Exploration of autoregressive models for in-context learning on tabular data

Stefan K. Baur
SAP SE
s.baur@sap.com

Sohyeong Kim
SAP SE
sohyeong.kim@sap.com

Abstract

We explore different auto-regressive model architectures for in-context learning on tabular datasets trained in a similar manner to TabPFN [Hollmann et al., 2022]. Namely, we compare transformer based models with a structured state-space model architecture (Mamba) and a hybrid architecture (Jamba), mixing transformer and Mamba layers. We find that auto-regressive transformer models perform similarly to the original TabPFN transformer architectures, albeit at the cost of a doubled context length. Mamba performs worse than similar sized transformer models, while hybrid models show promise in harnessing some advantages of state-space models such as supporting long input context length and fast inference.

1 Introduction

Foundation models for image and language domains have had significant impact in recent years. This has not been the case for foundation models for the tabular data domain and recently there has been considerable effort to challenge the status quo [Van Breugel and Van Der Schaar, 2024]. Various directions of pre-training have been explored in the context of tabular data, either by pre-training on large real world datasets, hoping to exploit semantic similarity between tables [Kim et al., 2024, Gardner et al., 2024], or by pre-training on synthetic datasets combined with in-context learning (Prior-Data Fitted Networks or PFNs) [Hollmann et al., 2022, Müller et al., 2022]. The focus of this work is to explore various autoregressive model architectures for in-context learning on tabular data, using the same training and evaluation pipeline as was used for TabPFN [Hollmann et al., 2022] and MotherNet [Müller et al., 2023]. Recently, architectures based on structured state-space models, such as Mamba, have been proposed as an alternative to transformer models [Gu and Dao, 2024] and provide competitive performance across several domains involving sequential data (e.g. natural language, audio or DNA data). It has also been suggested that Mamba has similar in-context learning capabilities as transformer models [Grazzi et al., 2024] [Park et al., 2024].

Real world datasets found in enterprise applications can have many features (of which typically many are categorical), many target classes and training data sets can be much larger than 2k samples (typically 10k-100k samples), exceeding some of the limitations of TabPFN[1]. The issue of limited context size, which ultimately traces back to the quadratic complexity of self-attention in transformers [Rabe and Staats, 2022], can be addressed to some degree by a retrieval augmented generation (RAG) like approach, selecting a local context of K nearest neighbors of the sample X_{query} according to some distance function [Thomas et al., 2024] or by selecting an optimized context that summarizes the training data combined with feature dimensionality reduction [Feuer et al., 2023]. Our motivation is to compare the recent structured state-space model (SSM) architecture Mamba [Gu and Dao, 2024] with transformer based models [Vaswani et al., 2017], to explore whether these could be a viable alternative. The advantage of SSMs is that the context, i.e. the training dataset, can be processed

[1]In its original version, TabPFN was limited to a few thousand samples, 10 target classes and 100 numerical features [Hollmann et al., 2022].

