How Much Pretraining Does Structured Data Need?

Daniel Fadlon daniel.fadlon@post.runi.ac.il
Kfir Bar kfir.bar@runi.ac.il
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel
Data Science Institute, Reichman University, Herzliya, Israel

Abstract

Large language models (LLMs) are increasingly adopted for handling structured data, including tabular and relational inputs, despite mostly being pretrained on unstructured text. This raises a key question: *how effectively do pretrained representations from language-focused LLMs transfer to tasks involving structured inputs?* We address this through controlled experiments using two small open-source LLMs, systematically re-initializing subsets of layers with random weights before fine-tuning on structured datasets and comparing results to unstructured datasets. Our analyses show that, for structured data, most pretrained depth contributes little, with performance often saturating after the first few layers, whereas unstructured tasks benefit more consistently from deeper pretrained representations. Pretraining remains useful mainly in low-resource settings, with its impact diminishing as more training data becomes available.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language understanding and generation tasks. Their success is primarily attributed to large-scale pre-training on massive corpora of unstructured text, which equips them with rich linguistic and semantic representations. However, many real-world applications, such as medical diagnostics from laboratory records, financial forecasting from transaction tables, and recommendation systems from user profiles, require reasoning over structured data. This raises a fundamental question: *to what extent do the pre-training representations of standard LLMs, mostly optimized for unstructured language, transfer to downstream tasks that involve structured inputs?*

A growing body of research has explored developing specialized models for tabular and structured data. Notable examples include TabLLM (Hegselmann et al., 2023) and StructGPT (Jiang et al., 2023) and related approaches, which explicitly train models explicitly to consume structured inputs such as tables or databases, often achieving strong performance in tasks like clinical decision support or financial analysis. These works treat the structured nature of the data as first-class, designing pre-training objectives and architectures to accommodate strict typing, relational dependencies, and schema alignment. At the same time, researchers increasingly employ general-purpose language models for prediction over structured inputs, motivated by the observation that many feature values and names carry linguistic meaning. For instance, in clinical settings, a value such as "shortness of breath" in a symptom column or "metformin" in a medication field already encodes rich semantic content that LLMs can interpret through their pre-training. Similarly, feature names like "heart rate" or "blood pressure" convey meaningful relations that can guide the model's reasoning without requiring explicit schema encoding. Complementary lines of work focus on hybrid reasoning over structured and unstructured sources: GMELLO (Chen et al., 2024) integrates knowledge graphs for multi-hop question answering, SUQL (Liu et al., 2024) enables unified querying over structured data and free text, and LLAMACARE (Li et al., 2024) converts structured electronic health records into textual representations for downstream language-model training.

Therefore, in our work we consider standard LLMs, originally pre-trained mostly on unstructured text, and ask whether their internal representations are inherently useful for structured data tasks once appropriate adaptations are made. More specifically, we investigate how different transformer-based (Vaswani et al., 2017) LLM layers contribute to performance when models are fine-tuned on both structured and unstructured downstream tasks.

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 3352–3365

We operationalize "structured data" as datasets composed of records organized into tables of typed features (e.g., credit histories, sensor readings). To make these inputs compatible with text-based LLMs, we adopt a verbalization strategy, converting records into natural language sequences. We then fine-tune LLMs on classification tasks, and contrast these results with non-generative tasks involving unstructured text, chosen to stay as comparable as possible in format and complexity.

Inspired by earlier work (Tamkin et al., 2020), our methodology involves systematically altering the model's initialization: keeping a subset of layers from the original pre-training untouched while randomizing the rest, and then fine-tuning the model under these different conditions. By measuring performance and applying linear probing to hidden states, we analyze whether pre-trained layers encode information beneficial to structured tasks, and how these benefits compare to unstructured ones. We focus on relatively small generative language models (SLMs, up to 1 billion parameters), as the target tasks in this study are non-generative and do not require large-scale text generation capabilities.

Our experiments reveal an important insight: most of the layer weights in a pre-trained SLM, while highly beneficial for unstructured text tasks, appear to contribute little to structured data tasks. In fact, they can often be discarded without loss in performance when fine-tuning for such tasks.

To summarize, this study is guided by the following research questions:

RQ1: Are the weights of standard generative SLMs, created by the pre-training phase, beneficial for fine-tuning the model on downstream tasks that involve structured data inputs?

RQ2: What effect does reducing the size of training sets have on the performance of a standard SLM, when fine-tuned on downstream tasks involving structured data compared to those involving unstructured data?

RQ3: How do the contributions of individual layers in a fine-tuned SLM behave when handling structured data, and how do these behaviors differ from those observed in unstructured data tasks?

The rest of the paper is organized as follows. We first describe the methods used in our study, followed by the experiments addressing the three research questions. Next, we discuss related work and conclude with a discussion of our findings.

2 Methods

In this section, we outline the experimental framework used to investigate how relatively small generative language models (LMs) handle structured data tasks compared to unstructured, textual ones.

We design a series of experiments to isolate the contribution of pretrained representations, analyze the role of model depth, and evaluate how performance scales with dataset size. Our analysis combines both representational probing and controlled weight re-initialization to capture complementary aspects of model behavior. The following sections outline the techniques used in our experimental design.

2.1 Random Re-initialization

To quantify the contribution of pretrained weights, we conduct a controlled layer re-initialization analysis. Given a transformer-based (Vaswani et al., 2017) LM with X layers, we train variants where the top (i.e., closer to the classification head) y layers ($0 \leq y \leq X$) are re-initialized with random values, while the remaining layers retain their pretrained values. At one extreme ($y = 0$), the model remains entirely pretrained; at the other ($y = X$), it is trained from scratch. Re-initialization is applied to all parameters within the selected layer, including those of both the attention and feed-forward subcomponents. A standard transformer-based LM consists of an embedding layer, a stack of transformer layers, and a final unembedding or classification head. We re-initialize only the transformer layers, keeping both the embedding and unembedding layers fixed, since our goal is to isolate the effect of pretraining in the contextual layers rather than in static token embeddings. The unembedding weights are typically tied to the embedding matrix, using its transpose, as shown by Press and Wolf (2017).

