Towards Understanding Layer Contributions in Tabular In-Context Learning Models

Amir Rezaei Balef[1,2,3], **Mykhailo Koshil**[2,3] **and Katharina Eggensperger**[2,3]
[1]University of Tübingen [2] TU Dortmund University
[3]Lamarr Institute for Machine Learning and Artificial Intelligence
{amir.balef, mykhailo.koshil, katharina.eggensperger}@tu-dortmund.de

Abstract

Despite the architectural similarities between tabular in-context learning (ICL) models and large language models (LLMs), little is known about how individual layers contribute to tabular prediction. In this paper, we investigate how the latent spaces evolve across layers in tabular ICL models, identify potential redundant layers, and compare these dynamics with those observed in LLMs. We analyze *TabPFN* and *TabICL* through the "layers as painters" perspective, finding that only subsets of layers share a common representational language, suggesting structural redundancy and offering opportunities for model compression and improved interpretability.

1 Introduction

Tabular in-context learning (ICL) models have demonstrated that transformer-based architectures can achieve state-of-the-art performance on small and medium-sized predictive tabular tasks [Erickson et al., 2025]. However, their internal dynamics remain underexplored, and insights from architecturally similar LLM interpretability studies [Gromov et al., 2025, Sun et al., 2025] do not directly transfer due to the differences in inference and training. Unlike LLMs, prominent ICL models (*TabICL*, *TabPFN*) for tabular data do not perform inference in an autoregressive fashion, use attention within the token, and typically do not use positional encodings. Gaining a deeper understanding of how these tabular ICL models function can help identify their strengths, expose their failure modes, and guide future architectural improvements. Motivated by this goal, we aim to open research in this direction by asking the following question(s): **How does data representation evolve in tabular ICL models, and do such models utilize all layers to perform effectively?**

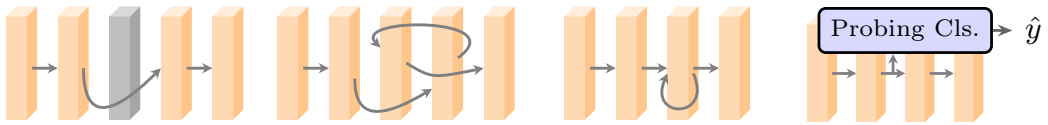
Here we provide a first step to investigate this question based on three popular tabular ICL models, *TabPFN(v1)* [Hollmann et al., 2023], *TabPFN(v2)* [Hollmann et al., 2025], and *TabICL* [Qu et al., 2024]. Inspired by the "Layers as Painters" framework of Sun et al. [2025], we modify the internal embedding flow within a model to analyze how information is transformed across layers.

2 Background and Motivation

We summarize key characteristics of tabular ICL models and findings on layer-wise interpretability.

Tabular ICL models are a particular branch of transformers trained to solve supervised learning problems via in-context learning. In these settings, the input to the model is a support set (train data; feature and target values) and a query (test data; only feature values). The model then predicts the target value for the query, without performing weight updates. Unlike LLMs, tabular ICL models are mostly trained to learn a map between a query token and a label (or value in case of regression). For the ICL to reach its full potential in transformer models, a large amount of data is required [Brown

EurIPS 2025 Workshop: AI for Tabular Data.



(a) Skipping layers (b) Swapping layers (c) Repeating layers (d) Probing Classifier

Figure 1: Layer reorganization in tabular ICL models following the "Layers as Painters" framework.

et al., 2020]. To satisfy this need, most ICL models are directly trained to approximate Bayesian inference on synthetically generated predictive tasks. *TabPFN(v1)* [Hollmann et al., 2023] was the first model operating in such fashion on tabular data, where the backbone is a vanilla transformer. *TabPFN(v2)* [Hollmann et al., 2025] improves by adding an attention mechanism within the tokens in addition to cross-tokens attention. *TabICL* [Qu et al., 2024] shares a similar backbone with *TabPFN(v2)*, but additionally introduces a transformer-based compression that efficiently transforms rows into semantically rich embeddings. Specifically, *TabICL* employs a two-stage architecture. The first stage, "tabular embedding", encodes table rows into dense vector representations while explicitly accounting for the inherent column–row structure through column-wise and row-wise interactions. The second stage, "ICL prediction", uses these embeddings along with their corresponding labels to make predictions.

