Memory Efficient Tabular Foundation Models

Anonymous Authors[1]

Abstract

Tabular Foundation Models, such as TabPFN, have received a large amount of recent attention due to their performance on in-context tabular machine learning tasks, which often exceeds classical baselines. However, practical deployment considerations of these models has received less attention. In this paper we investigate the memory requirements for these models. We demonstrate that employing model compression approaches can enable memory reductions of up to 7.6× with similar levels of performance, reducing deployment requirements by nearly 87%. Our work provides insight to practitioners seeking efficient deployment of these models in practical settings.

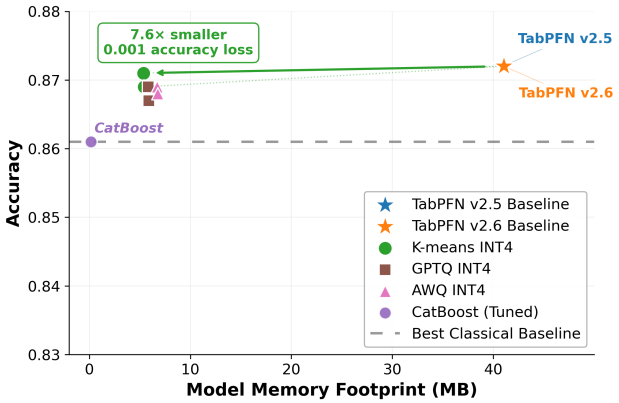


Figure 1. INT4 quantization makes Tabular Foundation Models substantially more memory-efficient while preserving its accuracy advantage. The quantized models reduce the memory footprint by 7.6× relative to full precision baselines, with negligible decline accuracy, and remain above the strongest tuned classical baseline. This demonstrates the potential for quantized TFMs to achieve high predictive performance at a fraction of the deployment cost.

1. Introduction

Tabular machine learning is used heavily in practical domains such as healthcare, finance, operations, and enterprise analytics, where models are often deployed under strict constraints on latency, cost, hardware availability, and data movement (Jiang et al., 2026; Shwartz-Ziv & Armon, 2021; Borisov et al., 2024). Recent tabular foundation models (TFMs), including the TabPFN and TabICL series, are attractive in these settings because they can make strong predictions in-context, reducing the need for task-specific training, hyperparameter tuning, and bespoke model-selection pipelines (Jiang et al., 2026; Hollmann et al., 2023; Qu et al., 2025; Grinsztajn et al., 2025b). This makes them appealing for organizations that need to solve many small or medium-sized tabular prediction problems repeatedly.

However, practical adoption depends not only on predictive performance, but also on whether these models can be deployed and communicated efficiently (Fu et al., 2024). Enterprises must contend with memory movements, hardware constraints, and practical bandwidth considerations. These issues are well-known constraints in servicing large language models, where memory footprint directly affects deployment cost, latency, and the feasibility of local inference (Menghani, 2023; Jeyaraman, 2025). To address this, memory-efficient techniques such as model quantization are now ubiquitous in practical pipelines (Dettmers et al., 2023; Gholami et al., 2022).

