Retrieval & Fine-Tuning for In-Context Tabular Models

Valentin Thomas[*] valentin.t@layer6.ai
Junwei Ma[*] jeremy@layer6.ai
Rasa Hosseinzadeh rasa@layer6.ai
Keyvan Golestan keyvan@layer6.ai
Guangwei Yu guang@layer6.ai
Maksims Volkovs maks@layer6.ai
Anthony Caterini anthony@layer6.ai

Abstract

Tabular data is a pervasive modality spanning a wide range of domains, and this inherent diversity poses a considerable challenge for deep learning. Recent advancements using transformer-based in-context learning have shown promise on smaller and less complex tabular datasets, but have struggled to scale to larger and more complex ones. To address this limitation, we propose a combination of retrieval and fine-tuning: we can adapt the transformer to a local subset of the data by collecting nearest neighbours, and then perform task-specific fine-tuning with this retrieved set of neighbours in context. Using TabPFN as the base model – currently the best tabular in-context learner – and applying our retrieval and fine-tuning scheme on top results in what we call a locally-calibrated PFN, or LoCalPFN. We conduct extensive evaluation on 95 datasets curated by TabZilla from OpenML, upon which we establish a new state-of-the-art with LoCalPFN – even with respect to tuned tree-based models. Notably, we show a significant boost in performance compared to the base in-context model, demonstrating the efficacy of our approach and advancing the frontier of deep learning in tabular data.

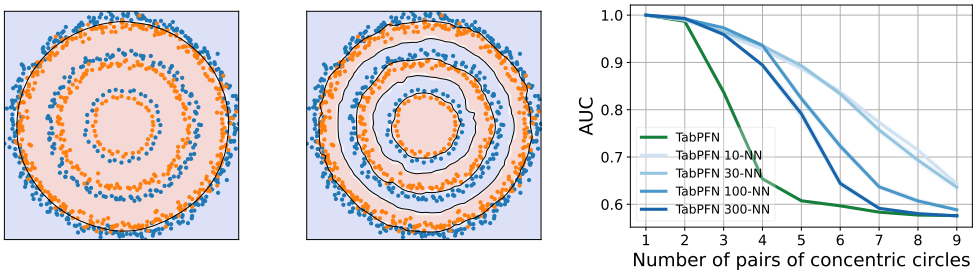
1 Introduction

Tabular data is the most pervasive modality for practical problems in data science, spanning across a wide variety of domains including finance, healthcare, and science [3, 47, 12, 46, 48]. The diversity and heterogeneity of tabular data pose great challenges for deep learning approaches [21], unlike modalities such as text and images in which neural networks can be designed to specifically exploit inductive biases underlying the data [9]. As such, obtaining a performant neural network on a particular tabular data task often results in expensive iterations of training and hyperparameter tuning. Meanwhile, tree-based methods such as XGBoost [11] and CatBoost [40] have proven to be more robust to the inherent challenges of tabular data, and thus have remained the dominant approach for this setting [21, 44, 9]. Yet recently, there has been progress made with transformers and In-Context Learning (ICL): one such example is TabPFN [26], which is trained using a prior-fitting procedure [36] that exposes the network to millions of possible data-generating processes, thus taking a step towards encapsulating the heterogeneity of tabular data. Such approaches differ from classical algorithms in that they process entirely new datasets in a single forward pass and obviate the need for training and hyperparameter tuning.

Despite the promise of transformer-based ICL methods in the tabular setting – particularly on smaller datasets – scaling remains an issue: memory scales *quadratically* with the size of the context. This limits performance when the entire dataset cannot fit into memory, and contrasts with classical algorithms that tend to improve as the amount of available data increases. In addition to this, and as depicted in Figure 1, TabPFN in particular can struggle with underfitting as dataset *complexity*

[*]Equal contribution

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) Vanilla TabPFN, full context (b) TabPFN-kNN, $k = 100$ (c) Performance vs. Complexity

Figure 1: a) TabPFN – even when using the entire training data as context – underfits and cannot classify patterns such as three pairs of concentric circles of two classes. Decision boundaries are in black and shaded areas show the predicted class. b) Applying an adaptive local context for each point using its k nearest neighbours can easily solve this problem. c) We observe that this approach is robust to the numbers of neighbours used (k) even when the dataset complexity increases and always performs better than vanilla TabPFN using full context ($N = 1000$). Each point is averaged over 25 seeds.

increases, even when the entire dataset fits into the context; we observe this shortcoming in real datasets as well, and suspect this could apply to any ICL-based model for tabular data.

To improve the scaling of tabular ICL methods in both dataset size and complexity, we draw on two techniques that have been incredibly successful in foundational large language models: retrieval [31] and fine-tuning [6]. On the retrieval side, we use the k-Nearest Neighbours (kNN) of a given query point as the context for classification; modifying the context in this way empirically allows for both enhanced processing of larger datasets and more complex decision boundaries. We also fine-tune end-to-end for each task, using an approximate neighbour scheme to facilitate backpropagation, and demonstrate significant performance gains beyond just kNN. We named our model Locally-Calibrated PFN – or LoCalPFN for short – to represent the addition of retrieval and fine-tuning on top of a base TabPFN model, although this idea should naturally transfer to potential future ICL-based tabular foundation models as well [49]. We demonstrate that LoCalPFN is state-of-the-art when comparing against both neural approaches and well-tuned tree-based techniques across a 95-dataset benchmark from TabZilla [35]. We summarize our contributions below:

1. Provide insights into TabPFN – the current state-of-the-art tabular ICL transformer-based framework – and analyze how its performance scales across several axes in both synthetic and real datasets. We identify failures to scale in both dataset size and complexity.
2. Propose LoCalPFN to address the scaling failures mentioned above, using a combination of retrieval and fine-tuning to allow for more effective use of the context.
3. Show LoCalPFN compares favourably to strong baselines on a large variety of datasets through extensive experimentation, analysis, and ablation.

2 Improving Tabular In-Context Learning with Retrieval and Fine-Tuning

In this section, we describe ICL applied to tabular data – in particular TabPFN – and the limitations of such an approach. Then, we present our contributions where we treat the in-context learner as a base model on top of which retrieval and fine-tuning are applied.

2.1 Preliminaries on In-Context Learning for Tabular Data and TabPFN

Our method generally applies to in-context learners, specifically for classification tasks on tabular data. While, at the time of writing the only successful model of that type is TabPFN [26], we expect other such base models to be published in the future. TabPFN is trained using a prior-fitting procedure [36] where a large number of synthetic datasets are generated using randomly initialized neural networks. This approach trains an underlying transformer-based network on various generative processes designed to simulate the diverse interrelations that exist among the features of realistic tabular datasets.

After the prior-fitting procedure, the learned TabPFN model ingests an entire training dataset $\mathcal{D}_{\text{train}} \triangleq \{(x^i_{\text{train}}, y^i_{\text{train}})\}_{i=1}^N$ consisting of feature-label pairs $x^i_{\text{train}} \in \mathbb{R}^D$ and $y^i_{\text{train}} \in \{1, \ldots, C\}$ for $i \in \{1, \ldots, N\}$, along with features of a query point x_{qy} (potentially in a batch), and outputs a distribution over labels $y_{\text{qy}} \in \{1, \ldots, C\}$. Specifically, denoting the TabPFN network (outputting logits) with parameters θ as f_θ, the resulting posterior predictive distribution is modelled by:

$$p_\theta(y_{\text{qy}} \mid x_{\text{qy}}, \mathcal{D}_{\text{train}}) = \frac{\exp(f_\theta(x_{\text{qy}}, \mathcal{D}_{\text{train}})[y_{\text{qy}}])}{\sum_{c=1}^C \exp(f_\theta(x_{\text{qy}}, \mathcal{D}_{\text{train}})[c])}, \tag{1}$$

where $[\cdot]$ denotes the vector indexing operation.

Contrary to classical machine learning methods which are trained on one dataset and then evaluated on the same distribution, TabPFN has been shown to be able to perform classification on a wide range of tasks without training, thanks to its diverse prior-fitting procedure. This makes it one of the rare foundation models for tabular data. Key to this is the ICL ability of TabPFN: by using various training *examples* as context, analogous to how transformers on language use the preceding *tokens* as context, TabPFN can classify new query points in a single forward pass.

2.2 What Constitutes a Good Context for Tabular Data?

The quadratic growth of memory usage with context length in transformers presents a challenge: the number of support examples we can use is limited. For instance, while TabPFN performs best on small and simple datasets, where the entire training set fits within the context, it is unclear how to best use TabPFN for large and complex datasets. Naïvely, we might consider a random subsample of the training data as context [35, 19]. However, Ma et al. [34] show that this method does not scale either and observe a drop in performance as the dataset size increases.

Given these limitations, it is natural to ask *"What constitutes a good context for tabular data?"*. This topic has been thoroughly researched in natural language processing, which resulted in various techniques for prompt engineering. The situation is more complicated in the tabular domain, as there is no natural order to tabular data as opposed to the natural order of the words in language.

Specifically for TabPFN, some attempts have been made to use a summary of the dataset as context, through either k-means centroids [19] or direct prompt optimization [19, 34]. Yet in either case the flexibility of the method is limited by the use of a single context for all query points. Instead, we propose a different approach here, where we use a local context tailored to each individual point we wish to classify. For tabular data, we hypothesize that the most critical information to classify a query point x_{qy} is contained in its vicinity. Extensive evaluations [35] support this fact by showing that a simple kNN classifier can rival modern deep architectures designed for tabular data, such as TabNet [2] and VIME [54]. We thus believe that using nearby points as context is a good inductive bias for tabular data classification.

