNN. The RNN model processes an input sequence $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$, where $\mathbf{x}_t$ represents the input at time step t. At each time step, the hidden state $\mathbf{h}_t$ is updated based on the previous hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$. In this context, the *sequence* the RNN iterates over corresponds to the features of the tabular data, where each feature represents a step in the sequence. The hidden

state is computed using the following equation:

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \tilde{\mathbf{x}}_t + \mathbf{b}), \tag{2}$$

where $\mathbf{W}_h$ describes recurrent weight matrix applied to the previous hidden state and $\mathbf{W}_x$ is the input weight matrix applied to the current input. $\mathbf{b}$ describes a simple bias term, and σ is an activation function.

The final output representation, which is passed to a tabular MLP head, is computed by averaging over the hidden states, i.e., over the hidden state for each variable. Since this corresponds to each sequence having a fixed length, missing values can be handled by introducing special tokens, using imputation methods such as median imputation, or simply dropping them. All model configuration details can be found here: `https://github.com/basf/mamba-tabular`.

MambAttention MambAttention is a combination of Mambular (Thielmann et al., 2024) and the FT-Transformer (Gorishniy et al., 2021). It loosely follows the modeling architecture from Lieber et al. (2024). The input sequence is embedded just like in the RNN. The first layer is a Mamba Block, with the given matrices:

$$\mathbf{A} \in \mathbb{R}^{1 \times 1 \times d \times \delta}, \quad \mathbf{B} \in \mathbb{R}^{N \times J \times 1 \times \delta}, \quad \Delta \in \mathbb{R}^{N \times J \times d \times 1}, \bar{\mathbf{z}} \in \mathbb{R}^{N \times J \times d \times 1},$$

where δ denotes a inner dimension, $\bar{\mathbf{z}}$ has the same entries as $\tilde{\mathbf{x}}$, but one additional axis. The hidden states, $\mathbf{h}_j \in \mathbb{R}^{N \times d \times \delta}$, are updated as follows:

$$\mathbf{h}_j = \exp\left(\Delta \odot_3 \mathbf{A}\right)_{:,j,:,:} \odot_{1,2,3} \mathbf{h}_{j-1} + \left((\Delta \odot_{1,2} \mathbf{B}) \odot_{1,2,3} \bar{\mathbf{z}}\right)_{:,j,:,:}. \tag{3}$$

After the first Mamba block follows a standard multihead-attention layer. The sequences state is simply passed from the Mamba block to the attention block. These blocks are designed in alternating fashion with the first and the last block always being Mamba blocks. The final representation is achieved by averaging over the hidden states. This representation is passed to a final MLP task head.

Mambular-Triton For evaluation of efficiency of Mambular model, in addition to a purely pytorch implemented Mambular, a Triton based Mambular version[2] – closely following the original Mamba implementation (Gu and Dao, 2023) and (Thielmann et al., 2024) – is tested. This did not have any impact on performance, so the result of its implementation is only reported for effeciency. In the result section, the pure pytorch implementation is denoted as *Mambular* and the version following the Mamba1 implementation is denoted as *mambular-Triton*.

3 Results

3.1 Performance

The performance of the models are benchmarked using 12 open-source data sets across two tasks (regression and classification). Details of datasets are provided in the supplementary materials A. All experimental results were obtained using 5-fold cross-validation. The average results and standard deviations across the 5 folds are reported. All models utilize PLE encodings (Eq. 1). The results are reported in Table 1. Overall results confirm that sequence-based deep learning models perform very well for tabular tasks. Even the simple TabulaRNN shows strong performance, comparable to both Mambular and FT-Transformer. Alternating attention layers and Mamba blocks does not lead to any performance improvement. Overall, Mambular is the best performing model, closely followed by the FT-Transformer and TabulaRNN.

3.2 Efficiency

The memory-efficiency and computation time of all six models have been benchmarked in a more controlled set up. Each model is designed with approximately 350k trainable parameters to ensure comparability in terms of size. All models are analyzed on the same Nvidia-T4 GPU. Simulated data, with an equal number of numerical and categorical features is used for all efficiency results.

---

[2]For all Mambular-Triton experiments, Mamba1 from the mamba-ssm package, version 2.2.2 is used.

Table 1: Benchmarking results for the regression and classification tasks. Average mean squared error values and average AUC values over 5 folds and the corresponding standard deviations are reported, respectively. Smaller values are better for regression and larger values are better in classification tasks (marked with arrows). The best performing model is marked in bold.