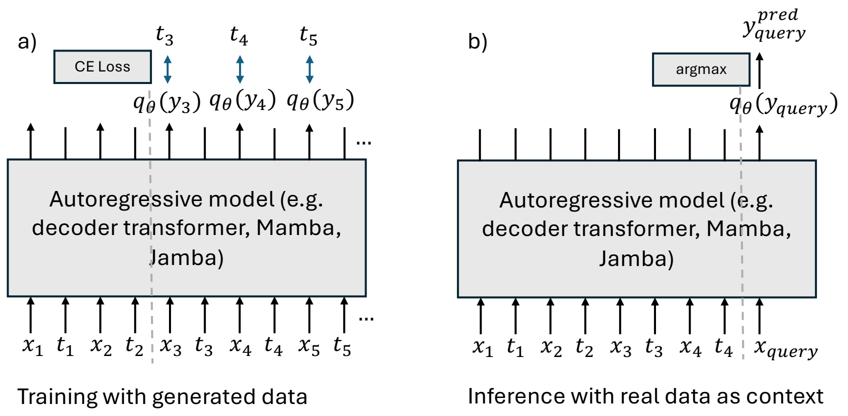

a)
CE Loss
t_3 t_4 t_5
$q_\theta(y_3)$ $q_\theta(y_4)$ $q_\theta(y_5)$
Autoregressive model (e.g. decoder transformer, Mamba, Jamba)
x_1 t_1 x_2 t_2 x_3 t_3 x_4 t_4 x_5 t_5
Training with generated data
b)
argmax
y_{query}^{pred}
$q_\theta(y_{query})$
Autoregressive model (e.g. decoder transformer, Mamba, Jamba)
x_1 t_1 x_2 t_2 x_3 t_3 x_4 t_4 x_{query}
Inference with real data as context

Figure 1: Auto-regressive models for in-context learning. a) For training, we use the same pipeline as TabPFN and MotherNet with the difference that we do not use a single split into training and test data, but instead due to the causal nature we can use the entire dataset for training (except for an initial context that enables the model to observe samples from all target classes). b) For inference, the training set is fed into the model as context and we predict on a single query item at a time. (Adapted from Fig. 1 of [Garg et al., 2023])

in a complexity that scales linearly in the sequence length instead of the quadratic complexity of transformers. Furthermore, once the context has been processed, inference on a single query item can be done in constant time and does not require storing a cache of size proportional to the sequence length as is required for auto-regressive transformer models [Wang et al., 2024]. This would pave the way to scaling Prior-Data Fitted Networks to larger tabular datasets [Grazzi et al., 2024].

2 Approach

In order to explore the feasibility of using auto-regressive models for in-context learning on tabular data, we start by establishing a baseline using a standard decoder transformer model. Instead of using the GPT-2 architecture as [Garg et al., 2023], we choose GPT-NeoX [Black et al., 2022] as backbone as this architecture uses rotary positional embeddings [Su et al., 2023] which is beneficial over trained positional embeddings for sequences exceeding the sequence length used for training. While this approach by itself does not provide many benefits over the original TabPFN architecture, as the context length is doubled which is particularly disadvantageous for the transformer architecture due to the quadratic complexity of self-attention, it serves as a prerequisite and benchmark for testing other auto-regressive model architectures. We compare this baseline with a state-space model architecture (Mamba [Gu and Dao, 2024]) and a hybrid architecture (Jamba [Lieber et al., 2024]) that mixes transformer and Mamba layers using the implementation available from HuggingFace [Wolf et al., 2020]. For these two models, no positional encodings are required.

Our training pipeline is the same as the one used for TabPFN and MotherNet with the model architecture replaced with the respective auto-regressive model, with the mapping of inputs and outputs similar to what was done by the authors of [Garg et al., 2023].

The training data is sampled from the same prior distribution as [Hollmann et al., 2022] [Müller et al., 2023] and consists of N samples feature-target pairs $(x_i, t_i)_{i=1}^{N}$ with D features and T target classes. As in [Hollmann et al., 2022], we limit ourselves to a maximum of $T = 10$ target classes and $D = 100$ features, and use $N = 1152$ feature-target pairs for each training set sample. While this synthetic training dataset is explicitly split into training and test data as a consequence of the TabPFN architecture [Hollmann et al., 2022], we can use almost all feature-target pairs of each dataset for training as in the causal nature of the auto-regressive models each output is conditioned only on the previous inputs.

As shown in Fig. 1, the auto-regressive model maps a sequence of feature-target pairs $(x_i, t_i)_{i=1}^{k-1}$ and a query item x_k to a probability distribution over predicted target classes $y_k \in \{1, \ldots, T\}$ $q_\theta\left(y_k | (x_i, t_i)_{i=1}^{i=k-1}, x_k\right)$ parametrized by the model weights θ.

Each feature vector x_i is embedded into the input space of the auto-regressive backbone architecture using a linear layer and the target class t_i is encoded using a one-hot linear layer (as in [Müller et al., 2023]). The output embeddings of the backbone model are then passed into a classification head to produce q_θ if the input was an embedded feature. If the input was an embedded target value, the output is ignored.