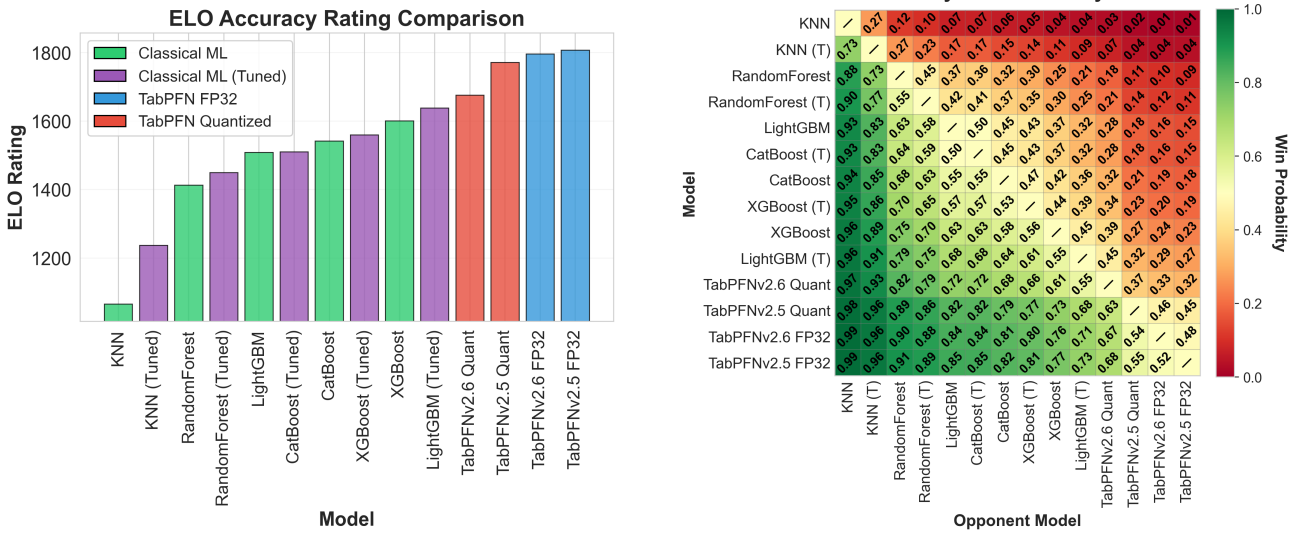
These memory-based limitations also exist for TFMs, however to date most work has focused on reducing *context* constraints, due to architectural limitations and attention-based inference that grows quickly with sample and feature dimensions (Zabërgja et al., 2026; Grinsztajn et al., 2025b; Qu et al., 2026; Zeng et al., 2024; Liu & Ye), leaving practical deployment considerations relatively unexplored.

This paper asks the question: **Can we reduce the deployment requirements for TFMs while retaining their predictive performance?** We evaluate whether existing quantization methods can be employed to reduce the memory footprint of the TabPFN family of models, using standard tabular benchmarks and classical baselines for context.

Our results indicate that quantization is a simple and practical route to memory-efficient TFMs. Across 30 datasets, we find that quantizing to INT4 precision reduces memory

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.



(a) ELO ratings from pairwise accuracy comparisons. Quantized TabPFN models remain close to their FP32 counterparts and rank above all tuned and untuned classical baselines.

(b) Pairwise win probabilities on evaluated datasets. Quantized TabPFN models consistently beat classical baselines, with most losses concentrated against FP32 TabPFN variants.

Figure 2. Quantization preserves most of the accuracy gains of Tabular Foundation Models. Quantized TabPFN variants incur only a modest drop relative to FP32 models while maintaining a clear advantage over strong classical ML baselines.

footprint of TFMs while maintaining similar predictive performance to the full precision counterparts. These results demonstrate that deployment needs can be reduced by up to 7.6× while maintaining performance against tree-based models such as CatBoost (Prokhorenkova et al., 2018).

In this paper we make the following contributions:

- We highlight the practical need for memory-efficient tabular machine learning models and demonstrate that large improvements can be obtained using standard quantization methods.
- We provide a systematic evaluation of these quantization approaches, and qualitative considerations for practical deployment.

We demonstrate our experiments on a range of standard tabular datasets using the TabPFNv2.5 and TabPFN2.6 variants from the TabPFN family of models (Hollmann et al., 2023).

2. Related Literature

2.1. Tabular Foundation Models

Tabular Foundation Models are a class of pre-trained models that can be applied to tabular machine learning problems without additional training (Jiang et al., 2026). These models are trained to predict a posterior prediction (y_{test}) based on a tuple of in-context data ($x_{train}, y_{train}; x_{test}$) (Hollmann et al., 2023; Qu et al., 2025). The training of these models involves extensive pre-training on simulated data of causal relationships and are known as Prior Fitted Networks (PFNs) (Müller et al., 2022; Hollmann et al., 2023). Follow up work has sought to extend these models to more realistic tabular scenarios (e.g. missing data, categorical data, time-series), and to handle limitations in the feature and sample dimension (Hollmann et al., 2025; Grinsztajn et al., 2025a; Qu et al., 2025; 2026; Hoo et al., 2025). The strong performance of these models relative to classical baselines has generated applied interest in a wide range of practical domains such as medical machine learning, financial forecasting, and a variety of industrial uses (Grinsztajn et al., 2025b). The practical handling of large contexts has received a large amount of interest (Zeng et al., 2024; Zabërgja et al., 2026). However, memory requirements of weight deployment have received relatively little attention.