2.3 Better Expressivity and Scaling Using Local Information

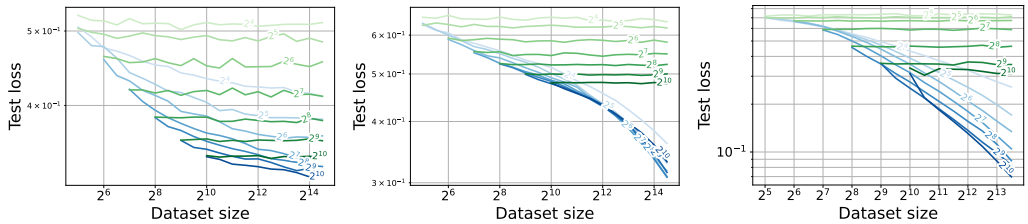
To do this, the first step is to replace the *global* context by a *local* context, i.e., with $k\text{NN}(x_{\text{qy}})$ as the k-nearest neighbours of the query x_{qy} in the training data $\mathcal{D}_{\text{train}}$, we replace equation 1 by

$$p_\theta(y \mid x_{\text{qy}}, \mathcal{D}_{\text{train}}) = \frac{\exp(f_\theta(x_{\text{qy}}, k\text{NN}(x_{\text{qy}}))[y])}{\sum_{c=1}^C \exp(f_\theta(x_{\text{qy}}, k\text{NN}(x_{\text{qy}}))[c])}. \tag{2}$$

Better Expressivity It is well known that in kNN regression and classification, the number of neighbours k controls the bias/variance trade-off and as such the expressivity of the model. More precisely, large k tends to "oversmooth" and suffer from high bias/underfitting, while small k enables more complex decision boundaries but can suffer from more variance/overfitting [24]. We show that this phenomenon is still true for transformers, beyond the simple kNN classifier, in Figure 1. We generate datasets of size $N = 1000$ so that it can be used as context by TabPFN without subsampling. As we increase the complexity of the dataset, measured by the number of concentric circles in this case, TabPFN fails to accurately classify (e.g., for 3 pairs of circles in (a) and more generally in (c)). Retrieving fewer samples ($k = 10$, 30, 100, or 300) for each query point using its k-nearest neighbours from the training data leads to large improvements in AUC over TabPFN as the complexity of the data increases ((b) and (c)). Note that $k = 1000$ would correspond to using all samples as

context, and thus is equivalent to vanilla TabPFN. As such there is a continuum between TabPFN using the full dataset as context and our local context method using kNN, which we call TabPFN-kNN.

While Figure 1 is on toy synthetic data, we believe this result remains surprising: *a priori*, we would expect a 25-million-parameter model (TabPFN) to be able to learn a few circles, even with just ICL. Meanwhile, we believe that using local contexts allows TabPFN to fit more complex patterns, such as the three circles of Figure 1, in the same way that using local linear regression enables more expressive (and in that case nonlinear) decision boundaries [13, 23].



(a) `adult-census` (b) `electricity` (c) `eeg-eye-state`

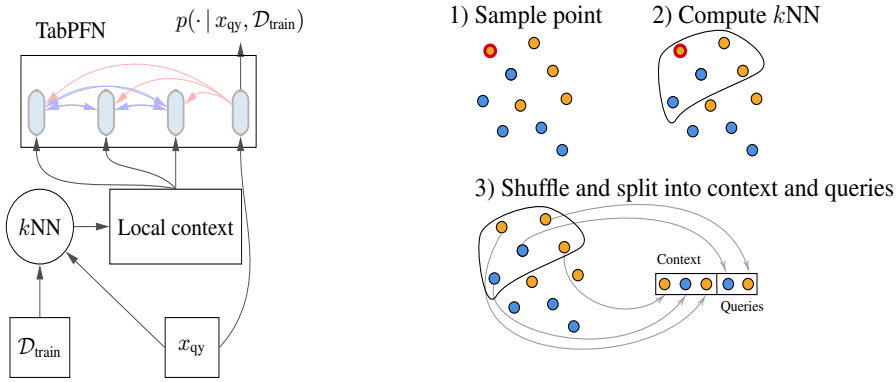
Figure 2: Example of the behaviour of TabPFN and TabPFN-kNN as we vary the dataset size and the context length for three large datasets. TabPFN is in shades of green and TabPFN-kNN is in shades of blue. The opacity represents the context length used (also labelled on each line). It corresponds to random training samples for TabPFN and nearest neighbours for TabPFN-kNN. TabPFN is limited by context size and cannot make efficient use of larger datasets. While for context length = dataset size ($k = N$), TabPFN and TabPFN-kNN have the same performance, TabPFN-kNN can leverage larger datasets with kNN-based contexts and shows improvements, often even for lower context lengths. Each point on this plot is the average of 100 random resamplings of the data.

Better Scaling Using a local context has another benefit: it allows our method's performance to scale with the training dataset size. In machine learning, it is generally expected that the performance of an algorithm improves as the training set size N increases, since the empirical risk converges to the expected risk [50]. However, ICL-based methods (such as TabPFN) that require subsampling when the maximum context length is smaller than N do not scale with N. TabPFN-kNN, on the other hand, can still benefit from larger training set sizes N even when the number of neighbours k is much smaller than N, as the search is performed over the whole training set. We demonstrate this fact in Figure 2 for three real datasets. While the exact patterns in the loss curves differ, we observe a similar trend across many datasets, where the benefits of using TabPFN-kNN grow as the dataset becomes larger. In Figure 9 we provide more detailed figures which include training loss.

2.4 Efficient End-to-End Fine-Tuning With Retrieved Samples

In addition to retrieval, we fine-tune the model end-to-end on each dataset to further improve performance, as is common in Retrieval-Augmented Generation (RAG) [31]. However, naïve fine-tuning is not computationally efficient. Transformer-based in-context models work with inputs of shape $(B, L_{\text{ctx}} + L_{\text{qy}}, d)$ where B is the batch size, L_{ctx} and L_{qy} are the context and query lengths, and d is the embedding dimension. TabPFN uses only one fixed context for all points, with $B = 1$, L_{ctx} the training dataset size (or maximum context length if too large), and $L_{\text{qy}} = N_{\text{qy}}$ the number of points to classify. Contrary to text, there is no auto-regressive attention mask: the context examples all attend to each other (blue arrows on Figure 3a) while the queries only attend to the context and not to each other (red arrows on Figure 3a). Therefore, the predicted classes can be computed in parallel and at a reduced memory footprint.

By comparison, when using a local context with exact neighbours, the context is no longer shared, and therefore the batch dimension must be used for queries: the input has shape $B = N_{\text{qy}}$, $L_{\text{ctx}} = k$ – the number of neighbours – and $L_{\text{qy}} = 1$, since the queries use distinct contexts. This is significantly less efficient than the inference performed by TabPFN, which both requires much less memory, and also allows the queries to be processed in parallel. Therefore, our main limitation is in fact the forward and backward passes when using exact neighbours, unlike most applications where retrieval is the bottleneck. As such, most common approximate kNN methods cannot address this issue.



(a) Overall architecture of LoCalPFN (b) Efficient local context computation for fine-tuning

Figure 3: Details of the architecture and the efficient context used during fine-tuning. a) During inference, for each query x_{qy}, we compute its kNNs and use them as context. b) During fine-tuning, we have a modified procedure allowing shared context between many queries. We first select a random training point, then compute its kNNs. Finally we randomly split those into a context and a query set, allowing us to have a shared (yet local) context for many queries, similarly to vanilla TabPFN. Colours correspond to classes, highlighting that different classes can (and should) appear in the same context.

Instead, to improve computational efficiency during the end-to-end fine-tuning, we opt for a simple neighbour approximation technique wherein many queries share the same context. An illustration of the method is provided in Figure 3b for a single batch dimension. More generally, let us assume that we want to pass gradients on N_{qy} examples at once, using a context length of L_{ctx}. We propose to only use B different contexts, which we will use to classify N_{qy}/B samples each: First, B training examples are sampled. Then, their individual kNN search is performed with $k = L_{\text{ctx}} + L_{\text{qy}}$[2] for $L_{\text{qy}} = N_{\text{qy}}/B$. Finally, those batches of k samples are each shuffled and split into a context vector of length L_{ctx}, and a query vector of length L_{qy}, constructing the input vector of size $(B, L_{\text{ctx}} + L_{\text{qy}}, d)$. This allows us to efficiently trade-off accuracy of the neighbours versus computational complexity: with lower B we share contexts between many points but this comes at the cost of an approximation in the kNN search as the notion of neighbourhood is not transitive, i.e., the neighbour of your neighbour might not be your neighbour. However each sequence in each batch only contains examples which are in the general vicinity of each other. In practice, we observe that this method does not lead to any significant degradation in performance while allowing much faster training.

3 Related work

Foundational Techniques to Improve Tabular Deep Learners Deep learning techniques have historically struggled on tabular data [21], where inductive biases are much harder to capture architecturally [4] as compared to text or images. The comparative lack of progress on a large foundation model for tabular data [49] is yet more evidence of this. However, recent approaches have successfully begun to leverage foundational ideas to improve performance. For example, Non-Parametric Transformers [30] and SAINT [45] both combine row-attentive transformer-based backbones with some form of self-supervised pre-training; however, the former is limited by context size (a common theme for naïve ICL-based learners), whereas the latter is not based on ICL and thus does not as easily apply to novel datasets. Models such as RIM [41] and TabR [20] on the other hand demonstrate how to effectively design tabular deep learners incorporating retrieval modules, but still require costly and brittle rounds of hyperparameter tuning to adapt to any specific dataset. Our approach is meant to target some combination of all these methods: provide ICL-based generalization capabilities, but without limitations on the context size. The retrieval mechanism within TabR itself relies on kNN, which is one of the most straightforward and widely used retrieval-based machine learning methods [24]. In fact, kNN is still being actively studied in the literature, e.g., in Differential Nearest Neighbours Regression (DNNR) [38] and follow-up work [53], which aims to make kNN

[2]Note that, as we sample training points, these original B points are part of their own kNN. To avoid duplicates, we exclude the original B points from the kNN search.

differentiable; this showcases the potential of simple methods like kNN in different forms, although DNNR tackles a separate scope from our method.