Models	Regression Tasks							Classification Tasks					Avg. Rank
	DI ↓	AB ↓	CA ↓	WI ↓	PA ↓	HS ↓	CP ↓	BA ↑	AD ↑	CH ↑	FI ↑	MA ↑	
FT-Transformer*	0.018	0.458	0.169	**0.615**	**0.024**	**0.111**	**0.024**	0.926	0.926	**0.863**	0.792	0.916	2.25
	± 0.001	± 0.055	± 0.006	± 0.012	± 0.005	± 0.014	± 0.001	± 0.003	± 0.002	± 0.007	± 0.011	± 0.003	
MLP*	0.066	0.462	0.198	0.654	0.764	0.147	0.031	0.895	0.914	0.840	0.793	0.886	5.58
	± 0.003	± 0.051	± 0.011	± 0.013	± 0.023	± 0.017	± 0.001	± 0.004	± 0.002	± 0.005	± 0.011	± 0.003	
ResNet*	0.039	0.455	0.178	0.606	0.606	0.141	0.030	0.896	0.841	0.841	0.793	0.889	4.25
	± 0.018	± 0.045	± 0.006	± 0.013	± 0.031	± 0.017	± 0.002	± 0.006	± 0.002	± 0.006	± 0.013	± 0.003	
Mambular*	**0.018**	**0.452**	**0.167**	0.628	0.035	0.132	0.027	0.927	**0.928**	0.856	0.795	0.917	**2.00**
	± 0.000	± 0.043	± 0.011	± 0.010	± 0.005	± 0.020	± 0.002	± 0.006	± 0.002	± 0.004	± 0.011	± 0.003	
MambAttention	**0.018**	0.484	0.189	0.638	0.030	0.142	0.026	0.919	0.921	0.857	0.781	0.911	3.67
	± 0.000	± 0.052	± 0.006	± 0.003	± 0.006	± 0.024	± 0.002	± 0.004	± 0.002	± 0.004	± 0.009	± 0.001	
TabulaRNN	**0.018**	0.459	0.178	0.659	0.073	0.114	0.027	**0.930**	0.925	0.855	**0.796**	**0.922**	2.75
	± 0.000	± 0.047	± 0.013	± 0.013	± 0.012	± 0.014	± 0.001	± 0.004	± 0.002	± 0.006	± 0.011	± 0.002	

* adopted from Thielmann et al. (2024)

Where applicable, the models use an embedding size of 64 which is equal to each numerical feature dimension corresponding to 64 bins in PLE encodings. Categorical features were limited to 10 categories. The results are obtained with a simulated mini batch size of 32 for simulations with less than 100 features and 8 for simulations with more than 100 features.

In terms of memory usage, both ResNet and MLP are significantly superior compared to other models, exhibiting the lowest GPU consumption across different feature sets. The GPU consumption against number of features is visualized in Figure 1. TabulaRNN, while less performant than Mambular and FT-Transformer, stands out for its high efficiency, showing linear GPU usage as the number of features increases. In contrast, FT-Transformer's GPU consumption grows quadratically, making it less efficient for larger feature sets. However, for problems with more than 25 features, the Triton version of Mambular becomes more efficient than the attention-based FT-Transformer. On the other hand, in the pure PyTorch version, FT-Transformer remains more efficient than Mambular for problems with fewer than 350 features in our specific configuration. Given, that tabular problems most often do not entail such large numbers of features – the maximum number of features in Table 1 is 32 – Triton Mambular implementation or the transformer based models would be efficient alternatives compared to the recurrent pytorch Mambular model. MambAttention, on the other hand, proved to be neither efficient nor particularly performant, falling behind the other models in both respects.
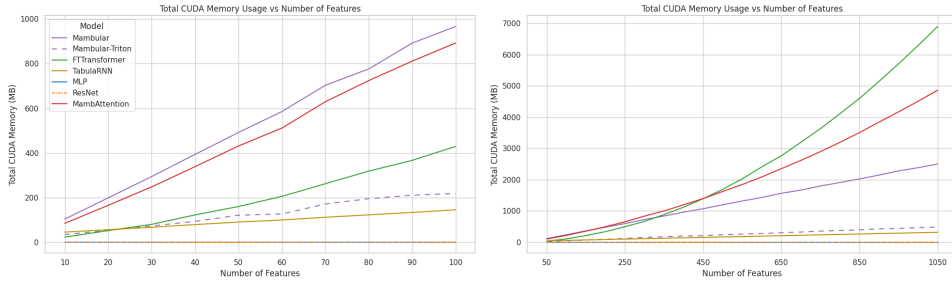

Figure 1: Memory consumption as a function of the number of features. Left shows GPU memory usage for datasets with <100 features/variables. MambAttention and Mambular show the highest memory usage, increasing linearly with the number of features. Mambular-Triton's GPU memory consumption is significantly lower than base Mambular. (Right) For a large number of features, FT-Transformer's memory usage increases quadratically, approaching Mambular and MambAttention at around 400 features.