We train the model using cross-entropy loss $\mathcal{L}$ summed over all samples in the training dataset, excluding the first $k_0 = 50$ samples to have high probability that all 10 possible target classes have appeared at least once in the initial context

$$\mathcal{L} \propto - \sum_{k=k_0+1}^{N} \sum_{c=1}^{T} t_{k,c} \log q_\theta \left(y_k = c | (x_i, t_i)_{i=1}^{k-1}, x_k \right), \tag{1}$$

where $t_{k,c}$ is the one-hot encoded target class vector.

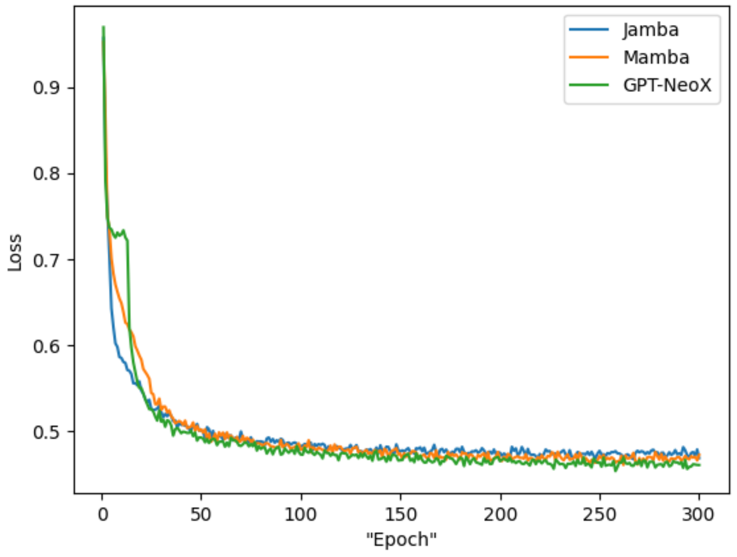


Figure 2: Training loss curves for the different auto-regressive models (each with parameter count around 5M). Each "epoch" corresponds to 8192 steps with a batch size of 8, corresponding to ~ 65.5k synthetic datasets (~ 19.7M datasets for 300 "Epochs").

2.1 Training

We trained all different models (TabPFN, GPT-NeoX, Mamba and Jamba) with the same optimizer and learning rate schedule and the same training data pipeline similar to [Hollmann et al., 2022] [Müller et al., 2023] [2]. We use the AdamW optimizer with a maximum learning rate of 3×10^{-5}, a warmup period of 20 epochs and cosine decay of the learning rate to zero over 300 epochs (where each epoch corresponds to 8192 steps with batch size 8, which is later doubled via gradient accumulation after 20, 50 and 200 epochs [Müller et al., 2023]). The loss curves (showing training loss, but as the dataset is continuously sampled from the prior, this can be viewed as an evaluation on unseen data) for the different models are shown in Fig. 2. While the loss declines smoothly for the Mamba and Jamba models (also for TabPFN; not shown) we always observed a step-like descend behavior for the GPT-NeoX model, which could potentially be mitigated by curriculum learning as discussed in

[2]We build our pipeline on the coding released by A. Müller on `https://github.com/microsoft/mothernet` which in turn builds upon the coding released by the authors of [Hollmann et al., 2022] `https://github.com/automl/TabPFN`.