TabPFN and Extensions TabPFN [26] is a transformer-based in-context learner that has emerged as a popular model for tabular data, demonstrating strong performance on some benchmarks [35]. It uses a prior-fitting process [36] allowing for rapid adaptation to new tasks. This strong ability to quickly generalize makes TabPFN somewhat of a foundation model for tabular data [49], from which techniques for generation [33] and dataset distillation [34] for example can emerge – interpretability is also being studied [43]. TuneTables [19] attempts to use tabular sketching [37] to summarize the incoming dataset and more effectively scale TabPFN's context; however, much like Ma et al. [34], this approach is limited by the use of a single context for all datapoints, as opposed to an adaptive local context. den Breejen et al. [15] is able to show some limited improvements by fine-tuning TabPFN, which we extend here by more closely pairing the retrieval and fine-tuning aspects. Concurrently, Xu et al. [52] are able to improve TabPFN by first clustering the training data with K-Means and routing each testing point to a given cluster, which is then used as a prompt; our method does not require any clustering of the data.

Links with LLMs The idea of pre-training a model on corpora of text prior to fine-tuning has been explored in the Natural Language Processing domain for both classification and generation tasks [14, 27, 42]. Later iterations refined this idea to train a model and use its in-context learning abilities for new tasks [10]. This elicited research into prompt engineering to determine what to actually put in a model's context [39, 51]. Similar to prompt engineering, to better utilize the model's context, one can search for similar examples from a corpora and use them to facilitate the task; this is known as Retrieval-Augmented Generation (RAG) [31] in the generative context. Other variants of the idea include training jointly with retrieval [22, 8] and augmenting the output of the model with kNN via interpolating [29]. These ideas are analogous to our approach of (i) fine-tuning and retrieving jointly, and (ii) disjoint kNN and fine-tuning in our ablations, respectively. LLMs have also been directly applied to tabular data [16, 25, 18] however, due to the pre-training of these foundation models on large text corpora, there is the possibility of data leakage, which causes concern with evaluations [7]. Note that this is not the case with TabPFN as it has been trained on synthetic data.

4 Experiments

In this section, we showcase the performance of LoCalPFN against a wide range of alternatives. We release all code to reproduce our results at `https://github.com/layer6ai-labs/LoCalPFN`.

4.1 Experimental Setup

We evaluate our methods against competitive baselines using 95 out of the 176 datasets from TabZilla [35], originally sourced from OpenML [5]. These datasets originate from diverse sources, including academic research, competitions, government agencies, and corporations. The 95 datasets are filtered from TabZilla to meet TabPFN's architectural requirements by ensuring that each dataset has at most 100 features, at most 10 classes, does not contain NaN values, and has at least one instance per class for each split. The details of the datasets are described in Appendix A.1. We further split the datasets into two subsets: "small" datasets which contain less than 2,000 instances, and "medium/large" which contain at least 2,000 instances (up to 130,064). For each dataset, we use the splits from TabZilla with train-validation-test ratio of 80:10:10. Since TabPFN was trained with a maximum of 1,024 data points as context size, the small datasets are roughly considered in-distribution for TabPFN whereas the large datasets are considered out-of-distribution.

We conduct our experiments using 10-fold cross-validation over all datasets for all methods. For all baselines, we apply 30 rounds of hyperparameter tuning as in McElfresh et al. [35] and choose the best hyperparameters for each fold according to validation AUC. In addition, the TabPFN baseline is reported without further ensembling or transformations, unless otherwise noted. Our methods also build on top of this same TabPFN baseline without further processing. We also compare against TabPFN with transformations in Section 4.4. More details of the baseline models can be found in Appendix A.2.1. We use the `faiss` [28, 17] library for efficient kNN search in our methods; this enables us to harness parallel computation to accelerate the nearest neighbour search. We evaluate our methods TabPFN-kNN and LoCalPFN against other models in the following sections. Notably, without further fine-tuning, LoCalPFN is identical to TabPFN-kNN. Details of our method are in Appendix A.2.2.

For LoCalPFN we also conducted some small experiments on hyperparameter optimization, but saw no real difference in performance across hyperparameter choices, besides learning rate which we tuned by hand on a global level (i.e., not on a per-dataset basis). Thus, we retained the default hyperparameters we had initially from the TabPFN repository (other than learning rate). We will see later in this section that LoCalPFN is also insensitive to choices in embedding and retrieval metric; combining this with the strong performance across hyperparameter choices shows that the approach is quite robust overall.

Note on evaluation and the computation of proper confidence intervals: While many works evaluate tabular data methods on a small set of datasets and report confidence intervals/standard deviations for those, we choose to evaluate on a large number of datasets in order to have more meaningful results. However, this makes it harder to compute meaningful uncertainty. Agarwal et al. [1] dealt with a related problem in reinforcement learning; we follow their lead by, for example, reporting the interquartile mean (IQM, i.e., the mean of the middle 50% of scores), and we use their library[3] to compute 95% confidence intervals via stratified bootstrapping.

4.2 Main Experiments

As shown in Table 1, averaged over 95 datasets, LoCalPFN outperforms all other baselines, with significant improvement over TabPFN itself. Among the 47 small datasets, we found that TabPFN is in fact quite competitive with other methods, similar to what had been reported by McElfresh et al. [35]. Nevertheless, LoCalPFN further improves the performance even in this setting and positions itself as the best method. For the 48 medium/large datasets, TabPFN underperforms the tree-based methods by a wide margin. Simply applying kNN on top of TabPFN leads to a drastic performance increase on top of TabPFN. Finally, LoCalPFN further improves on TabPFN-kNN, and either performs on par with, or outperforms, all other methods. We also measure the accuracy, F1 score, and relative AUC metrics such as average rank and z-score and see a similar pattern; those details can be found in Tables 6 to 8.

Deep Learning Model Comparisons: Note that most deep learning baselines are significantly more expensive to train and tune on larger datasets, and as such, most of them could not be run on all datasets [35]. Nevertheless, in Table 5 we compare TabPFN-kNN and LoCalPFN to other deep learning based methods on the datasets on which the baselines have been able to run, and show an even larger improvement in performance. The datasets we used for this comparison can be found in Table 4.

Table 1: AUC scores and confidence intervals for all 95 datasets, 47 small datasets, and 48 medium/large datasets, respectively.

	All		**Small**		**Medium/Large**	
Algorithm	IQM AUC	Mean AUC	IQM AUC	Mean AUC	IQM AUC	Mean AUC
kNN	0.843 [0.838-0.847]	0.812 [0.808-0.816]	0.807 [0.798-0.816]	0.781 [0.772-0.789]	0.882 [0.880-0.884]	0.848 [0.847-0.850]
TabPFN	0.917 [0.914-0.919]	0.867 [0.864-0.870]	0.898 [0.892-0.904]	0.849 [0.843-0.856]	0.927 [0.925-0.929]	0.884 [0.883-0.885]
TabPFN 3k	0.924 [0.922-0.927]	0.873 [0.869-0.876]	0.903 [0.897-0.909]	0.852 [0.845-0.858]	0.938 [0.937-0.939]	0.893 [0.892-0.894]
LightGBM	0.940 [0.937-0.942]	0.885 [0.881-0.888]	0.884 [0.876-0.891]	0.838 [0.831-0.845]	**0.966** [0.964-0.967]	0.931 [0.930-0.932]
RandomForest	0.936 [0.934-0.939]	0.886 [0.883-0.890]	0.895 [0.888-0.901]	0.848 [0.841-0.854]	0.955 [0.954-0.956]	0.920 [0.919-0.921]
CatBoost	0.942 [0.939-0.944]	0.891 [0.888-0.895]	0.907 [0.901-0.914]	0.856 [0.849-0.862]	0.961 [0.960-0.962]	0.926 [0.925-0.927]
XGBoost	0.943 [0.940-0.946]	0.892 [0.889-0.895]	0.907 [0.900-0.914]	0.861 [0.854-0.867]	0.965 [0.964-0.966]	0.931 [0.929-0.932]
TabPFN-kNN	0.943 [0.941-0.946]	0.891 [0.887-0.894]	0.922 [0.916-0.927]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.916 [0.915-0.917]
LoCalPFN	**0.958** [0.956-0.960]	**0.908** [0.905-0.911]	**0.937** [0.931-0.942]	**0.882** [0.875-0.889]	**0.968** [0.967-0.969]	**0.934** [0.933-0.935]

4.3 Analysis: Scaling with Dataset Size and Complexity

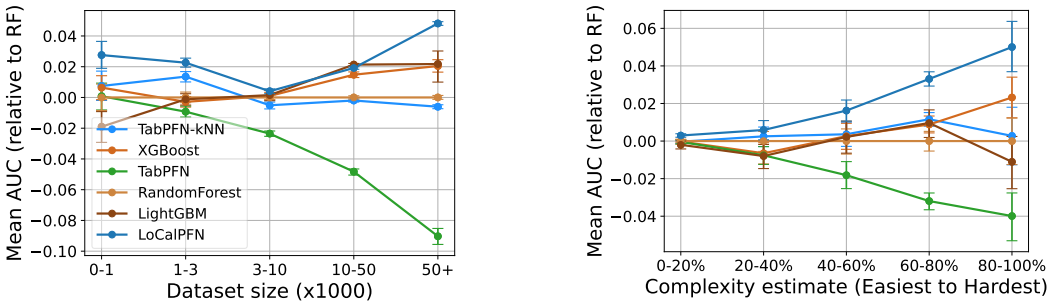
In this section, we further validate that LoCalPFN addresses the scaling problems of TabPFN. We see in Figures 1 and 2 that TabPFN scales badly with both size and complexity; here, we verify this phenomenon in real datasets. While this may appear contradictory to Table 1 of McElfresh et al. [35], which shows TabPFN excelling on a large benchmark suite, we note that the aforementioned study mostly contained small datasets and thus it did not show the same performance drop-off observed here.

Scaling with Size In Figure 4a, we report the AUC of different algorithms relative to the AUC of Random Forest for different dataset sizes. We choose relative AUC for clarity as there is no clear correlation between the maximum AUC attainable on a dataset and its size. We see that, compared to the Random Forest baseline, TabPFN's performance drops drastically when the dataset size increases

[3] `https://github.com/google-research/rliable`

beyond 3,000, indicating poor scaling with N. On the other hand, the other methods we report scale more favourably with the dataset size. We also see that LoCalPFN scales favourably compared to the Random Forest baseline, and even outperforms XGBoost for large datasets. Error bars represent the 95% confidence interval.