2.2. Quantization and Model Compression

Quantization is a mapping involving the discretization of a signal (Gersho & Gray, 1991). Within machine learning, the term is typically used to refer to *weight* or *activation* quantization, where the goal is to reduce memory by using lower precision values (Gholami et al., 2022).[1] We outline briefly broad trends in the literature and key design decisions.

[1]Note: the term *quantization* is occasionally used to refer to discretization of features in tabular machine learning contexts. We use the term to refer to weight quantization in this paper reflecting our interest in memory efficiency in Tabular Foundation Models.

Table 1. Comparison of model performance and compression methods (30 Datasets).

Model	Method	Accuracy	AUC	Memory (MB)	Compression Ratio
TabPFN-v2.5	Baseline	0.872	0.869	40.95	1.00x
	GPTQ INT4	0.869	0.866	5.81	7.04x
	KMeans INT4	0.871	0.867	5.36	7.65x
	AWQ INT4	0.869	0.866	6.70	6.12x
TabPFN-v2.6	Baseline	0.872	0.870	41.05	1.00x
	GPTQ INT4	0.867	0.865	5.89	6.97x
	KMeans INT4	0.869	0.868	5.43	7.56x
	AWQ INT4	0.868	0.866	6.77	6.06x
XGBoost	Baseline	0.860	0.850	0.096	–
	Tuned	0.862	0.855	0.142	–
CatBoost	Baseline	0.858	0.851	0.073	–
	Tuned	0.861	0.856	0.099	–
LightGBM	Baseline	0.860	0.851	0.124	–
	Tuned	0.864	0.854	0.183	–
KNN	Baseline	0.798	0.704	0.203	–
	Tuned	0.828	0.755	0.220	–
RandomForest	Baseline	0.859	0.850	0.367	–
	Tuned	0.861	0.852	0.853	–

Early work includes binary neural networks (Rastegari et al., 2016; Hubara et al., 2016), the use of quantization aware-training, and integer quantization (Jacob et al., 2018). Classical results show that MSE optimal quantizers correspond closely to a clustering problem (Gersho & Gray, 1991), motivating codebook k-means quantizers (Han et al., 2016).

Recent advances in quantization have included the use of random rotations (Zandieh et al., 2025; Liu et al.; Ashkboos et al., 2024); the use of singular value decomposition to reduce the effect of outlier values (Li* et al., 2025); activation-aware quantization (AWQ) (Lin et al., 2024); and dedicated libraries such as GPTQ and BitsandBytes (Dettmers et al., 2022a;b; 2023; Frantar et al., 2022). A large body of literature exists on applying quantization methods to reduce the memory requirements for modality specific use-cases such as large language model compression (Gholami et al., 2022). However, little work has been devoted to applying weight quantization within tabular foundation models.

2.3. Memory Efficient Tabular Models

Tnere is some work in reducing memory requirements for tabular machine learning models. For example, through the use of low-precision decision gradient boosted trees (Hermann et al., 2025; Shi et al., 2022); and in apply integer quantization to enable efficient performance on FPGA-enabled edge devices (Alsharari et al., 2025). Separately, within TFMs there are some works on achieving high tabular performance with small MLP or distilled models such as TabM (Gorishniy et al., 2025) and TabDistill (Dissanayake & Dutta, 2025). In addition, inference memory requirements have been reduced by compressing context (Zabërgja et al., 2026), or through hypernetworks (Mueller et al., 2025).

3. Method

3.1. Quantization Methods

We evaluate three INT4 post-training quantization approaches indicative of recent trends in model compression: K-Means Quantization (Han et al., 2016), AWQ (Lin et al., 2024), and GPTQ (Frantar et al., 2022). We focus on post-training quantization (PTQ) rather than quantization-aware training, as this can be conducted on a pre-trained model and does not require access to the original pipeline. For GPTQ and AWQ, originally developed for autoregressive language models, we adapt their calibration procedures to TabPFN's in-context learning setting by treating the $(x_{train}, y_{train}; x_{test})$ tuple as the calibration input. For K-Means clustering quantization, we implement an INT4 bit-packing scheme to reduce model artifact size on disk.