By systematically varying y and comparing downstream task performance, we isolate which layers contribute most critically to fine-tuning success on structured versus unstructured data.

Random re-initialization is drawn from a normal distribution with zero mean and a standard deviation matching that of the original pretrained layers in the same model.

Type	Dataset	Train	Test
Structured	Bank Marketing	36,168	9,043
	California Housing	16,512	4,128
	German Credit	800	200
	Adult Income	39,074	9,768
	Heart Disease	734	184
Unstructured	SNLI (Bowman et al., 2015)	549,367	9,842
	SST-2 (Socher et al., 2013)	67,349	1,821
	TREC (Voorhees and Tice, 2000)	5,452	500

Table 1: Datasets used in this study. Structured datasets are adapted from the TABLLM framework, consisting of tabular binary classification tasks. Unstructured datasets are drawn from the GLUE benchmark and evaluate linguistic understanding.

2.2 Datasets

We use two types of datasets: structured tabular data and unstructured textual data. The complete dataset statistics are presented in Table 1. License information is provided in Appendix E.

Structured data. Inspired by prior research on language modeling for tabular data, we follow the experimental setup of the TABLLM framework and adopt its collection of datasets, which consist of binary classification tasks designed to evaluate model performance on structured tabular inputs. Following previous surveys (Kadra et al., 2021; Grinsztajn et al., 2022; Borisov et al., 2023), we select five datasets that include textual feature names suitable for natural language serialization: **Bank Marketing** (45,211 rows; 16 features), **California Housing** (20,640; 8), **German Credit** (1,000; 20), **Adult Income** (48,842; 14), and **Heart Disease** (918; 11). We adopt the same prompt templates and serialization strategies as in TABLLM for consistency and comparability.

Unstructured data. To contrast with structured tasks, we also evaluate the models on three unstructured datasets from the GLUE benchmark (Wang et al., 2019): **SNLI** (Bowman et al., 2015) for natural language inference, **SST-2** (Socher et al., 2013) for sentiment analysis, and **TREC** (Voorhees and Tice, 2000) for question type classification. These datasets are substantially larger than the tabular ones and probe semantic and linguistic understanding, offering a complementary perspective on layer effectiveness and the utility of pretrained weights.

2.3 Evaluation

We use two complementary evaluation perspectives to assess both task performance and representational quality.

Evaluation with ROC-AUC and accuracy. For structured data, we report the Area Under the ROC Curve (ROC-AUC) as the primary metric, following standard practice in tabular-data learning, since it captures performance across decision thresholds. For textual benchmarks, we use accuracy as the main metric, which directly reflects classification correctness. When referring to *performance* in the results, we report ROC-AUC for structured datasets and accuracy for unstructured datasets.

Evaluation with probing. Beyond task-level metrics, to address RQ3, we examine how information is internally represented across layers. For each intermediate layer's hidden state, we extract activations and train lightweight classifiers—linear logistic regression—to predict the gold labels, following standard probing methodologies (Conneau et al., 2018; Tenney et al., 2019b). This layer-wise probing approach provides a fine-grained view of how structured information is encoded at different depths, revealing which layers capture the most discriminative representations and whether redundant encoding occurs in deeper layers.

3 Experiments

We begin by describing the general experimental setup and then present the experiments designed to address the three research questions outlined above.

3.1 Experimental Settings

All models are fine-tuned on the provided training splits and evaluated on validation and test sets. Training runs for up to ten epochs with early stopping based on validation-set performance, measured after each epoch. Each experiment is repeated with three random seeds for robustness. For layer-wise evaluation, we employ a regularized Lo-

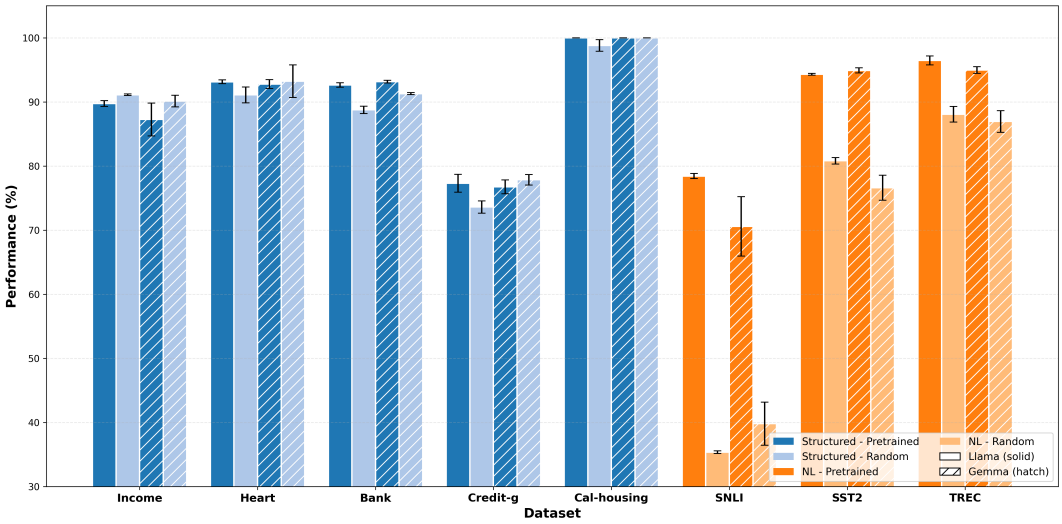

Figure 1: Comparison of pretrained models vs. randomly re-initialized models. Blue bars (left) correspond to structured datasets, while orange bars (right) correspond to unstructured datasets. Dark shades indicate *pretrained* models, and light shades indicate *randomly re-initialized* models. Each block contains four bars: the two leftmost solid bars represent the pretrained and randomly re-initialized variants of the `Llama-3.2-1B-Instruct` model, and the two rightmost hatched bars represent the same corresponding variants of `Gemma-3-1B`.

gistic Regression (LoR) probe trained using 5-fold cross-validation. Implementation details, including optimizer settings, batch sizes, prompt templates, and the LoR configuration, are provided in Appendix A.