LLM's hidden state dynamics is widely studied. Zhang et al. [2024] propose a framework to study the contribution of the individual layers by performing an ablation study on layer exclusion via an adapted framework of Shapley values. Results suggest that early layers, called "cornerstone" layers, process initial embedding into the space where subsequent "non-cornerstone" layers operate. Such "non-cornerstone" layers can overlap in function, and their individual contribution is not significant. Gromov et al. [2025] makes an argument that deeper layers do not store knowledge, but rather are utilized for the computations, e.g., reasoning and dealing with long context. Similar to prior work, they studied model performance by ablating model layers and optionally fine-tuning. Further works show that it is possible to improve computational complexity and/or performance by modifying the flow of the embedding in the model [Heakl et al., 2025, Laitenberger et al., 2025, Li et al., 2025]. **Layers as Painters** is a particular framework by Sun et al. [2025] that inspired our work. It was originally designed to verify the hypothesis that different layers express shared representational "languages" by operating in a common representation space and complement each other's output. This contrasts with operating in a hierarchical feature space, similar to models with bottlenecks such as convolutional networks. If this holds, then the sequence of the trained layers can be reordered as illustrated in Figure 1 without catastrophic loss of performance.

Lastly, we complement this framework with **probing classifiers** [Belinkov, 2022], where a classification model is trained on an embedding produced by a model (or intermediate stage) that is studied to infer some (e.g., linguistic) quantity, as illustrated in Figure 1d. The performance of this probing classifier gives a rough estimation of the mutual information between the embedding and the quantity of interest. For *TabPFN(v2)*, a fine-tuned decoder can be used to early-exit the forward pass and reduce inference time while maintaining performance [Küken et al., 2025], suggesting that not all layers are equally important; however, it remains open to what extent this transfers across models. Additionally, Ye et al. [2025] analyzed the embeddings of each layer for *TabPFN(v2)* using PCA.

3 Methodology and Experiment Design

In this section, we formulate the objectives of our structured analysis and describe each of the experiments. Namely, we focus on three guiding questions:

Q1: Do all layers speak the same language? We analyze whether the application of the layers is commutative or whether there is hierarchy in the representations. First, to test representational alignment, we swap the order of layers and measure the performance of the forward pass, as shown in Figure 1b. If switching layers degrades performance, the representation might have a hierarchy. Secondly, we perform probing with linear classifiers trained on the representation from the same and other layers. Representations are similar if a trained probing classifier performs well on another layer's embedding. Thirdly, we study repeating layers as shown in Figure 1c. If repeating a layer increases performance, it might suggest that the layer performs iterative refinement, e.g., as in

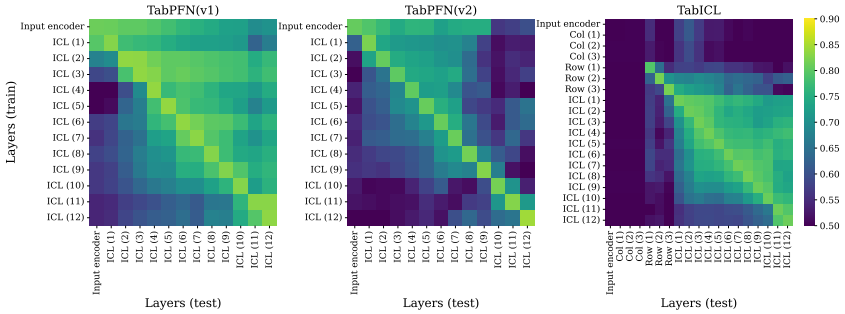


Figure 2: Average AUC for probing classifiers (logistic regression) trained on embeddings at different layers of the models.

recurrent architectures. This also means that the layers' in- and output embedding spaces must be closely aligned for the layer to operate in a recurrent manner [Dehghani et al., 2019].