Scaling with Complexity While in Figure 1 we could easily control the complexity of the task, there is no generally accepted measure of complexity for an arbitrary dataset. Here, we propose a simple proxy for complexity: for a given dataset, we measure the difference between the best and worst AUCs of a given set of algorithms, similarly to McElfresh et al. [35]. The rationale is that AUC itself cannot capture complexity, as for instance learning to separate two Gaussians can be done optimally by a linear classifier, but the error rate depends on their variances. In Figure 4b, we analyze performance across different levels of this complexity measure. We first calculate the difference in AUC for each dataset using all listed methods in Table 1, then we divide the datasets into five quantiles on the x-axis, with increasing complexity as we move to the right; on the y-axis, we report the mean AUC relative to Random Forest across 10 folds and across the datasets in each bin. We see that TabPFN scales poorly with increasing complexity, and LoCalPFN still outperforms all other methods in the quantiles of higher complexity, demonstrating that its improvements are not just limited to "easy" datasets.



(a) AUC vs. Size

(b) AUC vs. Complexity

Figure 4: Analysis of AUC as a function of size and complexity. TabPFN fails to scale both in size and complexity while LoCalPFN is able to still outperform on the far end of the spectrum. See Figure 8 for a version with absolute AUC. Note that each of the plots contain all datasets in the 95-dataset benchmark, and no subsampling is performed.

4.4 Ablation Studies

In this section, we provide ablation studies on different design choices for LoCalPFN.

Importance of Joint Retrieval and Fine-Tuning One could naïvely consider simply fine-tuning the in-context learner on randomly sampled context during training. In Figure 5 (left) and Table 9, we see that this indeed improves performance over the original TabPFN baseline. However, applying TabPFN-kNN on top of a naïvely fine-tuned model does not improve performance further. Therefore, it is crucial to fine-tune the model end-to-end *with* the retrieval (LocalPFN).

Choice of Embedding We also try different embeddings. The simplest approach is to use the raw standardized features for the nearest neighbour retrieval. In Figure 5 (centre) and Table 10, it is shown that this simple approach is actually very competitive. We compare it to two additional approaches: using one hot encodings (when the size of the resulting vector does not exceed 100 features), and the output of the encoder layer of TabPFN. For the latter, we recompute the search index every 30 gradient steps. The results show that the former, using one hot encodings, does lead to some improvement, however mostly for smaller datasets (see Figure 10 and Figure 11).

Why do simple embeddings work so well? While tabular data is complex in many regards [21], features in tabular data are often semantically meaningful. For this reason, we expect metrics that decompose over individual features, i.e., $d(x, x') = \sum_i d_i(x_i, x'_i)$, to be a good inductive bias for tabular data, especially when it is normalized. This would not be the case for most natural signals. We experimented with two different metrics here, including the Euclidean (L2) distance and cosine similarity; in practice, these two choices are quite similar when applied to standardized features and so we stuck with the Euclidean distance.

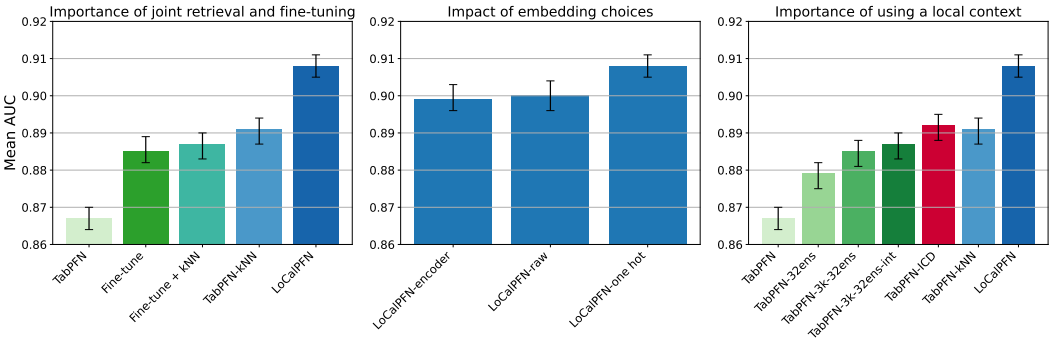


Figure 5: Ablations for different design choices on all 95 datasets. **Left:** Fine-tuning jointly with retrieval yields better performance. **Centre:** The choice of embeddings for retrieval does not change the performance drastically but can lead to some improvements. **Right:** Methods using a context that does not depend on the current query do not match the performance of methods that use a local context.

Importance of Using a Local Context Up until now, we have mostly compared to TabPFN with a random context of size 1,000. To prove our point that using a local context is inherently better than a global context (same context for all queries), we attempt to find the best model using a global context by first using an ensemble of 32 TabPFN models (with randomized feature and class ordering as in Hollmann et al. [26]), which we denote TabPFN-32ens, and then by increasing the context size of the ensembled TabPFN to 3,000 (TabPFN-3k-32ens). As depicted in Figure 5 (right) and detailed in Table 11, while improving significantly upon TabPFN, these are still not competitive even with our TabPFN-kNN. As one can criticize the use of a single context to classify queries, we further experimented with a "Bayesian" view of the probability by averaging it over contexts $p_\theta(y_{\text{qy}} \mid x_{\text{qy}}, \mathcal{D}_{\text{train}}) \triangleq \int_{\mathcal{C}} p_\theta(y_{\text{qy}} \mid x_{\text{qy}}, \mathcal{C}) p(\mathcal{C} \mid \mathcal{D}_{\text{train}}) d\mathcal{C}$, where $\mathcal{C}$ is a context obtained from the training data $\mathcal{D}_{\text{train}}$. We experimented with splitting $\mathcal{D}_{\text{train}}$ into chunks of size 3,000, and averaging the probabilities over those chunks. We call this method TabPFN-32ens-3k-int (for *integral*) and show that, while it does improve upon the single random context, it does not outperform TabPFN-kNN. Additionally, this method is very expensive as: (i) using 3,000 context examples is GPU memory intensive, and (ii) the integral over chunks makes the inference scale as $\mathcal{O}(N)$. The last method we compared to is "In-Context Distillation" [34] (TabPFN-ICD) where, similarly to Feuer et al. [19], the authors directly optimize the context. While this last method leads to better performance (including on larger datasets, see Figure 11), since it performs task-specific tuning it is more comparable computationally to LoCalPFN, which remains superior.

Sensitivity to Number of Neighbours We also ablate the choice of the number of neighbours used as context. This is the only hyperparameter for TabPFN-kNN and also an important hyperparameter for LoCalPFN. In practice, for the number of neighbours, we use the minimum of (i) 10 times the square root of the training set size, and (ii) a pre-defined maximum. For large datasets, the number of neighbours should roughly align with the pre-defined maximum. In Figure 6, we vary this pre-defined maximum while observing the mean AUC on the 48 medium/large datasets. We found that TabPFN-kNN is not very sensitive to this choice as long as it is at least 100. We also see that LoCalPFN is able to improve TabPFN-kNN on all context sizes. Surprisingly, we observe that LoCalPFN is able to outperform the random forest baseline using a maximum context size of only 50, and also outperform the XGBoost baseline with maximum context size of 500. The details of the ablation can be found in Table 12.

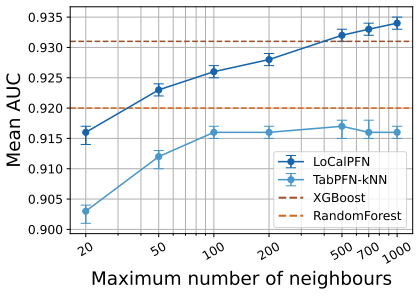


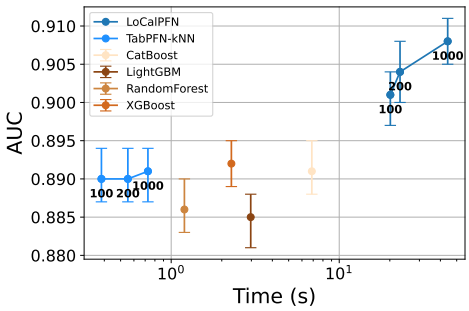
Figure 6: Ablating max # of neighbours

Quality of Approximate Local Context In addition to the above, we also assess the quality of the approximate local context in real datasets in Appendix A.5.5 and Table 13.

4.5 Runtime Study

We also conduct a runtime analysis for LoCalPFN, TabPFN-kNN, and other tree-based models, showing mean test-set AUC as a function of runtime. In Figure 7a, we measure this runtime as the total time taken for training and evaluation. We can see that the general trend for all our algorithms shows a positive correlation between runtime and AUC. We also observe that TabPFN-kNN runs surprisingly fast while still achieving quite high AUC on the 95 datasets. The fast runtime together with very few hyperparameters suggests that TabPFN-kNN will perform very well in practical machine learning engineering and research. LoCalPFN achieves significantly higher performance than all other techniques, and even though it obtains this performance which a much higher runtime, it is also worth noting that the deep learning baselines shown in Table 5 take an even longer time for training and evaluation.

One of the drawbacks of using a local context, though, is that TabPFN-kNN's and LoCalPFN's inference time is slower by 1 to 2 orders of magnitudes when compared to tree-based methods. This fact is to be taken into account when extremely high throughput inference is needed.



(a) Train+Inference time vs. AUC

Algorithm	Training (s)	Inference (s)
CatBoost	6.86	0.01
LightGBM	2.94	0.02
RandomForest	1.15	0.04
XGBoost	2.26	0.01
LoCalPFN-1000	43.47	0.72
LoCalPFN-200	22.45	0.55
LoCalPFN-100	19.76	0.38
TabPFN-kNN-1000	0	0.72
TabPFN-kNN-200	0	0.55
TabPFN-kNN-100	0	0.38

(b) Training and inference times. Note that LoCalPFN has the same inference time as TabPFN-kNN.

Figure 7: a) AUC vs. Runtime for all 95 datasets. TabPFN-kNN has very low runtime and strong performance, while LoCalPFN is able to achieve the highest AUC overall. We use bold text to denote maximum number of neighbours k used. b) Breakdown of the total time in training time and inference time for all algorithms. As local in-context methods are all significantly larger than tree-based methods, their raw inference time is slower.