3.2. Experimental Setup

We evaluate on a subset of 30 datasets from the OpenML library (Vanschoren et al., 2013), selected to match feature and sample limitations. A full list of datasets is provided in Appendix A.1. We test our three quantization methods on two TabPFN variants: TabPFNv2.5, and TabPFNv2.6 (Grinsztajn et al., 2025b). We use the BZIP2 algorithm to evaluate compressed memory, replicating a practical deployment pipeline (Seward, 1996). We benchmark against standard tabular machine learning methods: XGBoost (Chen

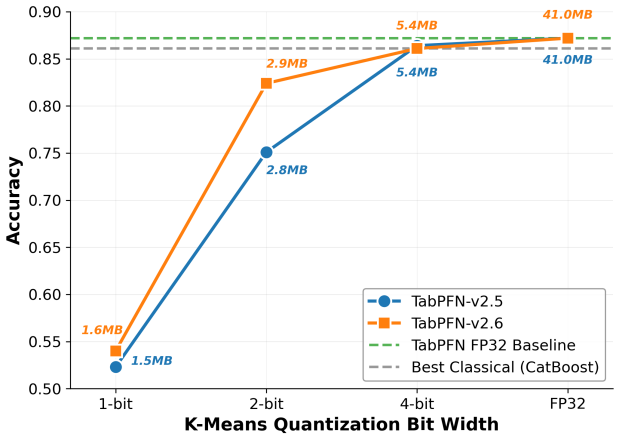


Figure 3. Effect of K-means quantization bit width on TabPFN accuracy and model size. Accuracy improves sharply from 1-bit to 2-bit quantization and largely saturates by 4-bit quantization, where both TabPFN-v2.5 and TabPFN-v2.6 match or approach full precision performance while requiring only 5.4 MB. Further increasing to FP32 yields negligible accuracy gains but increases model size substantially to 41.0 MB, indicating that 4-bit quantization provides the best accuracy-memory trade-off.

& Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), LightGBM (Ke et al., 2017), KNN (Fix & Hodges Jr, 1951; Cover & Hart, 1967), and Random Forest (Breiman, 2001), evaluated in both default-configuration and tuned variants. Hyperparameter details are provided in Appendix A.2.

4. Results

Table 1 and Figure 1 present quantization performance across TabPFNv2.5 and TabPFNv2.6. All three INT4 quantization methods achieve substantial compression (6.0-7.65×) with minimal accuracy degradation ($\leq$ 0.005). Among quantization methods, K-means INT4 achieves the highest compression ratio (7.65××for TabPFNv2.5, and 7.56× for TabPFNv2.6) while incurring the smallest accuracy loss (0.001), outperforming both GPTQ and AWQ across both model versions. Critically, all quantized variants maintain accuracy above the best classical baseline (CatBoost tuned: 0.861), demonstrating that compression preserves TabPFN's performance advantage over traditional methods.

Several further observations support these results. First, a bit-width ablation (Table 3) confirms INT4 as the practical optimum: 2-bit quantization recovers only ∼95% of baseline accuracy and 1-bit collapses to near-random performance, while INT4 retains over 99%. Secondly, Figure 2 shows that the quantized models retain a consistent dataset-level advantage over classical baselines, with a marginal relative ELO loss relative to full precision versions. Third, performance variance across the 30 datasets remains essentially unchanged between baseline and quantized variants (see Appendix Table 4), indicating that quantization does not introduce dataset-dependent instability. Finally, while deployable memory is substantially reduced, inference-time memory and runtime are largely unchanged (see Appendix Table 5). This is consistent with TabPFN's inference cost being dominated by attention activations rather than weights, motivating activation-level compression as future work.

5. Conclusion and Limitations

In this work, we investigated the memory efficiency of TFMs, with a particular focus on quantization-based model compression. Our results demonstrate that substantial reductions in deployment memory is achievable with quantization without compromising predictive performance, making these models more practical for use within industry.