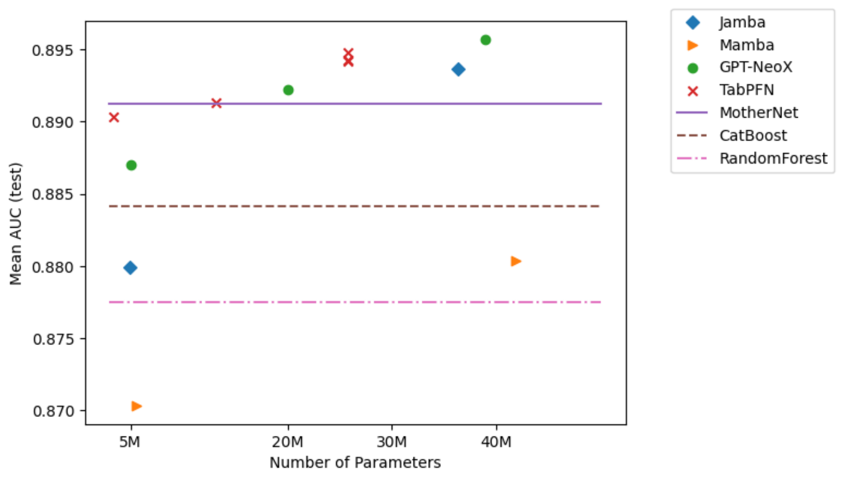


Figure 3: Comparison of different model architectures on the test set of 30 datasets from the OpenML CC-18 dataset (also evaluated on in [Hollmann et al., 2022] [Müller et al., 2023]). Note: We evaluated several TabPFN models with 25.8M parameters (trained by us, and versions released by the authors of [Hollmann et al., 2022] and [Müller et al., 2023]. All achieved similar scores.)

[Garg et al., 2023]. We trained all auto-regressive models with the same learning rate schedule for 300 epochs on a single GPU (on either a A10G or A100 GPU with 24GB/80G of memory).

2.2 Inference

Inference in the autoregressive setting is performed as shown in Fig. 1b. We feed the entire training dataset as context into the model and then we predict for the query (test) items x_k the distribution of confidences over target classes $q_\theta(y_k)$. To avoid reprocessing the context for each query (test) item we want to predict on, we have used the caching functionality of the HuggingFace transformers library [Wolf et al., 2020].

3 Results

3.1 Evaluation on OpenML CC-18

We used the same datasets and evaluation protocol as in [Müller et al., 2023]. We have evaluated the models on the validation set of 149 OpenML datasets and the test set of 30 datasets of the OpenML CC-18 dataset [Bischl et al., 2021] as selected by [Hollmann et al., 2022] and [Müller et al., 2023]. The metric used is ROC AUC (one-vs-one for multiclass) averaged over 5 random splits (50/50 split between training and evaluation samples for each OpenML dataset). The datasets were originally selected to match the constraints of TabPFN in terms of dataset size, number of features and targets. For each model we used ensembling (with 32 realizations as in [Hollmann et al., 2022]) and report the mean ROC AUC score over the validation and test datasets in Table 1 and for the test datasets only in Fig. 3. In addition to the permutations used for ensembling in [Hollmann et al., 2022], we have also permuted the ordering of training set (in-context) samples for the auto-regressive models. Overall, we observe that the transformer based models (GPT-NeoX and TabPFN) perform similarly, while the Mamba model performs worse than the transformer based models. The hybrid model Jamba shows improved performance over Mamba, but still performs slightly worse than the transformer based models of similar model size.

3.2 Top 10 datasets with largest performance difference

We evaluated each model on the test and validation splits of each dataset and ordered the datasets from left to right in descending order based on the difference in model performance. This difference

Model	Validation	Test	Parameters (M)	N_L	E
TabPFN (ours)	0.833	0.890	3.3	6	256
TabPFN (ours)	0.834	0.891	13.2	6	512
TabPFN (ours)	0.836	0.894	25.8	12	512
TabPFN [Hollmann et al., 2022]	0.834	0.894	25.8	12	512
TabPFN [Müller et al., 2023]	0.838	0.895	25.8	12	512
GPT-NeoX	0.828	0.887	5.0	6	256
GPT-NeoX	0.833	0.892	18.9	6	512
GPT-NeoX	0.838	0.896	39.0	12	512
Mamba	0.814	0.870	5.6	12	256
Mamba	0.827	0.880	41.8	24	512
Jamba (1 transf. layer)	0.822	0.880	4.9	4	256
Jamba (2 transf. layers)	0.838	0.894	36.4	10	512
MotherNet [Müller, 2024]	0.835	0.891	-	-	-
CatBoost	-	0.884	-	-	-
Random Forest	-	0.877	-	-	-

Table 1: Table of results on the validation and test datasets for various models and backbone architectures. The reported scores are mean ROC AUC scores averaged over the validation/test datasets. N_L stands for the number of transformer or Mamba layers and E for the embedding size. All multi-head self-attention layers had 4 heads, we set the state size to 16 for the Mamba layers and we disabled dropout in all cases. For comparison we have also evaluated the tree-based CatBoost [Prokhorenkova et al., 2018] and Random Forest models on the test set.