Q2: Does the model use all layers? To study this, we skip layers in the forward pass (see Figure 1a) and measure the impact on downstream performance. A negligible change in performance might indicate that a layer is redundant (or its contribution is not required to solve the task).

Q3: How consistent are findings across models? Finally, to assess whether our findings are specific to a single instantiation of a tabular ICL model or apply to multiple models, we repeat the analysis for TabPFN-v1, TabPFN-v2, and TabICL.

Models and Datasets: We run experiments on a subset of TabArena comprising 15 binary classification tasks [Erickson et al., 2025] (see Appendix A). We use *TabPFN(v1)*, *TabPFN(v2)*, and *TabICL* as tabular ICL models. We use ROC-AUC, averaged across datasets, as the evaluation metric.

Classifier probing. For this task, we use three different models: logistic regression, K-Nearest Neighbors (KNN), and a fine-tuned MLP from the respective ICL model's decoder layer. We begin by extracting embeddings for each layer from its hidden states in the ICL setup. Specifically, we train and evaluate the probing classifier only on the embedded query set. For this experiment, we extend the query set by including half of the original training set (excluded from the support set) to serve as training data for the probing classifier.

4 Experimental Results

Q1: Some layers do speak the same language. The **probing** results in Figure 2 indicate that while the outcomes are highly model-dependent, a consistent pattern emerges: A probing classifier trained on layer i performs well on the embeddings of a later layer $j > i$, whereas the reverse is not true. This suggests that later layers still contain information from earlier layers, and that new features emerge in higher layers not present in earlier ones. This is particularly pronounced for *TabPFN(v2)*. For *TabPFN(v1)*, this is only noticeable for early layers, whereas for *TabICL*, this behaviour is least pronounced. These results support the "layers as painters" hypothesis that the middle layers share a similar representational space, as probes trained at different layers can operate interchangeably. In Appendix B.1, we also provide concise similarity analyses between the embeddings of each layer and include the results of using a KNN classifier and a fine-tuned model decoder as probes. Figure 3 shows the effect of **layer swapping**. As expected, performance degrades most in the first (or early layers), indicating that layer order is important. Interestingly, for *TabPFN(v2)*, there is a significant loss in the middle layers, suggesting that while these layers may share a similar representation space, they perform different operations. Finally, we study **repeating individual layers** in Figure 4. Interestingly, and in contrast to LLMs, where repeating a single layer is highly detrimental [Sun et al., 2025], in tabular models, repeating a layer seems less harmful or can even improve performance for some tasks (see Figure 10).

Q2: Yes, some layers can be skipped. Figure 5 shows that skipping early layers of the ICL prediction stage of *TabICL* and *TabPFN(v1)* impacts performance most. This means that earlier layers are more important for final performance than later ones (and final layers can even be dropped with minimal loss). This is in line with findings from the LLM literature [Zhang et al., 2024, Sun et al., 2025]. Results are different for *TabPFN(v2)*; multiple intermediate layers appear essential

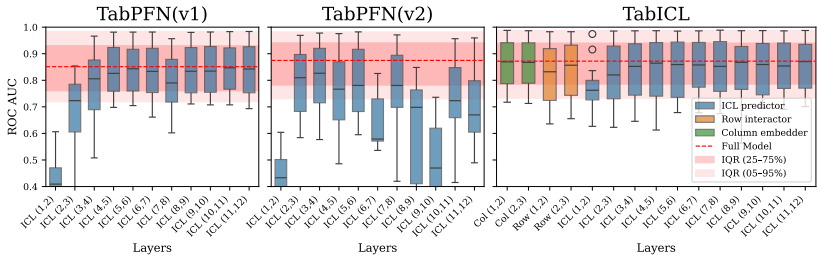


Figure 3: Impact on average AUC when swapping two layers in the forward pass of the models.