5 Conclusion and Limitations

In this paper, we demonstrate how to use retrieval and fine-tuning to improve performance on tabular data classification tasks, introducing LoCalPFN as a version of this framework that uses TabPFN as the base model. LoCalPFN breaks new ground for neural approaches on tabular data, even showing improvements over workhorse tree-based techniques. We also provide TabPFN-kNN as a variant without fine-tuning, demonstrating its superiority over the base TabPFN model and practical utility.

However, despite its successes, our framework also has some limitations. The first is that we have only shown that retrieval and fine-tuning improve TabPFN, since it is the only proven ICL-based tabular model. Thus, we cannot be certain that our ideas would directly transfer to new base models, although the success of these concepts in other domains provides some evidence. It is also worth noting that the original RAG paper [31] only initially demonstrated success on BART. Next, the reliance on TabPFN as a base model brings some limitations: besides the constraints on number of features and classes discussed in Section 4.1, we are also unable to easily test our ideas in regression tasks since TabPFN is not designed for them. Although we expect these constraints to gradually be lifted as tabular foundation models improve and increase their scope, we also note that tree-based methods are not nearly as susceptible to these issues. Going further on the comparison with tree-based methods, while we note that LoCalPFN performs better than them in our experimental study, we also point out in Section 4.5 that the runtime of LoCalPFN is slower. However, it is still faster than other deep learning approaches, and the cheaper TabPFN-kNN variant runs as fast as any tree-based method on datasets we studied, while still attaining respectable performance. Overall, we believe that the benefits of our framework far outweigh the limitations, as LoCalPFN greatly expands the capabilities of deep learning on tabular data.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, 2021.

[2] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI Conference on Artificial Intelligence*, pages 6679–6687, 2021.

[3] Omar Benjelloun, Shiyu Chen, and Natasha Noy. Google dataset search by the numbers. In *The Semantic Web – ISWC 2020*, pages 667–682, 2020.

[4] Ege Beyazit, Jonathan Kozaczuk, Bo Li, Vanessa Wallace, and Bilal Fadlallah. An inductive bias for tabular deep learning. In *Advances in Neural Information Processing Systems*, 2023.

[5] 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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[6] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[7] Sebastian Bordt, Harsha Nori, and Rich Caruana. Elephants never forget: Testing language models for memorization of tabular data. *arXiv preprint arXiv:2403.06644*, 2024.

[8] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240, 2022.

[9] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[11] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SigKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[12] Jillian M Clements, Di Xu, Nooshin Yousefi, and Dmitry Efimov. Sequential deep learning for credit risk monitoring with tabular financial data. *arXiv preprint arXiv:2012.15330*, 2020.

[13] William S Cleveland and Susan J Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403): 596–610, 1988.

[14] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, 2015.

[15] Felix den Breejen, Sangmin Bae, Stephen Cha, Tae-Young Kim, Seoung Hyun Koh, and Se-Young Yun. Fine-tuning the retrieval mechanism for tabular deep learning. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[16] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. LIFT: Language-interfaced fine-tuning for non-language machine learning tasks. In *Advances in Neural Information Processing Systems*, 2022.

[17] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *arXiv preprint 2401.08281*, 2024.

[18] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. Large language models (LLMs) on tabular data: Prediction, generation, and understanding – A survey. *Transactions on Machine Learning Research*, 2024.

[19] Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. TuneTables: Context optimization for scalable prior-data fitted networks. In *Advances in Neural Information Processing Systems*, 2024.

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

[21] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, 2022.

[22] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938, 2020.

[23] Trevor Hastie. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

[24] Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009.

[25] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581, 2023.

[26] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

[27] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[28] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[29] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[30] Jannik Kossen, Neil Band, Clare Lyle, Aidan Gomez, Tom Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. In *Advances in Neural Information Processing Systems*, 2021.

[31] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[33] Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen – Tabular data generation with TabPFN. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[34] Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with TabPFN. *arXiv preprint arXiv:2402.06971*, 2024.

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

[36] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

[37] Alexander Munteanu and Chris Schwiegelshohn. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *KI-Künstliche Intelligenz*, 32:37–53, 2018.

[38] Youssef Nader, Leon Sixt, and Tim Landgraf. DNNR: Differential nearest neighbors regression. In *International Conference on Machine Learning*, pages 16296–16317, 2022.

[39] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[40] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, 2018.

[41] Jiarui Qin, Weinan Zhang, Rong Su, Zhirong Liu, Weiwen Liu, Ruiming Tang, Xiuqiang He, and Yong Yu. Retrieval & interaction machine for tabular data prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1379–1389, 2021.

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[43] David Rundel, Julius Kobialka, Constantin von Crailsheim, Matthias Feurer, Thomas Nagler, and David Rügamer. Interpretable machine learning for TabPFN. *arXiv preprint arXiv:2403.10923*, 2024.

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

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

[46] Qi Tang, Guoen Xia, Xianquan Zhang, and Feng Long. A customer churn prediction model based on XGBoost and MLP. In *International Conference on Computer Engineering and Application (ICCEA)*, pages 608–612, 2020.

[47] Dennis Ulmer, Lotta Meijerink, and Giovanni Cinà. Trust issues: Uncertainty estimation does not enable reliable OOD detection on medical tabular data. In *Machine Learning for Health*, pages 341–354, 2020.

[48] Christopher J Urban and Kathleen M Gates. Deep learning: A primer for psychologists. *Psychological Methods*, 26(6):743, 2021.

[49] Boris van Breugel and Mihaela van der Schaar. Why tabular foundation models should be a research priority. In *International Conference on Machine Learning*, 2024.

[50] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2013.

[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[52] Derek Xu, Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. *arXiv preprint arXiv:2405.16156*, 2024.

[53] Han-Jia Ye, Huai-Hong Yin, and De-Chuan Zhan. Modern neighborhood components analysis: A deep tabular baseline two decades later. *arXiv preprint arXiv:2407.03257*, 2024.

[54] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela Van der Schaar. VIME: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 2020.

A Appendix

A.1 Datasets

Table 2: 47 Small Datasets

dataset	did	# instances	# feat	# classes	# cat	imbalance ratio
Australian	146818	690	14	2	8	1.248
LED-display-domain-7digit	125921	500	7	10	0	1.541
acute-inflammations	10089	120	6	2	5	1.400
balance-scale	11	625	4	3	0	5.878
banknote-authentication	10093	1372	4	2	0	1.249
blood-transfusion-service-center	10101	748	4	2	0	3.202
breast-cancer	145799	286	9	2	9	2.365
car-evaluation	146192	1728	21	4	21	18.615
car	146821	1728	6	4	6	18.615
climate-model-simulation-crashes	146819	540	18	2	0	10.739
cmc	23	1473	9	3	7	1.889
credit-g	31	1000	20	2	13	2.333
diabetes	37	768	8	2	0	1.866
dresses-sales	125920	500	12	2	11	1.381
fertility	9984	100	9	2	0	7.333
hayes-roth	146063	160	4	3	0	2.097
hill-valley	145847	1212	100	2	0	1.000
ilpd	9971	583	10	2	1	2.491
ionosphere	145984	351	34	2	0	1.786
iris	59	150	4	3	0	1.000
kc2	3913	522	21	2	0	3.879
monks-problems-2	146065	601	6	2	6	1.917
pc1	3918	1109	21	2	0	13.403
pc3	3903	1563	37	2	0	8.769
pc4	3902	1458	37	2	0	7.191
postoperative-patient-data	146210	88	8	2	8	2.667
profb	3561	672	9	2	4	2.000
qsar-biodeg	9957	1055	41	2	0	1.963
socmob	3797	1156	5	2	4	3.516
sonar	39	208	60	2	0	1.144
steel-plates-fault	146817	1941	27	7	0	12.236
tae	47	151	5	3	2	1.061
tic-tac-toe	49	958	9	2	9	1.886
transplant	3748	131	3	2	0	1.729
vehicle	53	846	18	4	0	1.095
wdbc	9946	569	30	2	0	1.684
yeast	145793	1269	8	4	0	2.704

Table 3: 48 Medium/Large Datasets

dataset	did	# instances	# feat	# classes	# cat	imbalance ratio
GesturePhaseSegmentationProcessed	14969	9873	32	5	0	2.956
JapaneseVowels	3510	9961	14	9	0	2.064
MagicTelescope	3954	19020	10	2	0	1.844
MiniBooNE	168335	130064	50	2	0	2.563
PhishingWebsites	14952	11055	30	2	30	1.257
Satellite	167211	5100	36	2	0	67.000
adult-census	3953	32561	14	2	8	3.153
adult	7592	48842	14	2	8	3.179
artificial-characters	14964	10218	7	10	0	2.360
bank-marketing	14965	45211	16	2	9	7.548
cardiotocography	9979	2126	35	10	0	10.925
churn	167141	5000	20	2	4	6.072
connect-4	146195	67557	42	3	42	6.896
eeg-eye-state	14951	14980	14	2	0	1.228
electricity	219	45312	8	2	1	1.355
elevators	3711	16599	18	2	0	2.236
first-order-theorem-proving	9985	6118	51	6	0	5.255
jannis	168330	83733	54	4	0	22.835
kc1	3917	2109	21	2	0	5.469
kr-vs-kp	3	3196	36	2	36	1.093
magic	146206	19020	10	2	0	1.844
mfeat-fourier	14	2000	76	10	0	1.000
mfeat-karhunen	16	2000	64	10	0	1.000
mfeat-morphological	18	2000	6	10	0	1.000

Continued on next page

Table 3: 48 Medium/Large Datasets

dataset	did	# instances	# feat	# classes	# cat	imblance ratio
mfeat-zernike	22	2000	47	10	0	1.000
mushroom	24	8124	22	2	22	1.075
numerai28.6	167120	96320	21	2	0	1.021
nursery	9892	12958	8	4	8	13.171
optdigits	28	5620	64	10	0	1.032
ozone-level-8hr	9978	2534	72	2	0	14.838
page-blocks	30	5473	10	5	0	175.464
pendigits	32	10992	16	10	0	1.084
phoneme	9952	5404	5	2	0	2.407
pollen	3735	3848	5	2	0	1.000
satimage	2074	6430	36	6	0	2.450
segment	146822	2310	16	7	0	1.000
shuttle	146212	58000	9	7	0	4558.600
spambase	43	4601	57	2	0	1.538
splice	45	3190	60	3	60	2.158
sylvine	168912	5124	20	2	0	1.000
wall-robot-navigation	9960	5456	24	4	0	6.723
wilt	146820	4839	5	2	0	17.540