We employ two recent small generative language models, `Llama-3.2-1B-Instruct` (Touvron et al., 2024) and `Gemma-3-1B`(Mesnard and the Gemma Team, 2025), as representative architectures for analyzing the role of pretraining in transfer to structured data tasks. Both are open-weight, transformer-based models that preserve the design principles of their larger counterparts while remaining computationally efficient for extensive probing, layer truncation, and controlled fine-tuning. Their compact size enables controlled and interpretable experiments without the confounding effects of scale, making them ideal for isolating the representational contribution of pretrained weights to structured-data tasks. More information about why we choose to experiment with these two models is provided in Appendix B. Model license information is provided in Appendix E.

Re-initialization is fully random and consistent per seed. Our code and full experimental configurations are publicly available.[1]

[1] `https://github.com/DanielFadlon/pretraining-for-structured`

3.2 RQ1: Importance of Pretrained Weights in Fine-Tuning on Structured Data

To isolate the contribution of pretraining from architectural or capacity-related factors, we compare two variants of the same model architecture: (1) a fully pretrained version and (2) an identical model whose parameters are entirely re-initialized with random weights. We fine-tune each model variant on all downstream tasks and evaluate its performance across the two data type categories, comparing the results to analyze their relative effectiveness. By contrasting their performance, we can quantify the specific advantage conferred by pretraining and determine whether the observed gains arise from learned representations or merely from the model's architectural expressiveness.

Figure 1 presents the results across both structured and unstructured datasets. As illustrated, the performance gap between the pretrained and randomly initialized models is notably small for structured datasets. In several instances, the model trained from random initialization performs comparably to, or even exceeds the pretrained version. These results indicate that, for structured data, a substantial portion of the model's effectiveness stems from its architectural design and fine-tuning process rather than from the linguistic knowledge acquired during pretraining.

In contrast, for unstructured datasets, the gap remains pronounced, emphasizing the continued

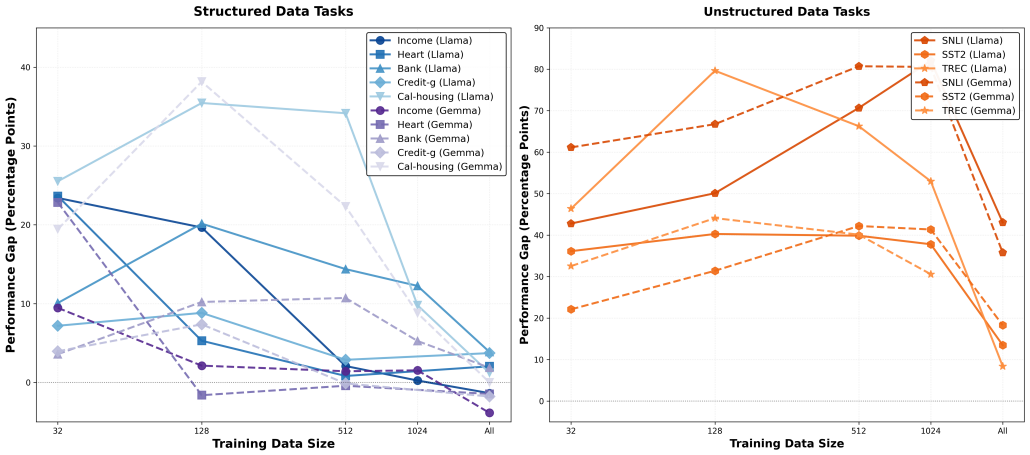

Figure 2: Performance gap (in percentage points) between pretrained and randomly re-initialized model variants across varying training set sizes. Solid lines denote `Llama-3.2-1B-Instruct`; dashed lines denote `Gemma-3-1B`. Markers correspond to individual tasks (e.g., SST-2). Positive values indicate a gain from pretraining. Structured data results are shown on the *left*; unstructured data on the *right*.

importance of pretrained representations even after extensive fine-tuning with a relatively large number of examples. However, for structured datasets, across tasks that vary in training set size, from the larger Bank Marketing dataset (45, 211 rows, 16 features) to the smaller Heart Disease dataset (918, 11), this advantage largely disappears, revealing a fundamental difference in how pretraining effects carry over across data types.

To conclude RQ1, our results indicate that for structured datasets, pretrained weights contribute little, and in some cases not at all, to the overall performance of the model. However, this finding is unlikely to generalize to all settings. In many real-world transfer learning scenarios, the amount of training data is limited or partially observed, conditions in which pretrained knowledge may become substantially more valuable. This leads to our next question: *how does the amount of training data affect the relative advantage of pretraining?* We explore this in the following section.

3.3 RQ2: Impact of Training Set Size

To study whether the impact of pretraining on models fine-tuned for structured data tasks depends on the amount of training data available, we perform a controlled scaling analysis. For each structured dataset, we fine-tune both the pretrained and the randomly re-initialized variants on smaller subsets of the training data. This design allows us to assess whether pretrained representations become more beneficial when data is limited, as commonly hypothesized in transfer learning.

Like before, we repeat each run three times, sampling a different training subset from the full dataset while maintaining a consistent set of random seeds across tasks. In other words, we use three distinct seeds, each determining one training subset of the target size for every dataset.

Figure 2 shows the performance difference, measured in percentage points, between the pretrained model variant and the randomly re-initialized variant, across different training-set sizes for `Llama-3.2-1B-Instruct` and `Gemma-3-1B`. Since each experiment is repeated three times, we compute the mean performance across the three runs for each model variant and then calculate the difference between these two mean values. For absolute performance numbers see Appendix C.

When trained on the full dataset (appears as All in the figure), the two variants trained on structured datasets achieve similar results, consistent with our findings in RQ1. However, as the training set size decreases, a modest but consistent advantage for the pretrained model emerges. The difference is most pronounced at intermediate sizes (e.g., 128–512 samples), where the pretrained model leverages prior knowledge to generalize better from limited examples.