Limitations Our study focuses exclusively on post-training quantization, which may be suboptimal relative to quantization-aware training, which may produce higher performance at lower-precision with access to the training pipeline (Rastegari et al., 2016; Han et al., 2016; Gholami et al., 2022). In addition, our evaluation considers only a subset of available quantization techniques, leaving more advanced approaches such as SVD-based quantization (Ren & Zhu, 2023), TurboQuant (Zandieh et al., 2026), and other emerging methods unexplored. As is standard in quantization for memory deployment, our method requires dequantization prior to inference and so does not enable run-time memory improvements (Gholami et al., 2022). We also do not incorporate kernel-level or hardware-aware optimizations, which may enable FPGA/edge-device-specific implementations. Finally, our method retains the architectural limitations of the foundation models tested which constrain the maximum context window (Grinsztajn et al., 2025a).

Future Work Several promising directions exist for extending this work. Beyond quantization, complementary forms of model compression including pruning (Han et al., 2016), low-rank adaptation (LoRA) (Hu et al., 2022), and knowledge distillation (Hinton et al., 2015), may yield further gains in memory and computational efficiency. Improving inference-time memory efficiency represents another valuable avenue, for example by combining memory-efficient attention mechanisms or context-handling strategies with a quantized approach. Exploring quantization-aware training may further improve efficiency while preserving low-precision predictive performance (Rastegari et al., 2016; Han et al., 2016). Activation quantization strategies, which enable inference in low-precision are a further avenue (Gholami et al., 2022). Finally, integrating hardware-aware optimizations and extending TFM models to physical edge devices remains an exciting potential future direction for a range of practical memory-bound applications.

References

Alsharari, M., Mai, S. T., Woods, R., and Reaño, C. Efficient integer-only-inference of gradient boosting decision trees on low-power devices. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 72(1):241–253, 2025. doi: 10.1109/TCSI.2024.3446582.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. Quarot: Outlier-free 4-bit inference in rotated LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=dfqsW38v1X.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, 2024. doi: 10.1109/TNNLS.2022.3229161.

Breiman, L. Random forests. *Machine Learning*, 45(1): 5–32, 2001. doi: 10.1023/A:1010933404324.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022a.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*, 2022b.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Dissanayake, P. and Dutta, S. Tabdistill: Distilling transformers into neural nets for few-shot tabular classification. *arXiv preprint arXiv:2511.05704*, 2025.

Fix, E. and Hodges Jr, J. L. Discriminatory analysis. nonparametric discrimination: small sample performance. Technical report, California Univ Berkeley, 1951.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Fu, Y., Xue, L., Huang, Y., Brabete, A.-O., Ustiugov, D., Patel, Y., and Mai, L. {ServerlessLLM}:{Low-Latency} serverless inference for large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 135–153, 2024.

Gersho, A. and Gray, R. M. *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media, 1991.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Gorishniy, Y., Kotelnikov, A., and Babenko, A. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

Grinsztajn, L., Flöge, K., Key, O., Birkel, F., Jund, P., Roof, B., Jäger, B., Safaric, D., Alessi, S., Hayler, A., Manium, M., Yu, R., Jablonski, F., Hoo, S. B., Garg, A., Robertson, J., Bühler, M., Moroshan, V., Purucker, L., Cornu, C., Wehrhahn, L. C., Bonetto, A., Schölkopf, B., Gambhir, S., Hollmann, N., and Hutter, F. Tabpfn-2.5: Advancing the state of the art in tabular foundation models, 2025a. URL https://arxiv.org/abs/2511.08667.

Grinsztajn, L. et al. Tabpfn-2.5: Advancing the state of the art in tabular foundation models, 2025b.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. URL https://arxiv.org/abs/1510.00149.

Herrmann, N., Stenkamp, J., Karic, B., Oehmcke, S., and Gieseke, F. Boosted trees on a diet: Compact models for resource-constrained devices. *arXiv preprint arXiv:2510.26557*, 2025.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations 2023*, 2023.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Hoo, S. B., Müller, S., Salinas, D., and Hutter, F. From tables to time: Extending tabpfn-v2 to time series forecasting. *arXiv preprint arXiv:2501.02945*, 2025.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Jeyaraman, B. P. *Deployment Strategies for LLMs*, pp. 103–134. Apress, Berkeley, CA, 2025. ISBN 979-8-8688-1700-7. doi: 10.1007/979-8-8688-1700-7_4. URL https://doi.org/10.1007/979-8-8688-1700-7_4.