Table 4: 71 Datasets Selected for Benchmarking Deep Learning Models

dataset	did	# instances	# feat	# classes	# cat	imblance ratio
Australian	146818	690	14	2	8	1.248
LED-display-domain-7digit	125921	500	7	10	0	1.541
Satellite	167211	5100	36	2	0	67.000
acute-inflammations	10089	120	6	2	5	1.400
balance-scale	11	625	4	3	0	5.878
banknote-authentication	10093	1372	4	2	0	1.249
blood-transfusion-service-center	10101	748	4	2	0	3.202
breast-cancer	145799	286	9	2	9	2.365
car-evaluation	146192	1728	21	4	21	18.615
car	146821	1728	6	4	6	18.615
cardiotocography	9979	2126	35	10	0	10.925
churn	167141	5000	20	2	4	6.072
climate-model-simulation-crashes	146819	540	18	2	0	10.739
cmc	23	1473	9	3	7	1.889
credit-g	31	1000	20	2	13	2.333
diabetes	37	768	8	2	0	1.866
dresses-sales	125920	500	12	2	11	1.381
eeg-eye-state	14951	14980	14	2	0	1.228
fertility	9984	100	9	2	0	7.333
first-order-theorem-proving	9985	6118	51	6	0	5.255
hayes-roth	146063	160	4	3	0	2.097
hill-valley	145847	1212	100	2	0	1.000
ilpd	9971	583	10	2	1	2.491
ionosphere	145984	351	34	2	0	1.786
iris	59	150	4	3	0	1.000
kc1	3917	2109	21	2	0	5.469
kc2	3913	522	21	2	0	3.879
kr-vs-kp	3	3196	36	2	36	1.093
mfeat-fourier	14	2000	76	10	0	1.000
mfeat-karhunen	16	2000	64	10	0	1.000
mfeat-morphological	18	2000	6	10	0	1.000
mfeat-zernike	22	2000	47	10	0	1.000
monks-problems-2	146065	601	6	2	6	1.917
mushroom	24	8124	22	2	22	1.075
optdigits	28	5620	64	10	0	1.032
ozone-level-8hr	9978	2534	72	2	0	14.838
page-blocks	30	5473	10	5	0	175.464
pc1	3918	1109	21	2	0	13.403
pc3	3903	1563	37	2	0	8.769
pc4	3902	1458	37	2	0	7.191
phoneme	9952	5404	5	2	0	2.407
pollen	3735	3848	5	2	0	1.000
postoperative-patient-data	146210	88	8	2	8	2.667
profb	3561	672	9	2	4	2.000
qsar-biodeg	9957	1055	41	2	0	1.963
satimage	2074	6430	36	6	0	2.450
segment	146822	2310	16	7	0	1.000
socmob	3797	1156	5	2	4	3.516
sonar	39	208	60	2	0	1.144

Continued on next page

Table 4: 71 Datasets Selected for Benchmarking Deep Learning Models

dataset	did	# instances	# feat	# classes	# cat	imblance ratio
spambase	43	4601	57	2	0	1.538
splice	45	3190	60	3	60	2.158
steel-plates-fault	146817	1941	27	7	0	12.236
tae	47	151	5	3	2	1.061
tic-tac-toe	49	958	9	2	9	1.886
transplant	3748	131	3	2	0	1.729
vehicle	53	846	18	4	0	1.095
wall-robot-navigation	9960	5456	24	4	0	6.723
wdbc	9946	569	30	2	0	1.684
wilt	146820	4839	5	2	0	17.540
yeast	145793	1269	8	4	0	2.704

A.2 Experiment Details

A.2.1 Baseline Details

We use the experimental results from TabZilla [35] when they are available; in particular, we do not use TabZilla's results for the TabPFN variants because they are not always complete, and there is one dataset for CatBoost which does not have any results in the TabZilla repository. These results include the tree-based models and the deep learning model baselines. These results can be found in `https://github.com/naszilla/tabzilla` and `https://drive.google.com/drive/folders/1cHisTmruPHDCYVOYnaqvTdybLngMkB8R`. For different variations of TabPFN inference techniques, we conduct experiments directly using the TabPFN repository `https://github.com/automl/TabPFN`.

A.2.2 LoCalPFN Details

For all TabPFN-kNN experiments, we use a fixed number of neighbours equal to the minimum of (i) 10 times the square root of the dataset size, and (ii) 1000. We find this works well since it adapts to small and large datasets. We use a batch size of 512 for inference using the faiss library for speedup.

For LoCalPFN experiments, we use the exact same setup as TabPFN-kNN during inference. Therefore, at step 0, LoCalPFN and TabPFN-kNN are equivalent. For training LoCalPFN, we adopt the AdamW [32] optimizer with a learning rate of 0.01 and weight decay of 0.01. We do not have warmup or a learning rate scheduler. For the approximate local context for training, we use the same number of neighbours as TabPFN-kNN. We use a fixed number of query points (1,000) sampled from the training set and a batch of 2. For our reported results, we also use one-hot encoding for neighbour retrieval and inference. In addition, we evaluate our model every 30 gradient steps and apply early stopping based on the validation set AUC for each fold respectively.

All experiments for our proposed methods can be run on a machine with a single NVIDIA RTX 6000 GPU Ada Generation, 995Gi RAM, and AMD Ryzen Threadripper PRO 5995WX 64-Cores CPU. Additional runtime analysis can be found in Figure 7a.

A.3 Additional Experiments

A.3.1 Comparison to Deep Learning Models

In addition to tree-based models, we also compare LoCalPFN and TabPFN-kNN with deep learning based methods. We use the results directly from the TabZilla repository. However, due to the fact that a lot of the deep learning baselines are very computationally expensive, many of them were not able to run on all datasets. Therefore, we propose a subset of the 95 datasets which contains 71 datasets upon which all the deep learning methods could run. The details of the 71 dataset subset can be found in Table 4. The complete results can be found in Table 5. We can see that LoCalPFN still outperforms all other models.

A.3.2 Comparison with Other Metrics

Here we also compare the performance of LoCalPFN with other models using accuracy, F1 score. or relative AUC measures such as average rank and z-score as the metric. We can observe a similar pattern here: LoCalPFN either matches or outperforms other models on either of these metrics as well.

Table 5: LoCalPFN outperforms deep learning baselines significantly.

	All 71 Datasets	
Algorithm	IQM AUC	Mean AUC
VIME	0.771 [0.760-0.782]	0.741 [0.732-0.750]
rtdl_MLP	0.855 [0.848-0.862]	0.812 [0.806-0.818]
TabNet	0.881 [0.874-0.888]	0.825 [0.818-0.832]
STG	0.877 [0.872-0.883]	0.829 [0.823-0.834]
rtdl_ResNet	0.917 [0.912-0.922]	0.862 [0.857-0.867]
rtdl_FTTransformer	0.919 [0.913-0.924]	0.869 [0.864-0.874]
TabPFN	0.929 [0.925-0.932]	0.875 [0.871-0.879]
Fine-Tune	0.936 [0.932-0.939]	0.881 [0.876-0.886]
TabPFN-kNN	0.948 [0.944-0.951]	0.889 [0.884-0.894]
LoCalPFN-encoder	**0.956** [0.953-0.959]	0.892 [0.887-0.897]
LoCalPFN-raw	**0.957** [0.954-0.960]	0.893 [0.887-0.898]
Fine-Tune+kNN	0.951 [0.948-0.954]	0.893 [0.888-0.897]
LoCalPFN	**0.959** [0.956-0.962]	**0.903** [0.899-0.907]

Table 6: Accuracy comparison for LoCalPFN and the baseline models.

	All		Small		Medium/Large	
Algorithm	IQM Acc	Mean Acc	IQM Acc	Mean Acc	IQM Acc	Mean Acc
TabPFN	0.856 [0.853-0.859]	0.817 [0.815-0.820]	0.836 [0.830-0.842]	0.806 [0.801-0.811]	0.871 [0.869-0.872]	0.828 [0.826-0.830]
TabPFN 3k	0.862 [0.859-0.865]	0.823 [0.820-0.826]	0.839 [0.833-0.845]	0.808 [0.803-0.813]	0.881 [0.879-0.882]	0.837 [0.835-0.839]
RandomForest	0.875 [0.873-0.878]	0.839 [0.837-0.841]	0.834 [0.827-0.840]	0.807 [0.802-0.812]	0.900 [0.899-0.901]	0.866 [0.865-0.867]
LightGBM	0.878 [0.875-0.881]	0.842 [0.839-0.845]	0.830 [0.824-0.837]	0.807 [0.802-0.812]	**0.918** [0.916-0.919]	0.886 [0.885-0.887]
CatBoost	0.883 [0.880-0.886]	0.847 [0.844-0.849]	0.844 [0.838-0.850]	0.815 [0.810-0.820]	0.908 [0.907-0.909]	0.876 [0.875-0.877]
XGBoost	0.889 [0.886-0.892]	0.848 [0.845-0.851]	0.840 [0.833-0.847]	0.811 [0.804-0.817]	**0.919** [0.918-0.920]	0.887 [0.886-0.888]
TabPFN-kNN	0.877 [0.874-0.879]	0.843 [0.841-0.845]	0.856 [0.851-0.862]	0.825 [0.820-0.829]	0.891 [0.890-0.892]	0.861 [0.860-0.862]
LoCalPFN	**0.902** [0.900-0.905]	**0.865** [0.863-0.868]	**0.875** [0.869-0.881]	**0.840** [0.835-0.845]	**0.918** [0.916-0.919]	**0.890** [0.889-0.891]

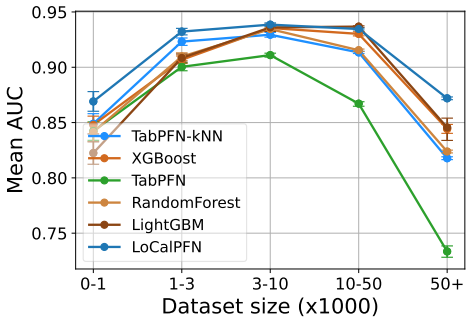
Table 7: F1 score comparison for LoCalPFN and the baseline models.