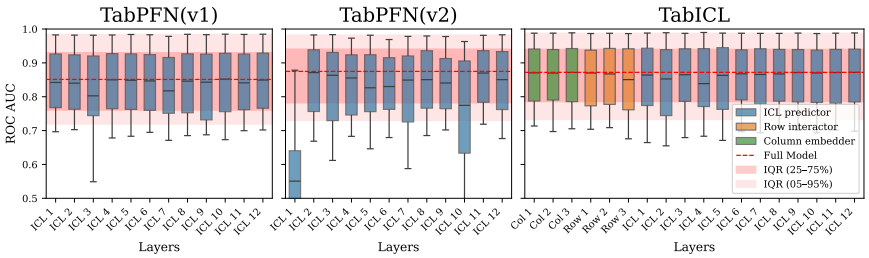


Figure 4: Impact of repeating the layers of the models.

for final performance. Prompted by these findings, we test early exiting like Küken et al. [2025] in Appendix B.3. Our results indicate that for *TabPFN(v1)* and *TabICL*, early exit is feasible without fine-tuning the decoder components. In contrast, *TabPFN(v2)* requires an individually fine-tuned decoder, as also observed by Küken et al. [2025].

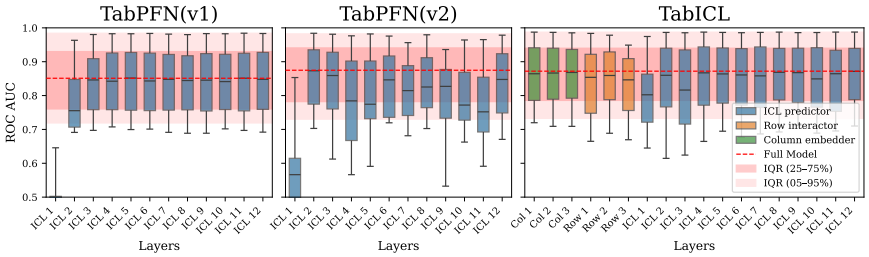


Figure 5: Impact on average AUC when skipping a layer during the forward pass.

Q3: Some common patterns do emerge; the results vary greatly. Our results suggest that early layers have the most impact for all models. Unlike in LLMs, *TabPFN(v2)* seems to have less redundancy according to our probing and layer skipping results. Also, repeating a layer often neither degrades nor improves performance, suggesting that the layer has learned to modulate its contribution to the output as needed. Overall results on specific layers are model-specific. Additionally, in Appendix B.2, we also provide win–tie–lose results for our experiments, which suggest that performance changes caused by each layer reorganization operation are also task-dependent.

5 Conclusion and Discussion

Our analysis highlights fundamental differences between tabular ICL models and LLM layer dynamics. We find that not all layers in tabular ICL models contribute equally, with some redundancy across depth. Specifically, we observed that later layers in *TabICL* and *TabPFN(v1)* are not critical to the final performance. We note that our evaluation averages effects over datasets using a single fold, repetition, and model initialization, and our layer-reorganization experiments modify only one layer at a time. However, even within these limitations, our empirical study raises promising questions for future research: How stable are these observations across (1) tasks and (2) initializations [Wang et al., 2018]? And more broadly, can these insights guide the development of (3) lightweight and (4) interpretable tabular ICL models?

Acknowledgments and Disclosure of Funding

This research has been funded by the Federal Ministry of Research, Technology and Space of Germany and the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence. Additionally, this research utilized compute resources at the Tübingen Machine Learning Cloud, DFG FKZ INST 37/1057-1 FUGG. A. Balef and M. Koshil also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS).

References

N. Erickson, L. Purucker, A. Tschalzev, D. Holzmüller, P. M. Desai, D. Salinas, and F. Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Curran Associates, 2025.