Jiang, J.-P., Liu, S.-Y., Cai, H.-R., Zhou, Q.-L., and Ye, H.-J. Representation learning for tabular data: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Li*, M., Lin*, Y., Zhang*, Z., Cai, T., Li, X., Guo, J., Xie, E., Meng, C., Zhu, J.-Y., and Han, S. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

Liu, S. and Ye, H.-J. Tabpfn unleashed: A scalable and effective solution to tabular classification problems. In *Forty-second International Conference on Machine Learning*.

Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., and Blankevoort, T. Spinquant: Llm quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*.

Menghani, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.*, 55(12), March 2023. ISSN 0360-0300. doi: 10.1145/3578938. URL https://doi.org/10.1145/3578938.

Mueller, A., Curino, C., and Ramakrishnan, R. Mothernet: Fast training and inference via hyper-network transformers. In *International Conference on Learning Representations*, volume 2025, pp. 76666–76686, 2025.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0VvD1PmNzM.

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. Tabiclv2: A better, faster, scalable, and open tabular foundation model, 2026. URL https://arxiv.org/abs/2602.11139.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision – ECCV 2016*, pp. 525–542, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Ren, Y. and Zhu, X. An ensemble technique for large language model compression. *arXiv preprint arXiv:2307.xxxx*, 2023.

Seward, J. Bzip2 and Libbzip2, 1996. URL https://sourceware.org/bzip2/. Publication Title: bzip2 : Home.

Shi, Y., Ke, G., Chen, Z., Zheng, S., and Liu, T.-Y. Quantized training of gradient boosting decision trees. *Advances in neural information processing systems*, 35: 18822–18833, 2022.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. URL https://openreview.net/forum?id=vdgtepS1pV.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL https://doi.org/10.1145/2641190.2641198.

Zabërgja, G., Kamel, R., Kadra, A., Frey, C. M., and Grabocka, J. End-to-end compression for tabular foundation models. *arXiv preprint arXiv:2602.05649*, 2026.

Zabërgja, G., Kamel, R., Kadra, A., Frey, C. M. M., and Grabocka, J. End-to-end compression for tabular foundation models, 2026. URL https://arxiv.org/abs/2602.05649.

Zandieh, A., Daliri, M., Hadian, M., and Mirrokni, V. Turboquant: Online vector quantization with near-optimal distortion rate, 2025. URL https://arxiv.org/abs/2504.19874.

Zandieh, A., Daliri, M., Hadian, A., and Mirrokni, V. Turboquant: Online vector quantization with near-optimal distortion rate. In *International Conference on Learning Representations (ICLR)*, 2026. Preprint available at arXiv:2504.19874.

Zeng, Y., Kang, W., and Mueller, A. C. Tabflex: Scaling tabular learning to millions with linear attention. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=f8aganC0tN.

A. Additional Details

A.1. Datasets

All datasets were evaluated by standardised train:test split as provided by Open ML dataset tasks. Details of the datasets used are provided below:

Dataset	Samples	Features	Classes	Description
Amazon employee access	32,769	9	2	Amazon employee access prediction
anneal	898	38	5	Steel annealing process classification
Bank Customer Churn	10,000	10	2	Bank customer churn prediction
Bank marketing	45211	13	2	Bank marketing classification
blood-transfusion-service-center	748	4	2	Blood transfusion donor behavior
churn	5,000	19	2	Telecom customer churn prediction
coil2000 insurance policies	9,822	85	2	Insurance policy purchase prediction
credit card clients default	30,000	23	2	
credit-g	1,000	20	2	German credit risk assessment
diabetes	768	8	2	Pima Indians diabetes diagnosis
E-CommereShippingData	10,999	10	2	
Fitness Club	1,500	6	2	Fitness club membership churn prediction
hazelnut-spread-contaminant-detection	2,400	30	2	Food contamination detection
heloc	10,459	23	2	
HR Analytics Job Change of Data Scientist	19,158	12	2	
in vehicle coupon recommendation	12,684	24	2	
Is-this-a-good-customer	1,723	13	2	Customer quality classification
jm1	1,723	13	2	
Marketing Campaign	2,240	25	2	Marketing campaign response prediction
maternal health risk	1,014	6	3	Maternal health risk classification
MIC	1,699	111	8	Text/molecular classification
NATICUSdroid	7,491	86	2	Android malware detection
online shoppers intention	12,330	17	2	
polish companies bankruptcy	5,910	64	2	Polish company bankruptcy prediction
qsar-biodeg	1,054	41	2	Chemical biodegradability prediction
seismic-bumps	2,584	15	2	Mining seismic hazard prediction
splice	3,190	60	3	DNA splice junction classification
students dropout and academic success	4,424	36	3	Student academic outcome prediction
taiwanese bankruptcy prediction	6,819	94	2	Taiwan company bankruptcy prediction
website phishing	1,353	9	3	Phishing website detection