	All		Small		Medium/Large	
Algorithm	IQM F1	Mean F1	IQM F1	Mean F1	IQM F1	Mean F1
TabPFN	0.843 [0.840-0.846]	0.796 [0.794-0.799]	0.818 [0.812-0.825]	0.783 [0.778-0.789]	0.861 [0.859-0.863]	0.809 [0.807-0.811]
TabPFN 3k	0.850 [0.847-0.853]	0.801 [0.798-0.804]	0.821 [0.814-0.828]	0.784 [0.779-0.789]	0.872 [0.870-0.874]	0.818 [0.816-0.820]
RandomForest	0.875 [0.872-0.877]	0.837 [0.835-0.839]	0.831 [0.824-0.838]	0.805 [0.800-0.811]	0.900 [0.898-0.901]	0.863 [0.862-0.864]
LightGBM	0.877 [0.874-0.881]	0.841 [0.838-0.844]	0.829 [0.823-0.836]	0.806 [0.801-0.811]	**0.917** [0.916-0.919]	**0.885** [0.884-0.886]
CatBoost	0.882 [0.879-0.885]	0.845 [0.843-0.848]	0.842 [0.836-0.849]	0.814 [0.808-0.819]	0.908 [0.907-0.909]	0.875 [0.874-0.876]
XGBoost	0.888 [0.885-0.891]	0.847 [0.844-0.850]	0.839 [0.832-0.846]	0.810 [0.804-0.816]	**0.919** [0.918-0.920]	**0.886** [0.885-0.887]
TabPFN-kNN	0.867 [0.864-0.870]	0.829 [0.827-0.832]	0.841 [0.834-0.847]	0.804 [0.800-0.809]	0.884 [0.883-0.886]	0.854 [0.853-0.855]
LoCalPFN	**0.897** [0.894-0.899]	**0.859** [0.856-0.861]	**0.869** [0.863-0.874]	**0.832** [0.827-0.837]	0.915 [0.913-0.916]	**0.885** [0.884-0.886]

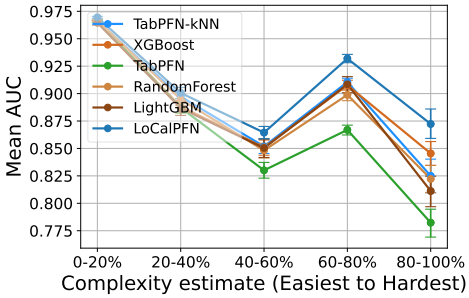
Table 8: Relative score comparison for LoCalPFN and the baseline models. For brevity, we exclude the split between small and medium/large, but that split also tells much of the same story.

Algorithm	Mean AUC Rank	Normalized AUC	AUC z-score
TabPFN	5.3 [4.9, 5.5]	0.56 [0.54, 0.57]	-0.28 [-0.31, -0.25]
TabPFN 3k	4.3 [4.1, 4.7]	0.62 [0.61, 0.64]	-0.07 [-0.10, -0.04]
RandomForest	4.2 [4.0, 4.5]	0.71 [0.70, 0.73]	0.18 [0.16, 0.21]
LightGBM	3.4 [3.0, 3.6]	0.72 [0.70, 0.73]	0.19 [0.15, 0.23]
CatBoost	3.1 [2.9, 3.4]	0.76 [0.75, 0.77]	0.33 [0.30, 0.36]
XGBoost	3.3 [3.1, 3.6]	0.73 [0.72, 0.74]	0.23 [0.19, 0.26]
KNN	7.1 [6.9, 7.3]	0.23 [0.22, 0.25]	-1.38 [-1.42, -1.33]
TabPFN-kNN	3.8 [3.5, 4.0]	0.72 [0.70, 0.73]	0.17 [0.15, 0.20]
LoCalPFN	**1.7** [1.3, 1.8]	**0.85** [0.84, 0.87]	**0.62** [0.59, 0.66]

A.4 Additional Analyses

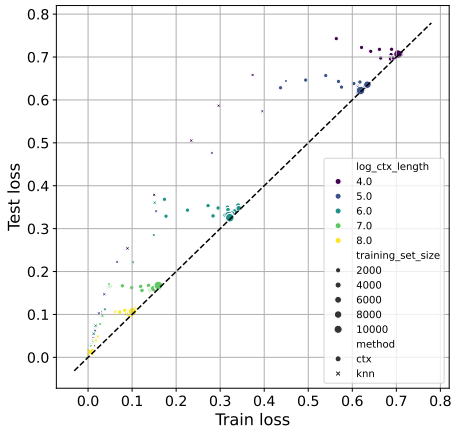


(a) Absolute (i.e., non-relative) mean AUC vs. dataset size

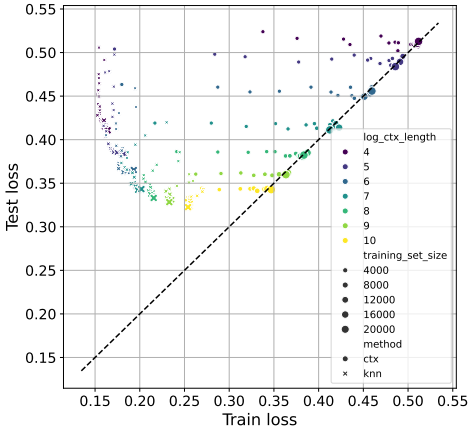


(b) Absolute (i.e., non-relative) mean AUC vs. complexity

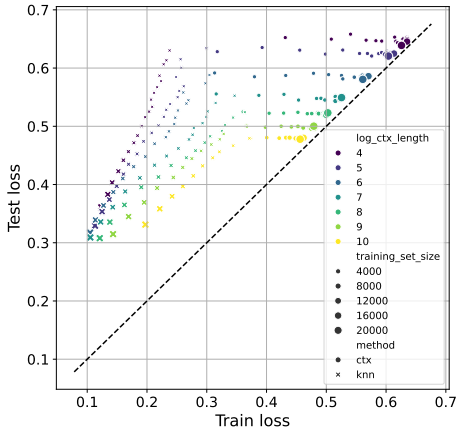
Figure 8: Analysis of AUC as a function of size and complexity. TabPFN fails to scale both in size and complexity while LoCalPFN is able to still outperform on the far end of the spectrum.



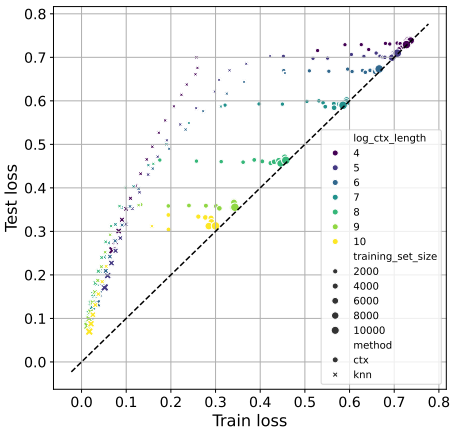
(a) Two pairs of concentric circles



(b) `adult-census`



(c) `electricity`



(d) `eeg-eye-state`

Figure 9: Test loss vs. training loss for TabPFN-kNN (crosses), TabPFN (circles) for different dataset sizes and context/number of neighbours used on four datasets. We observe generally that for low number of neighbours (dark crosses) and especially for small datasets (small crosses) there is significant overfitting (higher test loss than train loss). TabPFN tends to overfit less, especially on larger datasets, which is expected. Overall, using TapPFN-kNN results in better underfitting/overfitting trade-offs where we obtain both lower test and train losses, however the gap between them increases.

A.5 Ablation Studies

Figure 10 and Figure 11 show summaries of ablations on only the small datasets, and only the medium/large datasets, respectively. In the remainder of this subsection we see tables that show even further detail on the results presented in the main text.

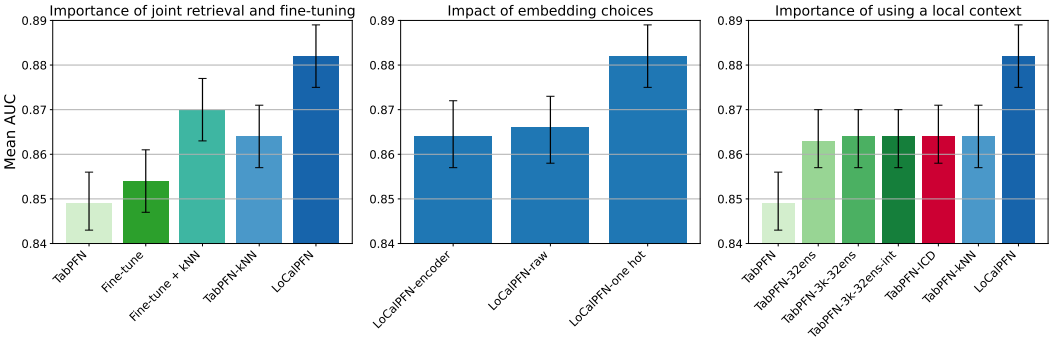


Figure 10: Ablations on Small Datasets

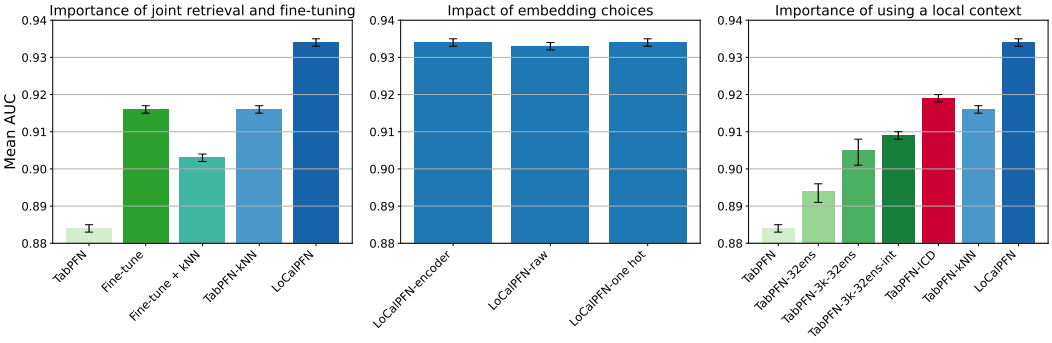


Figure 11: Ablations on Medium/Large Datasets

A.5.1 Importance of Joint Retrieval and Fine-tuning

Table 9: Ablation for fine-tuning. Applying TabPFN-kNN on a fine-tuned model degrades the overall performance. On the other hand, performing local calibration by jointly retrieving and fine-tuning improve performance drastically.