Although the absolute numbers are not shown in the figure, at the smallest data scale (32 samples), both model variants perform poorly, suggesting that neither architecture nor pretraining can compensate for extreme data scarcity. Conversely, when the data size increases to 1,024 samples or more, the gap narrows substantially—indicating that as

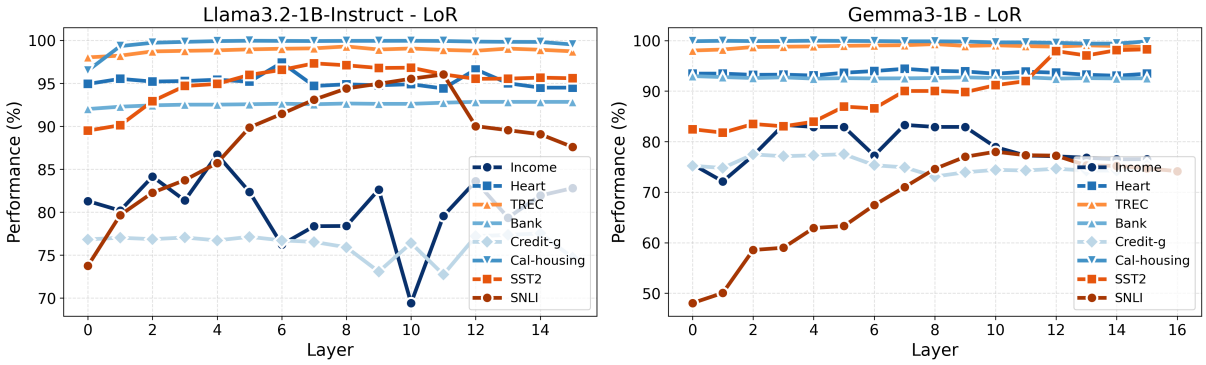

Figure 3: Layer-wise probing performance for the pretrained `Llama-3.2-1B-Instruct` model (*left*) and `Gemma-3-1B` (*right*) on structured and unstructured tasks using Logistic Regression probes.

the model gains sufficient exposure to structured examples, the architectural capacity alone becomes sufficient to capture the task structure.

For unstructured datasets, a somewhat different pattern emerges. The gaps between the pretrained and randomly re-initialized variants remain substantial across all training set sizes, though they decrease sharply when the full training data is used. It is important to note that the final curve segments, from 512 samples to the full dataset, are not directly comparable across datasets because their total training sizes differ. Nevertheless, these results indicate that pretrained weights consistently benefit models fine-tuned on unstructured data, particularly when training data is limited, but also to a meaningful extent even when large training sets are available.

To conclude RQ2, our findings show that for structured datasets, pretraining provides a clear advantage mainly in low-resource settings, while its contribution becomes marginal as the amount of supervised training data increases.

3.4 RQ3: Layer Analysis

Building on the evidence that pretrained representations provide useful information for tasks involving structured data, we next examine where these effects are concentrated within the model. Specifically, we analyze which layers carry the most impactful weights and how this pattern differs between structured and unstructured data settings. This comparison allows us to observe how pretraining influences the fine-tuning dynamics and representation depth across domains.

Probing layer representations. To investigate where pretrained representations contribute most to structured data performance, we first probe each layer's activation vector of the pretrained model in a zero-shot setting. This analysis measures the representational quality of each hidden layer, that is, its ability to capture informative latent features relevant to the target task. Figure 3 comprises two subfigures, each illustrating the probing results obtained via LoR for a different model: `Llama-3.2-1B-Instruct` on the left and `Gemma-3-1B` on the right. Both show a similar trend, where unstructured tasks (different shades of orange) exhibit a steadily improving pattern across layers, while structured tasks (different shades of blue) display an almost flat line. This indicates that deeper layers contribute little additional value for structured inputs, suggesting that most of the useful inductive bias is already captured in the lower layers.

Controlled layer re-initialization. To further validate these observations, we conduct a controlled experiment in which we progressively re-initialize higher layers (closer to the classification head) while keeping a fixed number of lower layers pretrained. Figure 4 shows the fine-tuned performance as a function of the number of preserved pretrained layers, ranging from 0 to 16, for both model architectures used in this study. For instance, a value of 4 on the x-axis indicates that the model retains the original pretrained weights for the first four layers, while layers 5 through 16 are randomly re-initialized. The results align with the probing analysis: in unstructured tasks (different shades of orange), each additional pretrained layer yields a gradual improvement in accuracy, though for TREC this improvement plateaus earlier. In contrast, for structured tasks (different shades of blue), the benefit of additional pretrained layers saturates almost immediately after the first layer. In many cases, models retaining only the first or

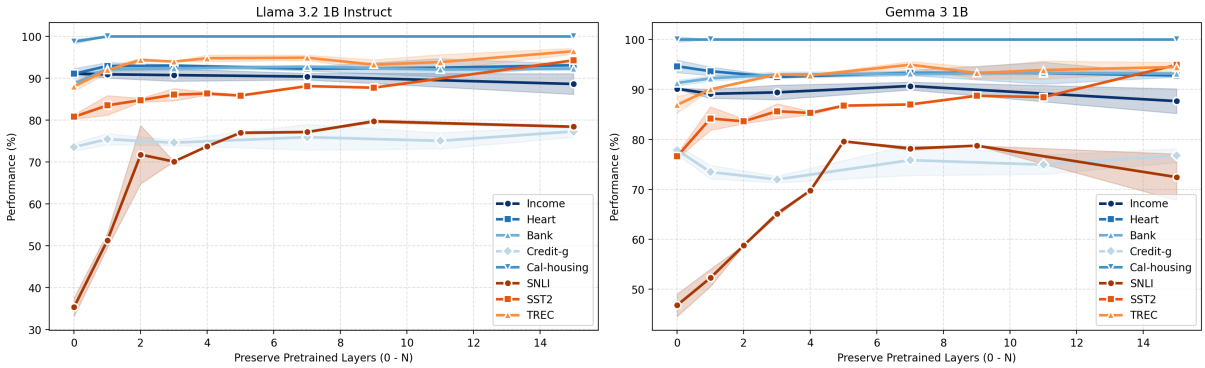

Figure 4: Performance (percentage points) of a randomly re-initialized model variants across different numbers of preserved layers, those are, the layers that retain their pretrained weights. The results are for `Llama-3.2-1B-Instruct` (*left*) and `Gemma-3-1B` (*right*). Some additional information about this set of experiments is provided in Appendix F.

second pretrained layer perform comparably to, or even slightly better than, the fully pretrained model (represented by 16 on the x-axis).