A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. Roberts. The Unreasonable Ineffectiveness of the Deeper Layers. In *The Thirteenth International Conference on Learning Representations (ICLR'25)*. ICLR, 2025.

Q. Sun, M. Pickett, A. K. Nain, and L. Jones. Transformer layers as painters. In *Proceedings of the Thirty-Eighth Conference on Artificial Intelligence (AAAI'25)*. Association for the Advancement of Artificial Intelligence, AAAI Press, 2025.

N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations (ICLR'23)*. ICLR, 2023.

N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

J. Qu, D. Holzmüller, G. Varoquaux, and M. Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, volume 251 of *Proceedings of Machine Learning Research*. PMLR, 2024.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, pages 1877–1901. Curran Associates, 2020.

Y. Zhang, Y. Dong, and K. Kawaguchi. Investigating layer importance in large language models. In *The 7th BlackboxNLP Workshop*, 2024. URL `https://openreview.net/forum?id=kjZIIvFtmK`.

A. Heakl, M. Gubri, S. Khan, S. Yun, and S. J. Oh. Dr.LLM: Dynamic layer routing in LLMs. *arXiv preprint*, 2025. URL `https://arxiv.org/abs/2510.12773`.

F. Laitenberger, D. Kopiczko, C. G. M. Snoek, and Y. M. Asano. What layers when: Learning to skip compute in LLMs with residual gates. *arXiv preprint*, 2025. URL `https://arxiv.org/abs/2510.13876`.

Z. Li, Y. Li, and T. Zhou. Skip a layer or loop it? test-time depth adaptation of pretrained LLMs. *arXiv preprint*, 2025. URL `https://arxiv.org/abs/2507.07996`.

Y. Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 2022. doi: 10.1162/coli_a_00422. URL `https://doi.org/10.1162/coli_a_00422`.

J. Küken, L. Purucker, and F. Hutter. Early stopping tabular in-context learning. In *1st International Workshop on Foundation Models for Structured Data (FMSD) @ ICML 2025*, 2025.

H. J. Ye, S. Y. Liu, and W. L. Chao. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. In *Proceedings of the 38th International Conference on Advances in Neural Information Processing Systems (NeurIPS'25)*. Curran Associates, 2025.

M. Dehghani, Ç. Gülçehre, Y. Bengio, et al. Universal transformers. In *The Seventh International Conference on Learning Representations (ICLR'19)*. ICLR, 2019.

L. Wang, L. Hu, J. Gu, Z. Hu, Y. Wu, K. He, and J. Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. In *Proceedings of the 31st International Conference on Advances in Neural Information Processing Systems (NeurIPS'18)*. Curran Associates, 2018.

A Datasets

Table 1 lists all datasets used in our experiments.

Table 1: Dataset.

index	task id	dataset name	#samples	#features	#categorical features
1	363619	Bank-Customer-Churn	10000	11	5
2	363621	blood-transfusion-service-center	748	5	1
3	363623	churn	5000	20	5
4	363624	coil2000-insurance-policies	9822	86	4
5	363626	credit-g	1000	21	14
6	363629	diabetes	768	9	1
7	363671	Fitness-Club	1500	7	4
8	363674	hazelnut-spread-contaminant-detection	2400	31	1
9	363682	Is-this-a-good-customer	1723	14	9
10	363684	Marketing-Campaign	2240	26	9
11	363689	NATICUSdroid	7491	87	87
12	363694	polish-companies-bankruptcy	5910	65	1
13	363696	qsar-biodeg	1054	42	6
14	363700	seismic-bumps	2584	16	4
15	363706	taiwanese-bankruptcy-prediction	6819	95	1