	All		**Small**		**Medium/Large**	
Algorithm	IQM AUC	Mean AUC	IQM AUC	Mean AUC	IQM AUC	Mean AUC
TabPFN	0.917 [0.914-0.919]	0.867 [0.864-0.870]	0.898 [0.892-0.904]	0.849 [0.843-0.856]	0.927 [0.925-0.929]	0.884 [0.883-0.885]
Fine-Tune	0.934 [0.932-0.937]	0.885 [0.882-0.889]	0.905 [0.897-0.911]	0.854 [0.847-0.861]	0.953 [0.951-0.954]	0.916 [0.915-0.917]
Fine-Tune + kNN	0.938 [0.935-0.940]	0.887 [0.883-0.890]	**0.928** [0.922-0.933]	**0.870** [0.863-0.877]	0.946 [0.945-0.948]	0.903 [0.902-0.904]
TabPFN-kNN	0.943 [0.941-0.946]	0.891 [0.887-0.894]	0.922 [0.916-0.927]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.916 [0.915-0.917]
LoCalPFN	**0.958** [0.956-0.960]	**0.908** [0.905-0.911]	**0.937** [0.931-0.942]	**0.882** [0.875-0.889]	**0.968** [0.967-0.969]	**0.934** [0.933-0.935]

A.5.2 Choice of Feature Encoding

Table 10: Ablation for choices of embedding. Converting categorical variables to one-hot gives a relatively moderate gain over other configurations.

	All		**Small**		**Medium/Large**	
Algorithm	IQM AUC	Mean AUC	IQM AUC	Mean AUC	IQM AUC	Mean AUC
LoCalPFN-encoder	0.955 [0.953-0.957]	0.899 [0.896-0.903]	0.926 [0.920-0.932]	0.864 [0.857-0.872]	0.969 [0.967-0.969]	0.934 [0.933-0.935]
LoCalPFN-raw	0.956 [0.954-0.958]	0.900 [0.896-0.904]	0.928 [0.922-0.934]	0.866 [0.858-0.873]	0.968 [0.967-0.969]	0.933 [0.932-0.934]
LoCalPFN-one_hot	0.958 [0.956-0.960]	0.908 [0.905-0.911]	0.937 [0.931-0.942]	0.882 [0.875-0.889]	0.968 [0.967-0.969]	0.934 [0.933-0.935]

A.5.3 Other Inference Methods of TabPFN

Table 11 shows the detailed performance values for TabPFN with different inference methods.

Table 11: Ablation for different TabPFN inference methods.

	All		Small		Medium/Large	
Algorithm	IQM AUC	Mean AUC	IQM AUC	Mean AUC	IQM AUC	Mean AUC
TabPFN-1k-1ens	0.917 [0.914-0.919]	0.867 [0.864-0.870]	0.898 [0.892-0.904]	0.849 [0.843-0.856]	0.927 [0.926-0.929]	0.884 [0.883-0.885]
TabPFN-1k-32ens	0.936 [0.934-0.938]	0.879 [0.875-0.882]	0.923 [0.917-0.929]	0.863 [0.857-0.870]	0.943 [0.941-0.944]	0.894 [0.891-0.896]
TabPFN-3k-32ens	0.943 [0.941-0.945]	0.885 [0.881-0.888]	0.924 [0.918-0.930]	0.864 [0.857-0.870]	0.954 [0.953-0.955]	0.905 [0.901-0.908]
TabPFN-3k-32ens-int	0.945 [0.942-0.947]	0.887 [0.883-0.890]	0.924 [0.918-0.930]	0.864 [0.857-0.870]	0.956 [0.955-0.957]	0.909 [0.908-0.910]
TabPFN-ICD	0.946 [0.944-0.948]	0.892 [0.888-0.895]	0.924 [0.919-0.930]	0.864 [0.858-0.871]	0.958 [0.957-0.959]	0.919 [0.918-0.920]
TabPFN-kNN	0.943 [0.941-0.946]	0.891 [0.887-0.894]	0.922 [0.916-0.928]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.916 [0.915-0.917]
LoCalPFN	**0.958** [0.956-0.960]	**0.908** [0.905-0.911]	**0.937** [0.931-0.942]	**0.882** [0.876-0.888]	**0.968** [0.967-0.969]	**0.934** [0.933-0.935]

A.5.4 Ablation for Maximum Number of Neighbours

Table 12 shows the detailed performance values for varying size of maximum number of neighbours.

Table 12: Ablation for sensitivity of k. The number after c indicates the maximum number of neighbours used.

	All		Small		Medium/Large	
Algorithm	IQM AUC	Mean AUC	IQM AUC	Mean AUC	IQM AUC	Mean AUC
TabPFN-kNN-c20	0.923 [0.920-0.925]	0.874 [0.871-0.878]	0.894 [0.887-0.901]	0.845 [0.838-0.852]	0.937 [0.936-0.939]	0.903 [0.901-0.904]
TabPFN-kNN-c50	0.935 [0.933-0.938]	0.886 [0.882-0.889]	0.911 [0.905-0.917]	0.859 [0.852-0.866]	0.949 [0.948-0.950]	0.912 [0.910-0.913]
TabPFN-kNN-c100	0.943 [0.940-0.945]	0.890 [0.887-0.894]	0.921 [0.916-0.927]	0.864 [0.857-0.871]	0.954 [0.952-0.955]	0.916 [0.915-0.917]
TabPFN-kNN-c200	0.943 [0.941-0.946]	0.890 [0.887-0.894]	0.922 [0.916-0.927]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.916 [0.915-0.917]
TabPFN-kNN-c500	0.943 [0.941-0.946]	0.891 [0.887-0.894]	0.922 [0.916-0.928]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.917 [0.915-0.918]
TabPFN-kNN-c700	0.943 [0.941-0.946]	0.891 [0.887-0.894]	0.922 [0.916-0.927]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.916 [0.915-0.918]
TabPFN-kNN-c1000	0.943 [0.941-0.946]	0.891 [0.887-0.894]	0.922 [0.916-0.927]	0.864 [0.857-0.871]	0.955 [0.953-0.956]	0.916 [0.915-0.917]
LoCalPFN-c20	0.941 [0.938-0.944]	0.890 [0.887-0.894]	0.920 [0.913-0.926]	0.865 [0.858-0.872]	0.953 [0.952-0.955]	0.916 [0.914-0.917]
LoCalPFN-c50	0.950 [0.948-0.953]	0.898 [0.894-0.902]	0.932 [0.925-0.938]	0.873 [0.865-0.881]	0.960 [0.959-0.961]	0.923 [0.922-0.924]
LoCalPFN-c100	0.953 [0.951-0.955]	0.901 [0.897-0.904]	0.932 [0.926-0.938]	0.875 [0.868-0.882]	0.963 [0.962-0.964]	0.926 [0.925-0.927]
LoCalPFN-c200	0.955 [0.953-0.958]	0.904 [0.900-0.908]	0.935 [0.929-0.941]	0.879 [0.872-0.886]	0.965 [0.964-0.966]	0.928 [0.927-0.929]
LoCalPFN-c500	0.957 [0.955-0.959]	0.905 [0.901-0.908]	0.935 [0.930-0.941]	0.877 [0.870-0.883]	0.968 [0.967-0.968]	0.932 [0.931-0.933]
LoCalPFN-c700	0.958 [0.955-0.960]	0.906 [0.902-0.910]	0.935 [0.930-0.941]	0.879 [0.871-0.886]	0.968 [0.967-0.969]	0.933 [0.932-0.934]
LoCalPFN-c1000	0.958 [0.956-0.960]	0.908 [0.905-0.911]	0.937 [0.931-0.942]	0.882 [0.875-0.889]	0.968 [0.967-0.969]	0.934 [0.933-0.935]

A.5.5 Quality of Efficient Local Context

In order to show the efficacy of the efficient local context, we compare LoCalPFN with the exact version where we use the exact neighbours for the context during training. In Table 13, LoCalPFN-exact-b32 indicates the aforementioned configuration with a batch size of 32, which is capped because of the GPU memory constraint. We compare this with another variant of LoCalPFN where we use 32 queries for training, i.e., LoCalPFN-approx-q32. These two variants turn out to have very similar AUCs, which indicates the efficacy of the efficient approximate neighbour search method.

Table 13: Exact nearest neighbour search vs. approximate nearest neighbour search.

	Medium/Large	
Algorithm	IQM AUC	Mean AUC
LoCalPFN-exact-b32	0.967 [0.966-0.968]	0.931 [0.930-0.932]
LoCalPFN-approx-q32	0.968 [0.967-0.968]	0.931 [0.930-0.932]
LoCalPFN	0.968 [0.967-0.969]	0.934 [0.933-0.935]

Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main contributions are highlighted in abstract and introduction. These are supported experimentally in section 2 and 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss related work thorough the paper and have a related work section. Some limitations are discussed thorough the paper and others are addressed in the limitation and conclusion section

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not contain proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We aim at providing all the information necessary to reproduce our figures. We explain how they are constructed and provide all our hyperparameters and the details of the tasks in appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We are releasing code for all experiments and providing a simple-to-use library to encourage adoption of this method before the conference occurs.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Yes, we provide all necessary details to reproduce our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we provide 95% confidence intervals based on stratified bootstrapping following Agarwal et al. [1].

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details of the computational resources needed to conduct all experiments in Appendix A.2.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work does not involve humans or proprietary data. We do not foresee direct risks from the method itself either.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We do not foresee societal impacts for this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We use public models and public data. We do not foresee risks directly arising from the use of our method.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite and reference all code repositories that we use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We use public models and data.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.