The assessment of the effect of embedding size on CUDA memory usage (Figure 2) revealed that Mamba-based architectures' efficiency decreases at a more rapid pace than FT-Transformer and TabulaRNN with the increase in embedding size. Figures 3 presents a comprehensive view of performance rank, CUDA memory usage, and computation time. It summarizes the results from real-world datasets (Table 1) and compares model efficiency with a fixed 20-feature configuration. While Mambular-Triton demonstrates an advantageous balance of low memory usage and high performance, it falls short in comparison to other evaluated alternatives when it comes to computation
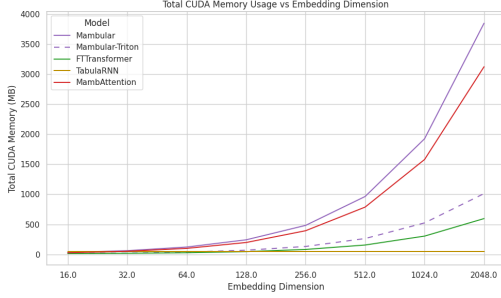
Figure 2: Memory consumption as a function of embedding dimension for a fixed set of 12 features. A pure pytorch Mambular's memory consumption increases significantly faster than that of the FT-Transformer.
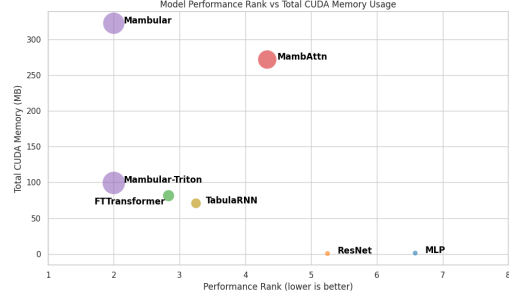
Figure 3: Average rank performance vs GPU memory usage. Circle size represents the total computation time for a forward pass on a batch of 32 with 10 numerical and 10 categorical features and an embedding size of 64.

time. Notably, both FT-Transformer and TabulaRNN remain highly performant while being significantly more efficient than Mambular.

As illustrated in Figure 4, When analyzing training memory consumption and time, particularly during backward passes, the results closely mirror the efficiency observed during inference. Both the pure PyTorch version of Mambular and the MambAttention model exhibit linear increases in memory consumption and time taken with respect to the number of features. While the Triton version of Mambular is more efficient than the FT-Transformer for fewer than 25 features, it is slower across all tested setups. The TabulaRNN is not only more memory-efficient than the FT-Transformer, but also as fast, if not faster for larger datasets with more than 50 features.
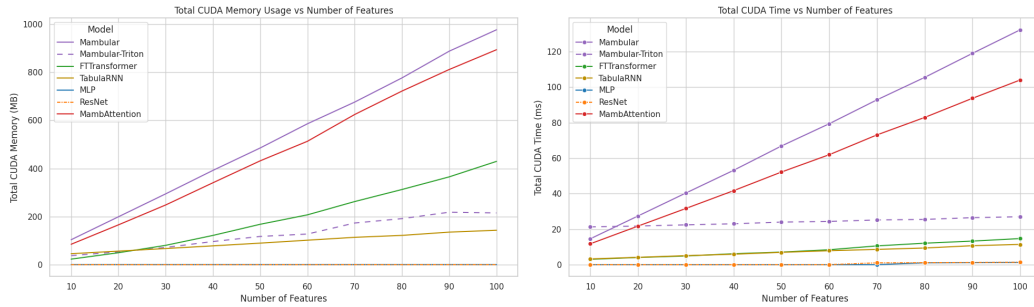

Figure 4: Memory consumption (left panel) and GPU time taken (right panel) during a backward pass as a function of the number of features.