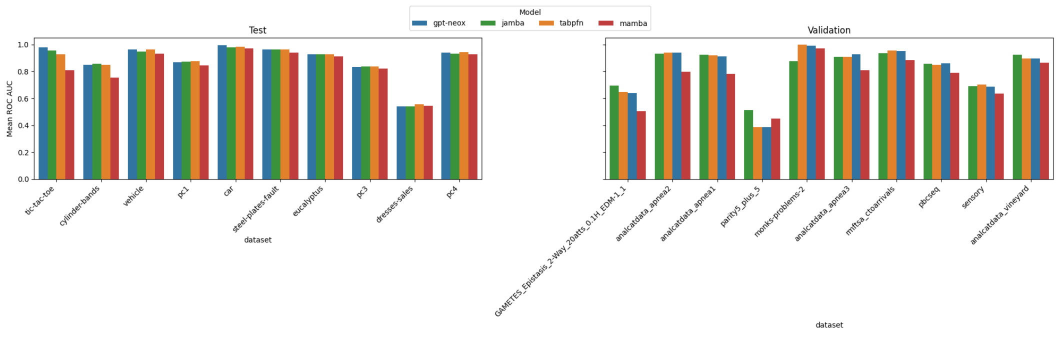


Figure 4: Comparison of model performance across datasets (for each architecture we used the largest evaluated model size). The datasets are ranked in descending order based on the difference in ROC AUC scores between the best and worst-performing models(with ensemble 32). The left plot shows results on the test datasets, while the right plot shows results on the validation datasets. The difference in performance highlights datasets where models exhibit the most variability in effectiveness.

is calculated by subtracting the mean ROC AUC score of the worst-performing model from that of the best-performing model on a given dataset. The result is shown in Fig. 4.

3.3 Analysis of failure cases of Mamba

In order to obtain a better understanding of the failure cases of the Mamba model, we will now focus on two datasets that have a simple structure from the validation and test sets.

From the validation set, we see quite a significant difference in the datasets MONKS Problem 1 and 2, but not 3[3] which might give an indication on what patterns Mamba fails to learn as opposed to the

[3]In the third MONKS problem the dataset has 6 integer features $a_1 \dots a_6$ and is the binary outcome of the logical formula $(a_5 = 3 \text{ and } a_4 = 1)$ or $(a_5 \neq 4 \text{ and } a_2 \neq 3)$. In this problem, which all models considered here perform well on, the models do not have to learn the equality of two features, but rather the equality of a feature to a specific value and the inequality of a feature to a specific value.

models with transformer layers. The MONKS problem datasets [Thrun, 1991] are synthetic datasets and thus offers insight on what kind of patterns the models are able to learn.

For the first MONKS problem, the Mamba models achieve a ROC AUC score (without ensembling) of $\sim 0.78 - 0.85$, while the transformer based models achieve scores close to 1 (see Fig. 6 of the Appendix). The first MONKS problem is a simple classification problem with 6 integer valued features $a_1, ..., a_6$ and 2 target classes, where the target class is determined by the condition $a_1 = a_2$ or $a_5 = 1$.

We performed a random split (500 samples in train and 56 samples in test) on the MONKS Problem 1 dataset from OpenML and performed in-context inference using the Mamba model (5.6M parameters, 12 layers, 256 embedding size) on the test set. The results are shown in Table 3.3 and we see that the Mamba model fails to detect the condition $a_1 == a_2$ (equality of two categorical or numerical features), while it learns the condition $a_5 = 1$ (feature matching a specific value) quite well. This is in contrast to the transformer based models which learn both conditions.