Layer truncation. Finally, to isolate whether this early-layer effectiveness stems from its interaction with deeper transformer blocks or from genuinely informative weights, we repeat the experiment using truncated models containing only one transformer layer connected directly to the language modeling head. As shown in Figure 5, these lightweight model configurations achieve performance comparable to the full model, confirming that the most impactful pretrained representations for structured data reside within the very first layer.

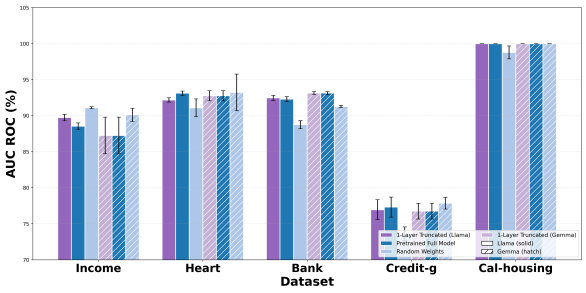

Figure 5: Performance comparison when fine-tuning only the first transformer layer (1-layer-truncated). Each block shows `Llama-3.2-1B-Instruct` results on the *left* (solid bars) and `Gemma-3-1B` results on the *right* (hatched bars). Purple shades represent 1-layer-truncated variants, dark blue indicates pretrained models, and light blue corresponds to randomly initialized weights.

To conclude RQ3, our findings highlight a clear distinction between structured and unstructured datasets. In unstructured tasks, knowledge builds progressively through deeper layers, while in structured data tasks, nearly all transferable information resides in the lower layers. This suggests that future model architectures or adaptation strategies for structured data may benefit more from optimizing the lower layers rather than depending on deep pretrained stacks.

4 Related Work

The growing body of research on transfer and adaptation in language models has revealed that pretrained representations are not uniformly useful across layers, domains, or data modalities. Our work builds on this line of inquiry by examining how and where pretrained knowledge persists when language models are fine-tuned on structured data.

4.1 Pretrained Models on Structured Data

Recent efforts have begun testing the adaptability of pretrained LLMs to tabular or otherwise structured inputs. Rabbani et al. (2025) show that fine-tuning GPT-based models on serialized tables can outperform traditional machine-learning baselines (i.e., XGBoost), implying that pretraining on text gives these models a broad ability to capture patterns useful for non-textual, structured data as well. Structure-aware parameter-efficient approaches for fine-tuning, such as `TableLoRA` introduced by He et al. (2025), further demonstrate that explicitly encoding table geometry in LoRA adapters improves performance, suggesting that structural awareness, rather than raw language exposure, drives much of the gain. Extending this idea, Rubachev et al. (2025) and Garg et al. (2025a) adapt the `TabPFN` family through continual pretraining on real tables, finding that better alignment between pretraining and downstream data modalities yields consistent

improvements. Together, these studies establish that pretrained language models can transfer to structured settings, but they leave open which components of the model actually enable this transfer and whether the observed benefits arise from the pretrained weights themselves or from the transformer architecture's inherent capacity to adapt to such data. Our work addresses this question directly by analyzing the layer-wise contribution of pretrained weights in such domains.

4.2 Contributions of Individual Layers to Fine-Tuning

A complementary research thread investigates how different layers evolve during fine-tuning. Early probing studies by Merchant et al. (2020) and Zhou and Srikumar (2022) revealed that fine-tuning mainly reshapes higher layers, while lower layers remain largely stable, suggesting that base representations learned during pretraining are preserved and reused. More recent large-scale analyses, such as the alignment experiments of Harada et al. (2025), confirm that the strongest correlation with downstream gains often arises in mid-level layers rather than the deepest ones. The most influential methodological contribution in this area (Tamkin et al., 2020), introduced *partial re-initialization* to isolate the contribution of specific layers in a BERT (Devlin et al., 2019) model. They showed that when the fine-tuning training set is limited in size, only a few early layers carry meaningful transferable signal; as the data grows, more layers become useful. This idea, that transferability is localized yet dependent on training set size, serves as a central premise of our work, which reexamines it in the context of structured, non-linguistic data using decoder-style small language models.

4.3 Model Truncation and Efficient Sub-models

Efficiency-driven studies offer additional insight into which parts of a model matter most. Sajjad et al. (2023) demonstrate that removing upper layers of BERT or RoBERTa (Liu et al., 2019) barely affects GLUE accuracy, implying that the lower layers contain most transferable capacity. Kavehzadeh et al. (2024) extended this observation to decoder architectures, showing that sub-models with only thirty-six out of forty layers can perform on par with the full `Llama-2-13B`. Similarly, works exploring top-layer re-initialization, such as (Zhang et al., 2020), report improved stability once task-specific patterns replace pretraining artifacts. These findings indicate that substantial pretrained capacity can be preserved even after removing or re-initializing higher layers, suggesting a non-uniform distribution of pretrained knowledge across depth.

4.4 Fine-Tuning Behavior and Transfer Learning

A broader body of work examines how pretrained language models behave under limited supervision and domain mismatch. A first line of research shows that fine-tuning pretrained language models in low-data regimes is often unstable and prone to overfitting (Dodge et al., 2020; Mosbach et al., 2020; Perez et al., 2021).

In contrast, (Hegselmann et al., 2023; Garg et al., 2025b) show that even a small number of highly out-of-distribution examples can enable effective transfer under minimal supervision.

While studies on truncation focus on identifying which layers preserve pretrained competence, complementary work examines how layers adapt during fine-tuning. Layer-wise analysis further indicated that transferable information in pretrained models is concentrated in early components: lower layers encode surface-level features and remain relatively stable across tasks, while higher layers capture semantic abstractions and adapt strongly during fine-tuning (Tenney et al., 2019a; Merchant et al., 2020). Consistent with this observation, studies on cross-domain transfer show that adapting only the early representations—such as token embeddings—can be sufficient for effective transfer, with deeper layers primarily supporting task-specific refinement (Artetxe et al., 2020).