A.2. Classical Comparisons

Baseline Methods Baseline models were run for comparison using fixed default configurations across multiple independent datasets. The results provided are the mean across all datasets for each model. Chosen parameters are included for reference:

```
XGBoost
    n_estimators: 100
    max_depth: 6
    learning_rate:0.1
    random_state: 42
    n_jobs: -1
    eval_metric: 'logloss'
    objective: 'binary:logistic' (binary) or 'multi:softmax' (multi-class)
    use_label_encoder: False

CatBoost
    iterations: 200
    depth: 6
    learning_rate: 0.1
    random_state: 42
    verbose: False
    thread_counts: -1
    loss_function: 'Logloss' (binary) or 'Multiclass' (multi-class)

LightGBM
    n_estimators: 100
    max_depth: 6
    learning_rate:0.1
    random_state: 42
    n_jobs: -1
    verbose: -1
    force_col_wise: True

RandomForest
    n_estimators: 100
    max_depth: 10
    random_state: 42
    n_jobs: -1
    verbose: 0

KNN
    n_neighbors: 5
    n_jobs: -1
```

Tuning Methods `RandomisedSearchCV` was used with stratified 3-fold cross validation (20 random hyperparameter configurations per model) using the training dataset and reporting the optimized model performance on the test data. Multiple parameter grid search spaces were used covering regularization, tree depth, learning rates, and other model-specific parameters. Details of the parameter search grids are included below:

```
XGBoost (9 parameters)
    max_depth: [3, 5, 7, 10]
    learning_rate: [0.01, 0.05, 0.1, 0.2]
    n_estimators: [50, 100, 200, 300]
    min_child_weight: [1, 3, 5]
    subsample: [0.6, 0.8, 1.0]
```

```
        colsample_bytree: [0.6, 0.8, 1.0]
        gamma: [0. 0.1, 0.2]
        reg_alpha: [0, 0.1, 0.5, 1.0]
        reg_lambda: [0.5, 1.0, 2.0]

    CatBoost (7 parameters)
        iterations: [50, 100, 200, 300]
        depth: [4, 6, 8, 10]
        learning_rate: [0.01, 0.05, 0.1, 0.2]
        l2_leaf_reg: [1, 3, 5, 7]
        border_count: [32, 64, 128]
        bagging_temperature: [0, 0.5, 1.0]
        random_strength: [0.5, 1.0, 2.0]

    LightGBM (9 parameters)
        num_leaves: [15, 31, 63, 127]
        max_depth: [3, 5, 7, 10, -1]
        learning_rate: [0.01, 0.05, 0.1, 0.2]
        n_estimators: [50, 100, 200, 300]
        min_child_samples: [10, 20, 30, None]
        subsample: [0.6, 0.8, 1.0]
        colsample_bytree: [0.6, 0.8, 1.0]
        reg_alpha: [0, 0.1, 0.5, 1.0]
        reg_lambda: [0, 0.1, 0.5, 1.0]

    RandomForest (7 parameters)
        n_estimators: [50, 100, 200, 300]
        max_depth: [5, 10, 20, 30, None]
        min_samples_split: [2, 5, 10]
        min_samples_leaf: [1, 2, 4]
        max_features: ['sqrt', 'log2', 0.5, 0.7]
        bootstrap: [True, False]
        criterion: ['gini', 'entropy']

    KNN (5 parameters)
        n_neighbours: [3, 5, 7, 9, 11, 15, 21]
        weights: ['uniform', 'distance']
        metric: ['euclidean', 'manhattan', 'minkowski']
        p: [1, 2, 3]
        algorithm: ['ball_tree', 'kd_tree', 'brute']
```

B. Hardware Details

Experiments were conducted on an AWS SageMaker `ml.g4dn.12xlarge` instance with 4 parallel workers to avoid resource contention during stress testing. Each worker runs independently with dedicated GPU and CPU resources.