	$N = 56$	
	$a_1 = a_2$	$a_1 \neq a_2$
$a_5 = 1$	5 (0)	22 (2)
$a_5 \neq 1$	3 (14)	10 (0)

Table 2: We count the number of correct (incorrect) predictions of the Mamba model on a test set of 56 samples from the MONKS Problem 1 dataset, depending on which condition is satisfied.

On the test set, we focus on the Tic-Tac-Toe dataset, where the Mamba model performs significantly worse than the transformer-based models.

The Tic-Tac-Toe dataset is a simple classification problem with 9 categorical features $a_1, ..., a_9$, representing the state ("o", "x", "b"=blank) of each field on the 3x3 square. The target class is determined by the condition that the player with the symbol 'x' wins the game.

For this problem we have computed the mean ROC AUC metric over 5 random 50/50 training/test splits on the Tic-Tac-Toe dataset from OpenML and performed in-context inference using the Mamba, GPT-NeoX and the best TabPFN model.

As shown in Table 3, the model performance depends on ordering of how the categorical input features are mapped to numerical values.

In particular, all models perform much better (for Mamba the mean ROC AUC score improves from about 0.78 to close to 1) when the blank field is mapped to the middle ordinal (1), before further preprocessing of numerical values occurs in the TabPFN inference pipeline we are using here.

Overall this suggests, that some of the shortcomings of the Mamba model observed here might be related to the preprocessing of the input data for categorical input features and this offers a potential direction for future work.

Encoding	Mamba (5.6M)	GPT-NeoX (5M)	TabPFN (25.8M)
x=0, o=1, b=2	0.78 ± 0.02	0.92 ± 0.00	0.95 ± 0.02
x=1, o=2, b=0	0.82 ± 0.01	0.94 ± 0.01	0.96 ± 0.01
x=2, o=0, b=1	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
x=1, o=0, b=2	0.82 ± 0.01	0.94 ± 0.01	0.97 ± 0.00
x=2, o=1, b=0	0.78 ± 0.01	0.92 ± 0.00	0.95 ± 0.02
x=0, o=2, b=1	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00

Table 3: Mean (averaged over 5 random splits) ROC AUC scores and standard deviations on the Tic-Tac-Toe dataset for all permutations of mappings of the categorical input features to integer ordinals 0,1,2.

4 Conclusion

Motivated by the recent success of state-space models (SSMs) in various domains, we explored the feasibility of using Mamba for in-context learning on tabular data.

Our results show that, while replacing the TabPFN transformer architecture by a decoder transformer model (GPT-NeoX) results similar prediction performance using Mamba results in a significant drop in performance (when used as a drop-in replacement). Additionally, we studied a hybrid architecture (Jamba) that mixes Mamba layers with transformer layers, showing that the addition of transformer layers already improves metrics significantly. This suggests that Mamba has limitations compared to transformer models in this context to learn certain patterns (already observed in [Jelassi et al., 2024] for selective copying).

There are several directions for future work. As the training loss (on the prior sampled data) of Mamba models is comparable to that the transformer-based models (see Fig. 5 of the Appendix), it could mean that the pre-processing used for inference by TabPFN [Hollmann et al., 2022] that was also used in this work is not optimal for this architecture. One obvious extension would be to investigate the impact of different input encodings for categorical features on the performance of Mamba and other auto-regressive models. Alternatively, one could investigate the impact of using bi-directional SSMs similar to the encoder in TabPFN for the training/in-context examples combined with a custom decoder for the prediction on unseen query examples. We also did not explore alternative SSM based architectures [Gu et al., 2022] [Dao and Gu, 2024], so this could be another direction for future experiments. Yet another direction might be to investigate whether other prior datasets distributions might be more suitable for SSM based models.

Acknowledgments and Disclosure of Funding

We thank Matthias Frank, Marco Spinachi and Sam Thelin for useful discussions and feedback on the manuscript.