Taken together, these studies support a consistent conclusion across tasks and domains: the benefit of pretraining is localized, data-dependent, and often tied to learning input structure rather than high-level task semantics. Our findings on structured data align closely with this pattern.

5 Discussion

Architecture versus pretraining: what truly matters for structured data. Our results suggest that, once sufficient labeled data is available, most of the performance on structured classification tasks comes from the model's architecture and the supervised fine-tuning process rather than from deep pretrained representations. In practice, the transformer's structural design—its attention

mechanisms and capacity to model feature interactions—appears sufficient to learn the relationships present in tabular data. Pretraining contributes only marginally beyond the earliest layers. These findings extend earlier observations on partial re-initialization and layer truncation, showing that transferable knowledge is concentrated near the model's input interface and that this behavior persists when language models are adapted to serialized structured data. From a practical standpoint, pretraining should be regarded as helpful mainly in low-data regimes rather than as a necessary condition for deep models to perform well on structured inputs.

Locating transferable knowledge. Both the layer-wise probing and progressive re-initialization analyses indicate that most reusable information resides in the model's input interface, particularly in the embedding layer and the first transformer block. This pattern aligns with the kinds of cues available in serialized tables. Feature names such as *heart rate* or *metformin* provide linguistic anchors that the model can interpret, while recurring numerical and formatting patterns, for example, blood pressure readings like "132/88", thousands separators, or units such as "mg/dL", form systematic input patterns that the model can exploit. This sensitivity to input format echoes findings from studies on tokenization and arithmetic robustness (e.g., Singh and Strouse, 2024), which show that even small variations in number formatting can degrade performance. Overall, transfer in structured data tasks appears to rely on shallow structural patterns rather than on deep semantic generalization.

6 Conclusion

This study addressed three research questions examining how pretrained language models transfer to structured data tasks. Through probing and layer re-initialization experiments, we found that, unlike in unstructured NLP tasks, most of the transferable signal for structured classification is concentrated in the first pretrained layer. Building on this insight, we showed that truncated models retaining only the earliest layer can match or even surpass the performance of fully pretrained counterparts. Overall, our results reveal the limited importance of model depth and large-scale pretraining in structured domains, suggesting that future work should focus on lightweight, domain-aligned architectures and pretraining strategies tailored to the statistical and relational properties of structured data.

Limitations

This study has several limitations that should be acknowledged. First, our experiments focus solely on classification tasks. This choice makes evaluation straightforward and enables precise measurement of performance differences, but it also limits the generalizability of our findings to other task types. Future research could explore whether similar patterns hold for regression, ranking, or generation tasks.

Second, we evaluate only two relatively small open-source models, which makes it possible to perform detailed and repeated analyses. However, this limited scope may reduce the breadth of our conclusions. Expanding the study to include additional architectures and larger-scale models could offer a more complete understanding of how pretraining effects vary across model families.

Finally, although we consider a reasonable range of structured and unstructured tasks, our dataset collection remains modest. Testing on a wider variety of datasets and domains would help confirm the stability of our observations and clarify the extent to which they generalize.

Our work is intended solely for research purposes. We believe that our work poses no potential risks.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Vadim Borisov, Thilo Leemann, Kevin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2023. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8334–8353.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642. Association for Computational Linguistics.

Ruirui Chen, Weifeng Jiang, Chengwei Qin, Ishaan Singh Rawal, Cheston Tan, Dongkyu Choi, Bo Xiong, and Bo Ai. 2024. LLM-based

multi-hop question answering with knowledge graph integration in evolving environments. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14438–14451, Miami, Florida, USA. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2126–2136. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. 2025a. Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data. *arXiv preprint arXiv:2507.03971*.

Anurag Garg, Noah Hollmann, Gregory Peters, Thomas Nannie, Ivan Rubachev, Simon Shaolei Li, and Damien Teney. 2025b. Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data. *arXiv preprint arXiv:2507.03971*.

Léon Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520.

Yuto Harada, Yusuke Yamauchi, Yusuke Oda, Yohei Oseki, Yusuke Miyao, and Yu Takagi. 2025. Massive supervised fine-tuning experiments reveal how data, layer, and training factors shape llm alignment quality. *arXiv preprint arXiv:2506.14681*.

Xinyi He, Yihao Liu, Mengyu Zhou, Yeye He, Haoyu Dong, Shi Han, Zejian Yuan, and Dongmei Zhang. 2025. Tablelora: Low-rank adaptation on table structure understanding for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206 of *Proceedings of Machine Learning Research*, pages 5549–5581. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 2021 Conference on Neural Information Processing Systems (NeurIPS) Workshops*. NeurIPS.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Ahmed Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. 2021. Regularization is all you need: Simple neural nets can excel on tabular data. *arXiv preprint arXiv:2106.11189*.

Parsa Kavehzadeh, Mojtaba Valipour, Fuxiao Liu, and Liang Ding. 2024. Sorted llama: Unlocking the potential of intermediate layers of large language models for dynamic inference. *arXiv preprint arXiv:2309.08968*.

Rumeng Li, Xun Wang, and Hong Yu. 2024. LlamaCare: An instruction fine-tuned large language model for clinical NLP. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10632–10641, Torino, Italia. ELRA and ICCL.

Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina Semnani, Chen Yu, and Monica Lam. 2024. SUQL: Conversational search over structured and unstructured data with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4535–4555, Mexico City, Mexico. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

Thomas Mesnard and the Gemma Team. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *Advances in Neural Information Processing Systems*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Shourav B. Rabbani, Ibna Kowsar, and Manar D. Samad. 2025. Transfer learning of tabular data by finetuning large language models. *arXiv preprint arXiv:2501.06863*.

Ivan Rubachev, Akim Kotelnikov, Nikolay Kartashev, and Artem Babenko. 2025. On finetuning tabular foundation models. *arXiv preprint arXiv:2506.08982*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. Dropping layers in pretrained transformers: How much do you need? *arXiv preprint arXiv:2305.00017*.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *arXiv preprint arXiv:2402.14903*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642. Association for Computational Linguistics.