Table 2. Compute environment configuration.

Component	Instance Total	Per Worker
GPU	4× Tesla T4 (16GB VRAM)	1 GPU (16GB)
CPU	48 cores (Xeon 8259CL)	12 cores
RAM	186 GB	40 GB

C. Ablations

C.1. K-means N-bit ablation study

Table 3. K-means N-bit ablation study

Model	Bit Width	Accuracy	ROC-AUC	Model Size (MB)	Compression Ratio
TabPFN-v2.5	Baseline FP32	0.872	0.869	40.95	1.00x
	1-bit	0.523	0.535	1.52	26.94x
	2-bit	0.751	0.863	2.80	14.63x
	4-bit	0.864	0.906	5.36	7.64x
TabPFN-v2.6	Baseline FP32	0.872	0.870	41.05	1.00x
	1-bit	0.540	0.558	1.59	25.82x
	2-bit	0.824	0.875	2.87	14.30x
	4-bit	0.861	0.905	5.43	7.56x

Note: Results computed on 5 representative datasets and may differ slightly from 30-dataset averages in Table 1 due to dataset sampling.

4-bit quantization achieves the optimal trade-off between compression and performance, recovering over 99% of baseline accuracy at 7.65× compression. 2-bit quantization offers a viable alternative at 14-15× compression with 95% accuracy recovery. 1-bit quantization is impractical for production use due to severe accuracy degradation (35% drop).

C.2. Performance stability across datasets

Table 4. Performance stability study across 30 classification datasets

Model	Method	Accuracy	ROC-AUC	Balanced Acc
TabPFN-v2.5	awq INT4	0.869±0.087	0.866±0.090	0.712±0.175
	baseline FP32	0.872±0.086	0.869±0.089	0.720±0.172
	gptq INT4	0.869±0.087	0.866±0.090	0.710±0.174
	kmeans INT4	0.871±0.085	0.867±0.090	0.718±0.171
TabPFN-v2.6	awq INT4	0.868±0.089	0.866±0.091	0.714±0.170
	baseline FP32	0.872±0.086	0.870±0.090	0.722±0.170
	gptq INT4	0.867±0.088	0.865±0.091	0.714±0.171
	kmeans INT4	0.869±0.087	0.868±0.090	0.721±0.167

Note: Results are averaged across 30 datasets. Standard deviation represents performance variation across datasets.

Standard deviation across datasets remains consistent between quantized and baseline variants, indicating that quantization does not introduce additional performance variance across different data distributions.

C.3. Inference memory and speeds

Table 5. Inference memory and speed averaged across 30 OpenML datasets.

Model	Method	Peak RSS (MB)	Inference Mem* (MB)	Δ vs Baseline	Speedup vs FP32
TabPFN-v2.5	FP32	5,893	4,737	-	1.00x
	GPTQ INT4	5,953	4,798	+61 (+1.3%)	1.10x
	KMeans INT4	5,867	4,712	-25 (-0.5%)	0.99x
	AWQ INT4	5,923	4,767	+30 (+0.6%)	1.05x
TabPFN-v2.6	FP32	5,717	4,561	-	1.00x
	GPTQ INT4	5,791	4,634	+73 (+1.6%)	1.00x
	KMeans INT4	5,727	4,571	+10 (+0.2%)	0.9–
	AWQ INT4	5,775	4,619	+57 (+1.3%)	1.00x

**Inference Mem = Peak RSS - Baseline RSS (process overhead removed).*

This table presents an analysis of inference efficiency. Our primary contribution focuses on weight compression to address deployment bottlenecks (model distribution and storage), achieving up to 7× compression. As an additional analysis, we measure inference memory, which remains largely unchanged ($< 2\%$ variation, $4.7GB$), as expected since memory usage is dominated by activations. This observation motivates future work on activation quantization. Inference speed remains stable (±10%), indicating no performance degradation from weight quantization.