References

Bernd Bischl, Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL `https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/c7e1249ffc03eb9ded908c236bd1996d-Paper-round2.pdf`.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé, editors, *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL `https://aclanthology.org/2022.bigscience-1.9`.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/dao24a.html`.

Benjamin Feuer, Niv Cohen, and Chinmay Hegde. Scaling tabPFN: Sketching and feature selection for tabular prior-data fitted networks. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023. URL `https://openreview.net/forum?id=b0OhN0ii36`.

Josh Gardner, Juan C. Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling, 2024. URL `https://arxiv.org/abs/2406.12031`.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2023. URL `https://arxiv.org/abs/2208.01066`.

Riccardo Grazzi, Julien Niklas Siems, Simon Schrodi, Thomas Brox, and Frank Hutter. Is mamba capable of in-context learning? In *AutoML 2024 Methods Track*, 2024. URL https://openreview.net/forum?id=rJhOGOP8nr.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *NeurIPS 2022 First Table Representation Workshop*, 2022. URL https://openreview.net/forum?id=eu9fVjVasr4.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying, 2024. URL https://arxiv.org/abs/2402.01032.

Myung Jun Kim, Léo Grinsztajn, and Gaël Varoquaux. Carte: Pretraining and transfer for tabular learning, 2024. URL https://arxiv.org/abs/2402.16785.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Andreas Müller. For the mothernet baseline, we evaluated a recent model checkpoint provided to us (private communication). Unpublished, 2024.

Andreas Müller, Carlo Curino, and Raghu Ramakrishnan. Mothernet: A foundational hypernetwork for tabular classification, 2023. URL https://arxiv.org/abs/2312.08598.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? A comparative study on in-context learning tasks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39793–39812. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/park24j.html.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

Markus N. Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory, 2022. URL https://arxiv.org/abs/2112.05682.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L. Caterini. Retrieval & fine-tuning for in-context tabular models. In *ICML 2024 Workshop on In-Context Learning*, 2024. URL `https://openreview.net/forum?id=LN5X5rdlb3`.

Sebastian Thrun. The monk's problems-a performance comparison of different learning algorithms, cmu-cs-91-197, 1991. URL `https://api.semanticscholar.org/CorpusID:59699060`.

Boris Van Breugel and Mihaela Van Der Schaar. Why tabular foundation models should be a research priority. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 48976–48993. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/van-breugel24a.html`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. Mambabyte: Token-free selective state space model. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=X1xNsuKssb`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL `https://arxiv.org/abs/1910.03771`.

5 Appendix

5.1 Training loss compared to validation score

Here we summarize the training loss (each data point averaged over one epoch) and validation set mean ROC AUC scores (without ensembling, logged every 10 epochs) as a function of the number of training epochs. Note that while the largest evaluated model with Mamba backbone attains a similar loss as the medium sized model with GPT-NeoX backbone, it underperforms in validation score.