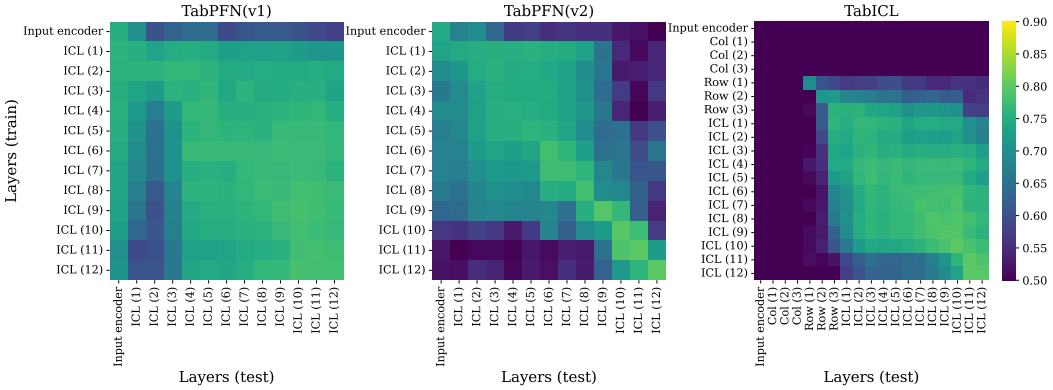


Figure 6: Probing with KNN trained on embedding from different layers of the models.

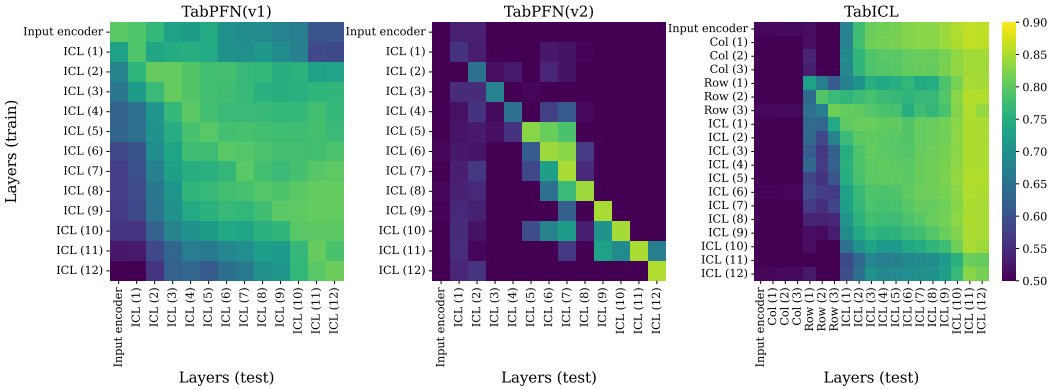


Figure 7: Probing with model decoder fine-tuned on embedding from different layers of the models.

B More results

B.1 Probing

We provide more results from linear probing using different classifiers. Figure 6 shows the performance of KNN trained on embeddings extracted from different layers of the models, while Figure 7 presents results from model decoders fine-tuned on the same embeddings.

We also report the cosine similarity between embeddings from different layers of the models, as shown in Figure 8.

B.2 Layer reorganization

Here, we provide win–tie–lose comparison plots against the full model evaluation to examine how architectural changes in the transformer affect performance across tasks. We adopt a tie threshold of 2×10^{-4} to ensure that numerical precision and minor fluctuations do not result in spurious wins or losses. Figure 9 shows the effect of skipping layers, Figure 10 illustrates the impact of repeating layers, and Figure 11 presents the results of swapping layers.

B.3 Early exit

We perform an early-exit strategy, meaning that after each layer, we pass the embeddings to the decoder. However, we did not fine-tune the decoder layer in contrast to Küken et al. [2025]. As observed in Figure 12 and Figure 13 for *TabPFN(v2)*, having an individual decoder is necessary, as shown by Küken et al. [2025], whereas for other models, the performance degradation is not drastic.