Alex Tamkin, Trisha Singh, Davide Giovanardi, and Noah Goodman. 2020. Investigating transferability in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1393–1401, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Hugo Touvron, Colin Raffel, Andrea Glaese, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 200–207. Association for Computing Machinery.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning. *arXiv preprint arXiv:2006.05987*.

Yichu Zhou and Vivek Srikumar. 2022. A closer look at how fine-tuning changes bert. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1046–1061, Dublin, Ireland. Association for Computational Linguistics.

A Appendix: Training Details

All experiments were conducted on a multi-GPU workstation equipped with 4× NVIDIA GeForce RTX 3090 GPUs, providing a total of 96GB distributed GPU memory. We estimate that all experiments combined required approximately 500 hours of computation time.

Our training approach combines both parameter-efficient fine-tuning with LoRA and regular full fine-tuning with float16. For each task, we adopt a *dual-focus training* setup, in which the model is fine-tuned on the complete prompt together with the corresponding target response.

LoRA Configuration . We employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) for parameter-efficient fine-tuning only on fully pretrained models alongside standard full fine-tuning. We used the following configuration: *Rank* (r): 64; *Alpha* (α): 32; *Scaling factor*: $\alpha/r = 0.5$; *Trainable parameters*: $\sim$0.5% of total parameters.

For each task, the best performance obtained across these settings is reported as the pretrained fine-tuned result.

Weight Injection Strategies We used two strategies: (1) FULLY-RANDOM: re-initialize a percentage of randomly selected parameters; (2) SELECTED-LAYERS: target specific layers for re-initialization; For both, we re-initialize the transformers layer by layer with random wights yielded from a normal distribution with mean 0 and same std as the source transformer.

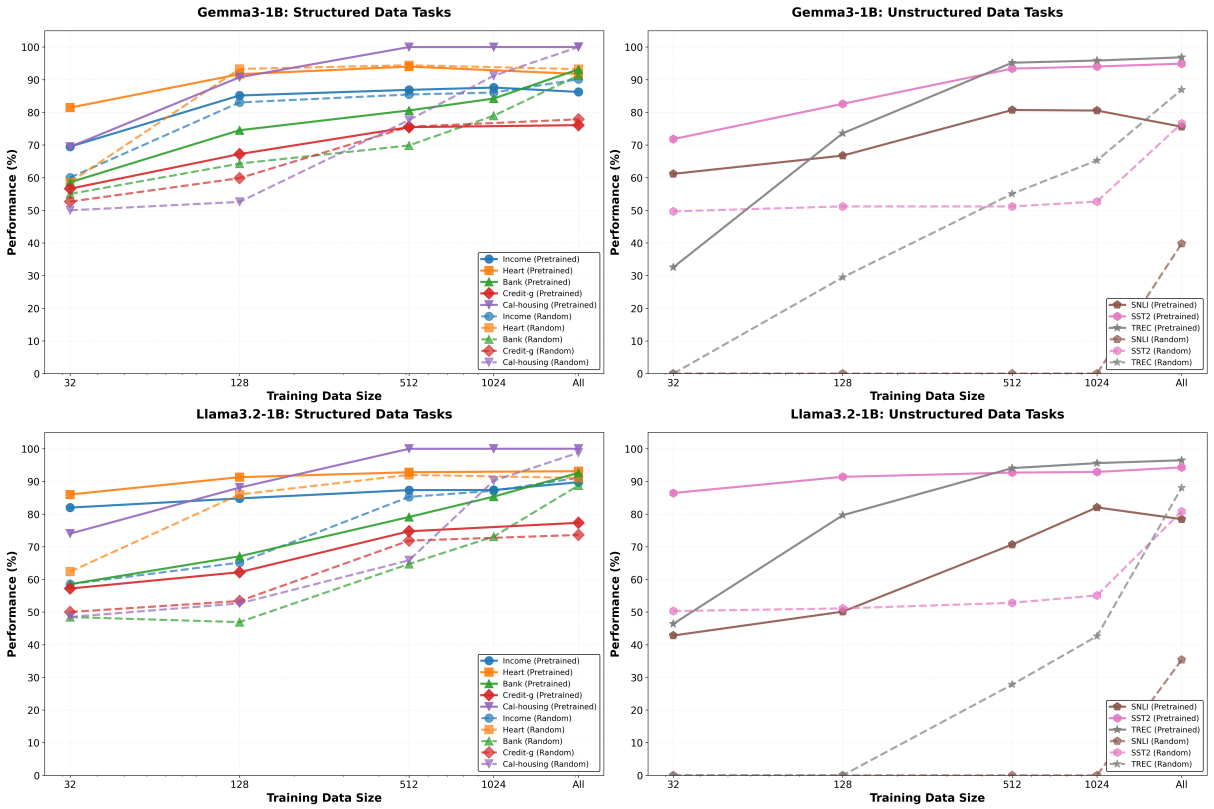

Figure 6: The performance of each task as a function the training size. the left figure present the structured tasks and the right side present the unstructured tasks. The shape present the task. The blue solid lines are the pretrained model and the dashed lines are the random weights models.

Training Hyperparameters Table 2 presents our training configuration. We train for 10 epochs with a learning rate of 2×10^{-4} and weight decay of 0.01. We use a warmup ratio of 0.06, applying warmup for 6% of total training steps.

Logistic Regression Setup. We train a regularized Logistic Regression (LoR) classifier implemented as a pipeline consisting of a `StandardScaler` for feature normalization followed by a `LogisticRegression` model. Model hyperparameters are selected using 5-fold cross-validation with `GridSearchCV`, optimizing for ROC-AUC (binary) or weighted one-vs-rest ROC-AUC (multiclass).

For binary classification, we evaluate both L1 and L2 regularization using the `liblinear` (L1) and `lbfgs` (L2) solvers, with the regularization strength $C \in \{0.1, 1.0, 10.0\}$. For multiclass classification, we use the `lbfgs` (L2), `saga` (L1), and `liblinear` (L2 fallback) solvers, employing a multinomial loss for `lbfgs`/`saga` and a one-vs-rest scheme for `liblinear`.