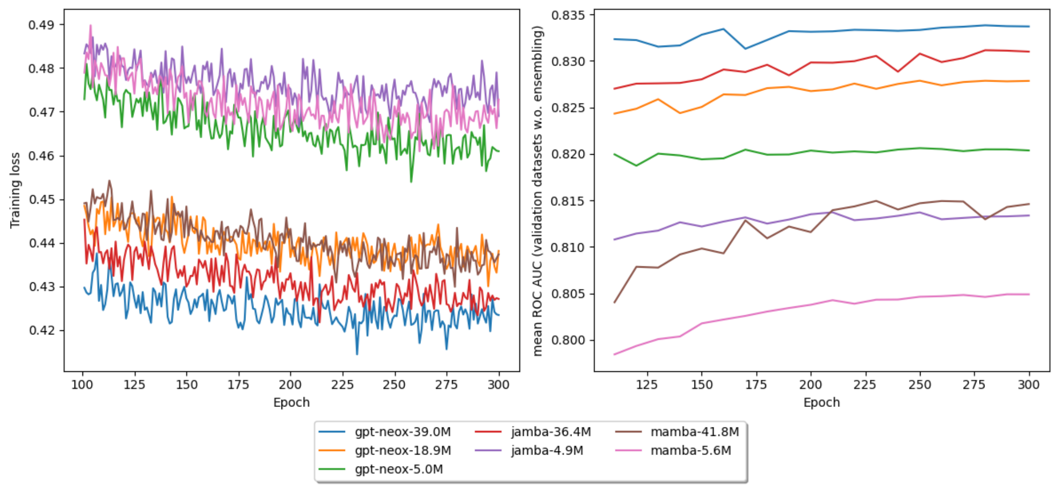
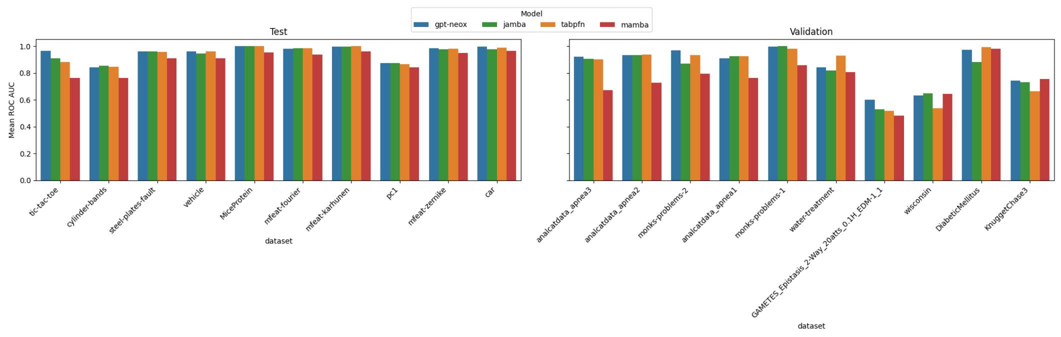


Figure 5: Training loss and validation scores (mean ROC AUC one-vs-one without ensembling).

5.2 Comparison of model performance across datasets without ensembling

Here we plot the same ranking plot as in 4, but this time without ensembling.



5.3 Comparison of the validation scores for the MONKS problems 1 and 3

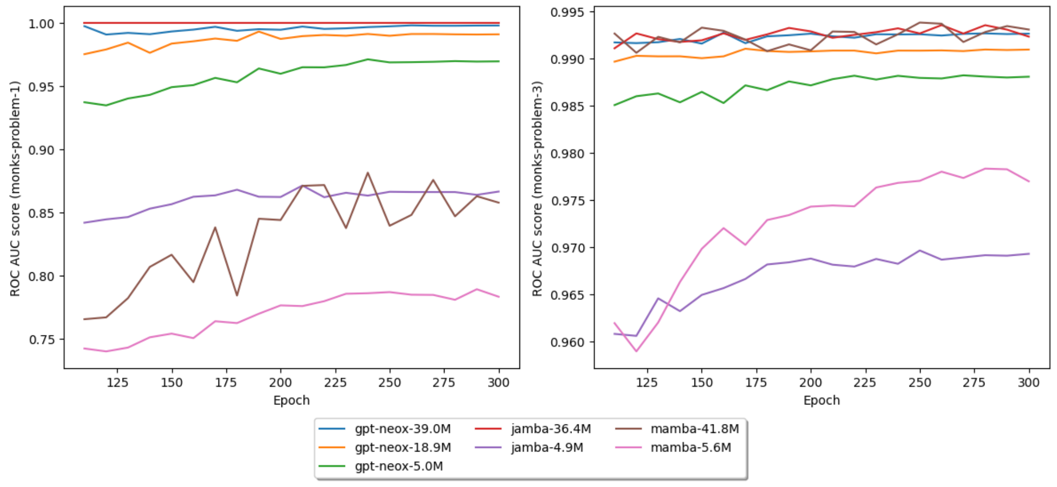


Figure 6: Validation scores for OpenML datasets monks-problem-1 and monks-problem-3 for different model backbones and model sizes (shown here without ensembling).