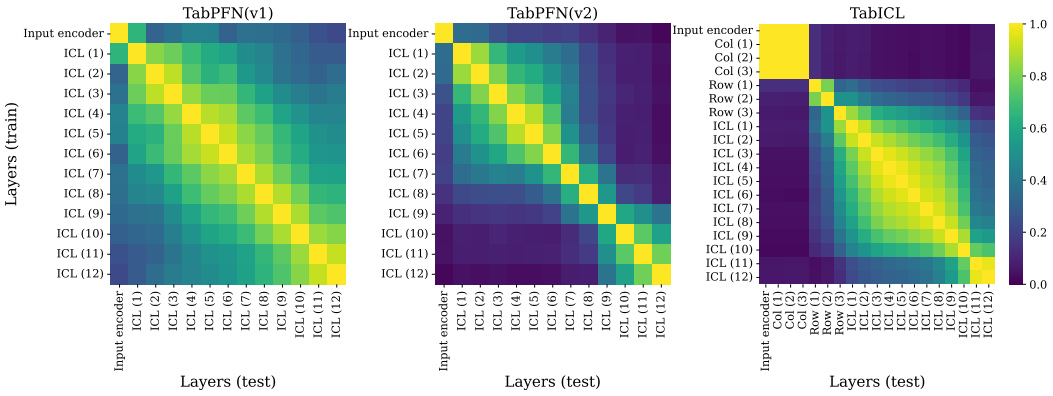


Figure 8: Cosine similarity of embeddings extracted from different layers of the models.

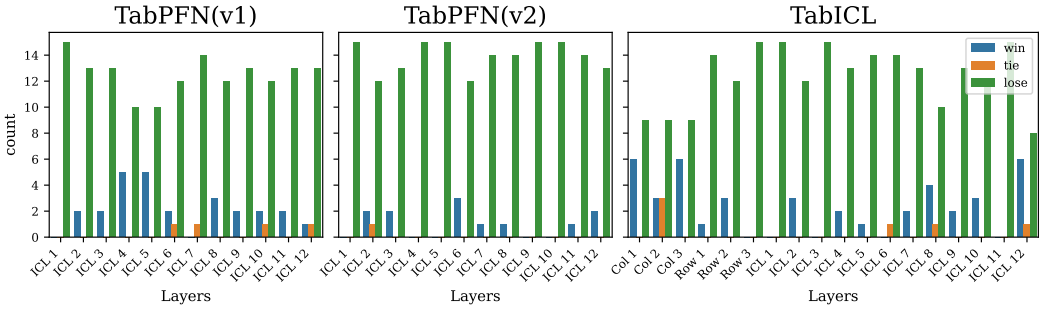


Figure 9: Effect of skipping layers in the tabular foundation model's architecture.

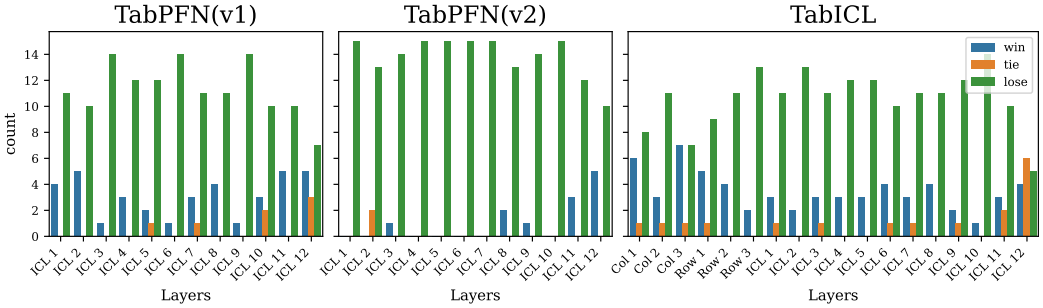


Figure 10: Effect of repeating layers in the tabular foundation model's architecture.