4 Conclusion

With the introduction of TabulaRNN and MambAttention this study confirms that sequential models over features dimension are performant alternatives for tabular DL. It examines the transferability of efficient NLP architectures to the tabular domain, revealing that while the Mamba architecture is highly effective for text, its efficiency does not translate as well to tabular tasks, specially for the common cases in tabular domain with less than fifty features. Although the Mambular-Triton implementation achieves efficiency on par with FT-Transformer even for small datasets, it is slightly slower in training and inference time. However, the introduced TabulaRNN not only performs on par with Mambular and the attention based alternative FT-Transformer, it can outperform them in terms of efficiency and computation time. Given these results and the findings from Thielmann et al. (2024) regarding sequence ordering, combinations of both architectures, similar to Lieber et al. (2024) could be promising alternatives for tabular tasks, inheriting the performance and efficiency of TabulaRNN and the positional invariance of Transformers.

Acknowledgments and Disclosure of Funding

References

Ahamed, M. A. and Cheng, Q. (2024). Timemachine: A time series is worth 4 mambas for long-term forecasting. *arXiv preprint arXiv:2403.09898*.

Apostolidis, E., Adamantidou, E., Metsai, A. I., Mezaris, V., and Patras, I. (2021). Video summarization using deep neural networks: A survey. *Proceedings of the IEEE*, 109(11):1838–1863.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., et al. (2021). Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876.

Behrouz, A. and Hashemi, F. (2024). Graph mamba: Towards learning on graphs with state space models. *arXiv preprint arXiv:2402.08678*.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*.

Brown, T. B. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Gorishniy, Y., Rubachev, I., and Babenko, A. (2022). On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

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

Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Gu, A., Goel, K., and Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2022). Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. (2020). Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.

Liang, A., Jiang, X., Sun, Y., and Lu, C. (2024). Bi-mamba4ts: Bidirectional mamba for time series forecasting. *arXiv preprint arXiv:2404.15772*.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., et al. (2024). Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Liu, J., Yu, R., Wang, Y., Zheng, Y., Deng, T., Ye, W., and Wang, H. (2024). Point mamba: A novel point cloud backbone based on state space model with octree-based ordering strategy. *arXiv preprint arXiv:2403.06467*.

Maurício, J., Domingues, I., and Bernardino, J. (2023). Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Applied Sciences*, 13(9):5521.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. (2024). When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36.

Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2022). Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Pióro, M., Ciebiera, K., Król, K., Ludziejewski, J., and Jaszczur, S. (2024). Moe-mamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081*.

Thielmann, A. F., Kumar, M., Weisser, C., Reuter, A., Säfken, B., and Samiee, S. (2024). Mambular: A sequential model for tabular deep learning. *arXiv preprint arXiv:2408.06291*.

Zhang, H., Wang, C., Tian, S., Lu, B., Zhang, L., Ning, X., and Bai, X. (2023). Deep learning-based 3d point cloud classification: A systematic survey and outlook. *Displays*, 79:102456.

Zhang, T., Li, X., Yuan, H., Ji, S., and Yan, S. (2024). Point could mamba: Point cloud learning via state space model. *arXiv preprint arXiv:2403.00762*.

A Datasets

All used datasets are taken from the UCI Machine Learning repository and publicly available[3]. We drop out all missing values. For the regression tasks the targets are normalized in standard format. Note, that before PLE encoding the numerical features are scaled to be within (-1, +1) range.

Table 2: The used datasets for benchmarking. All datasets are taken from the UCI Machine Learning repository. #num and #cat represent the number of numerical and categorical features, respectively. The number of features thus determines the "sequence length" for Mambular and MambAttention models. The train, test and val numbers represent the average number of samples in the respective split for the 5-fold cross validation. Ratio marks the percentage of the dominant class for the binary classification tasks.

Name	Abbr.	#cat	#num	train	test	val	ratio
			Regression Datasets				
Diamonds	DI	4	7	34522	10788	8630	-
Abalone	AB	1	8	2673	835	668	-
California Housing	CA	1	9	13210	4128	3302	-
Wine Quality	WI	0	12	4158	1299	1039	-
Parkinsons	PA	2	20	3760	1175	940	-
House Sales	HS	8	19	13832	4322	3458	-
CPU small	CPU	0	13	5243	1638	1310	-
			Classification Datasets				
Bank	BA	13	8	28935	9042	7233	88.3%
Adult	AD	9	6	31259	9768	7814	76.1%
Churn	CH	3	9	6400	2000	1600	79.6%
FICO	FI	0	32	6694	2091	1673	53.3%
Marketing	MA	15	8	27644	8638	6910	88.4%

---

[3]Available at https://archive.ics.uci.edu/datasets