All models are trained with balanced class weights, a maximum of 2,000 iterations, and full parallelization across available CPU cores.

B Appendix: Model Selection

We compared `Llama-3.2-1B-Instruct` with larger variants in the same model family, and concluded that it is competitive in performance on the tasks that we care about in this study. Table 3 summarizes the performance of several Llama model variants under two regimes: zero-shot learning and supervised fine-tuning. The results illustrate that large-scale pretraining provides some initial capability even without task-specific adaptation (e.g., `LLaMA-3.3-70B-Instruct` reaches 81% on structured datasets), yet substantial gains appear only after fine-tuning (up to 94% with `LLaMA-3.1-8B-Instruct`). The consistent improvement across Adult Income, Heart Disease, and Bank benchmarks highlights that while pretrained representations offer a reasonable initialization, effective adaptation to structured inputs still requires task-level optimization. Notably, in the zero-shot setting, Llama-3.1-8B-Instruct correctly identifies the California Housing task as out-of-distribution and responds by indicating that fine-

Hyperparameter	Value
Learning rate	2×10^{-4}
Epochs	10
Weight decay	0.01
Warmup ratio	0.06
Gradient accumulation	16 steps
Per-device batch size	2
Optimizer	AdamW

Table 2: Training hyperparameters for all experiments.

tuning is required, a behavior not observed for the other datasets.

For the core analyses, we focus on small-scale open-source models—`Llama-3.2-1B-Instruct` and `Gemma-3-1B`—which balance between architectural expressiveness and interpretability. Their modest size enables systematic layer-wise reinitialization, probing, and resource-efficient replication. Both families are instruction-tuned and publicly available, facilitating reproducibility and alignment with the broader small-language-model research trend. Moreover, Llama and Gemma differ slightly in tokenizer design and pretraining corpus, providing a useful contrast between decoder-only architecture and compact transformer, allowing us to test whether observed effects generalize across pretraining pipelines rather than being model-specific artifacts.

C Appendix: The Impact of Data Size

Figure 6 complements the gap analysis presented in Figure 2 by showing absolute performance trends across data sizes. For structured tasks (left panels), both pretrained and randomly initialized models start from comparable accuracy when datasets are scarce. Pretrained variants converge faster, but the gap narrows as training data grow—by the full-data regime, both models reach nearly identical scores. In contrast, unstructured tasks (right panels) show a consistent and large advantage for pretrained weights: models trained from scratch remain substantially below their pretrained baselines even with full supervision.

D AI Assistants

We utilized AI assistants, such as ChatGPT, to assist with code formatting, phrasing suggestions, and LaTeX styling during writing. All outputs were reviewed and edited by the authors, ensuring no content was generated or used without human verification.

E License

We detail the models and datasets we use and their respective licenses in Table 4. We use all resources in accordance with their original intended purpose.

F Preserved Layers Experiments

In some experiments shown in Figure 4 in the text, mainly with `Gemma-3-1B`, we received responses that did not follow the textual format specified in the prompt. We treated such outputs as hallucinations and excluded experiments where the hallucination rate was high. As a result, for certain layer-level experiments, only a single seeded run was used instead of three, which is the standard for all other settings. Specifically, for `Gemma-3-1B` on the SNLI task, we used a single seed for layers 4, 5, and 9, while for layer 11, no valid runs were obtained for either `Gemma-3-1B` or `Llama-3.2-1B-Instruct`. For all other tasks, we report on three seeded runs using both models.

Setting	Model	Income	Heart	Bank	Credit-g	Cal-housing
Zero-shot	Llama-3.3-70B-Instruct	81.60	87.10	59.52	50.56	81.86
	Llama-3.1-8B-Instruct	82.08	75.60	60.37	48.55	-
Fine-tuned	Llama-3.1-8B-Instruct	91.80	94.05	93.34	74.25	100
	Llama-3.2-1B-Instruct	88.55	93.13	92.30	77.87	100

Table 3: Performance comparison (in ROC-AUC %) across models and settings on structured datasets.

Heart Prompt Example

Given the patient condition: Age: 58, Sex: M, Chest Pain Type: ATA, Resting blood pressure: 130, Serum cholesterol: 251, Fasting blood sugar > 120 mg/dl: 0, Resting electrocardiogram results: Normal, Maximum heart rate achieved: 110, Exercise-induced angina: N, ST depression induced by exercise relative to rest: 0.0, Slope of the peak exercise ST segment: Up.
Does the coronary angiography of this patient show a heart disease?

Bank Prompt Example

Based on the following bank information about the client: age: 31, type of job: technician, marital status: married, education: university.degree, has credit in default?: no, has housing loan?: yes, has personal loan?: no, contact communication type: cellular, last contact month of year: aug, last contact day of the week: Thursday, last contact duration (in seconds): 95, number of contacts performed during this campaign and for this client: 1, number of days that passed by after the client was last contacted from a previous campaign: 999, number of contacts performed before this campaign and for this client: 0, outcome of the previous marketing campaign: nonexistent.
Does this client subscribe to a term deposit?

SST2 Prompt Example

Consider the following sentence: "It 's a charming and often affecting journey."
Based on its content, determine whether the sentiment expressed is positive or negative?

Figure 7: Examples of hydrated prompts for structured and unstructured tasks. Each task is phrased as a natural-language query—structured inputs (e.g., Bank Marketing) ask a factual question, while unstructured inputs (e.g., SST-2) require subjective interpretation.

Artifact	Type	License
Bank Marketing	Dataset	CC BY 4.0
California Housing	Dataset	CC BY 4.0
German Credit	Dataset	CC BY 4.0
Adult Income	Dataset	CC BY 4.0
Heart Disease	Dataset	CC BY 4.0
SNLI	Dataset	CC BY-SA 4.0
SST-2	Dataset	CC0 1.0
TREC	Dataset	CC BY 4.0
Llama-3.2-1B-Instruct	Model	Llama 3.2 Community License
Gemma-3-1B	Model	Gemma Terms of Use

Table 4: License and usage summary of all datasets and models used in this study.