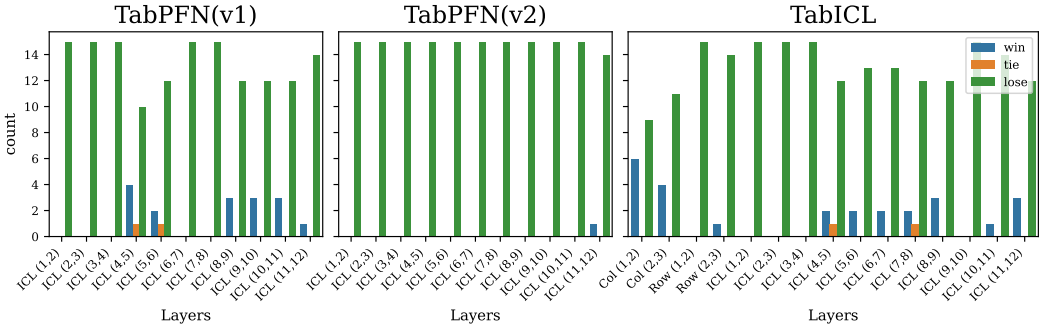


Figure 11: Effect of swapping layers in the tabular foundation model's architecture.

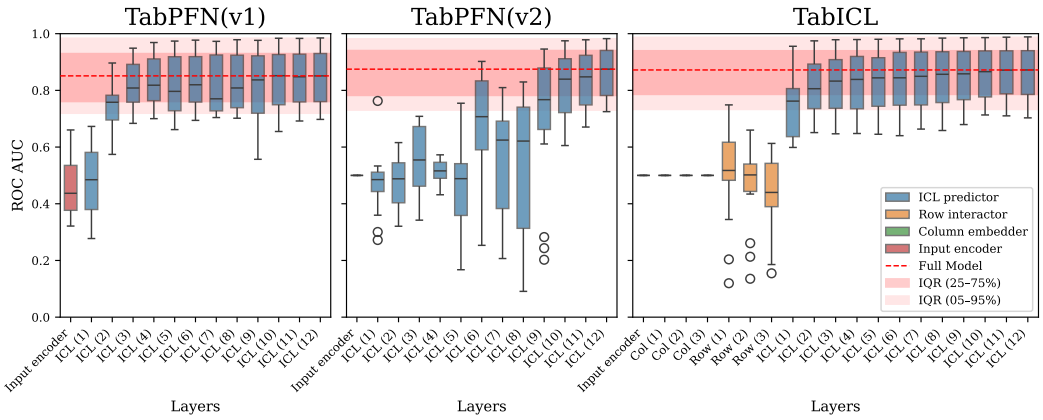


Figure 12: Effect of the early exit strategy on the tabular foundation model's performance.

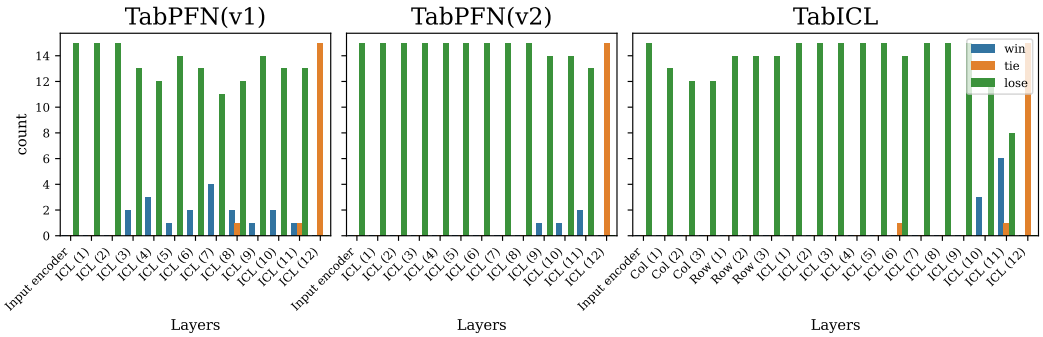


Figure 13: Comparison of performance with and without the early exit strategy in the tabular foundation model.