Scaling Experiments in Self-Supervised Cross-Table Representation Learning

Maximilian Schambach[1][*]   **Dominique Paul**[1][*]   **Johannes S. Otterbach**[2][†]

[1]Merantix Momentum, [2] nyonic
Berlin, Germany

Abstract

To analyze the scaling potential of deep tabular representation learning models, we introduce a novel Transformer-based architecture specifically tailored to tabular data and cross-table representation learning by utilizing table-specific tokenizers and a shared Transformer backbone. Our training approach encompasses both single-table and cross-table models, trained via missing value imputation through a self-supervised masked cell recovery objective. To understand the scaling behavior of our method, we train models of varying sizes, ranging from approximately 10^4 to 10^7 parameters. These models are trained on a carefully curated pretraining dataset, consisting of 135 M training tokens sourced from 76 diverse datasets. We assess the scaling of our architecture in both single-table and cross-table pretraining setups by evaluating the pretrained models using linear probing on a curated set of benchmark datasets and comparing the results with conventional baselines.

1   Introduction

Tabular data is abundant in many real-world applications across industries as well as research domains and has been argued to be the data type with the highest potential for AI impact [8]. Nevertheless, on tabular, deep learning approaches fail to consistently outperform established boosting implementations such as XGBoost, LightGBM, and CatBoost [7, 23, 11, 15]. Nevertheless, the success of the Transformer architecture [40] and self-supervised learning applied to large datasets in natural language and computer vision has motivated similar methods in the tabular domain. However, the scaling behavior of these approaches has not been investigated. This is mostly due to the fact that tabular benchmark data is often small and separate models are trained for each table, requiring that the models remain small for fast training and to avoid over-parametrization. This limits the scaling potential of the underlying architecture as both the model size as well as the training data would need to be scaled for a consistent increase in performance as shown in the language and vision domain [22, 16]. For most tables, however, accessing or creating more data is not possible.

In order to scale tabular deep learning approaches, the architecture needs to be able to generalize across multiple tables so that a large heterogeneous training corpus can be used. Furthermore, cross-table generalization amortizes the increased costs of training a large versatile model as opposed to training table-specific ones. Besides a potential performance gain from increased scale, a tabular general-purpose model that generalizes across multiple tables is of practical importance. For example, pretrained tabular backbones lend themselves as feature extractors and could be of interest in the zero- and few-show regime, with no or only limited training data, as well as in joint representation learning to be used with language models or incorporating inter-table dependencies in relational databases.

---

[*]Equal contribution. Corresponding author: `maximilian.schambach@merantix.com`.
[†]Work done while at Merantix Momentum.

While a variety of cross-table approaches based on Large Language Models (LLMs) have been proposed in the past [18, 47, 46], table-specific architectures are extremely scarce. Despite showing first promising results, we believe the potential of LLMs in the context of tabular data to be limited mainly due to the technical challenges around tokenization which we will discuss in detail. On the other hand, simple and straightforward Transformer-based architectures in the tabular domain are the exception [14, 51] while the field is scattered with, in our opinion, complex and sometimes convoluted architectures. We believe a solid understanding of a Transformer-based tabular architecture, and, in particular, the preceding table tokenization, to be the core of future developments and a successful scaling of architectures towards a new state of the art. In our opinion, the full potential of existing approaches, namely self-supervised Transformer-based architectures, has yet to be understood.

To address these limitations, we propose a clean and simple Transformer-based architecture, similar to the FT-Transformer [14], and generalize the architecture for cross-table self-supervised pretraining via masked cell recovery. Overall, our main contributions are as follows:

- We propose a novel architecture and training pipeline for cross-table pretraining based on self-supervised masked cell recovery. This loss can be naturally interpreted as multi-variate value imputation, a formidable problem in real-world applications.
- We investigate the scaling behavior of the proposed approach both in a single- as well as a cross-table pretraining setup. We do so by training four model configurations with backbone sizes ranging from roughly 10^4 to 10^7 parameters using a large curated heterogeneous pretraining corpus of 76 datasets and evaluating the pretrained models via linear probing using a small curated collection of benchmark datasets.

2 Cross-Table Representation Learning

While a wide range of approaches has been proposed in the context of learning representations for single tables, covering both supervised [20, 14] as well as self-supervised methods [38, 1, 35, 3, 34], how to best design architectures for learning representations across multiple tables is still an open question in the community. Following the tremendous success of deep learning in natural language and computer vision, Transformer-based architectures trained via self-supervision at scale are most promising to push the state of the art in tabular representation learning and perhaps finally surpass the strong conventional baselines. However, as opposed to natural language or computer vision, where tokenization and embedding methods naturally generalize across a wide range of datasets, the characteristics of tabular data are table-specific. Notably, different tables usually have different numbers of columns with numerical and categorical features, as well as column-specific statistics. That is, even if column names have a similar label indicating a shared semantic, the corresponding (joint and marginalized) statistics may be extremely diverse. Moreover, unlike language or images, tabular data does not possess a natural ordering and is invariant against column and row permutations.

Cross-table tokenization   Tokenization transforms tables (or individual rows) into a sequence of tokens, which are subsequently embedded in a shared embedding space and processed by the model backbone. In the single-table case, tokenization can be achieved by a combination of conventional tabular encoding and subsequent embedding [14, 13]. Numerical features can be tokenized via standardization or quantile transformation while categoricals can be tokenized via integer or one-hot encoding [21]. Linear projections or lookup embeddings map the tokens into the embedding space. However, a cross-table generalization of these approaches is not straightforward and has only recently been proposed within the XTab framework [51]. Here, table-specific tokenizers are used to extend the FT-Transformer approach [14], whereas the shared backbone contextualizes the embeddings.

A currently popular approach to cross-table tokenization and representation learning is to serialize a table's row into a string, e.g. "[Column A] is [Value 1], [Column B] is ..."), and then use a pretrained LLM to generate the row's embeddings. Many works exist in this area, notably utilizing pretrained BERT models [18, 47, 37, 43], as well as GPT-style generative architectures [6, 50]. Badaro et al. [2] and Hegselmann et al. [17] discuss and compare multiple forms of table serialization. This seemingly straightforward concept of table serialization and text-based tokenization comes with a few challenges and pitfalls. (i) Text tokenizers struggle with numerical features, which are typically broken down into multiple tokens by splitting at the decimal point and other subwords in the vocabulary. Recent research has shown that this likely leads to subpar performance on numerical tasks such as arithmetic and financial reasoning [31, 49, 45, 25]. While some workarounds, like

character-level tokenization for numeric features, have been used [50], they don't fully address the core issue and introduce additional complexity by requiring a separate decoder architecture. (ii) The coding scheme is not token-efficient, resulting in an excessive amount of tokens per cell. As Transformers scale quadratically with the input's length, the excessive representation length of a row requires more computational power than we believe is necessary. Hence, the number of columns that can be encoded is limited by the context length of the backbone model. (iii) When using causal language modeling, we need to artificially introduce a column order, despite the table's natural column permutation invariance. To break this artificial order, any-order learning needs to be enforced, leading to an exponential overhead in column orders that need to be trained, e.g. via permutation augmentation. On the other hand, in masked language modeling, the masking of individual tokens is not the same as blanking an entire table cell, requiring special treatment of the masking function.

Drawing parallels with text tokenization in natural language processing, we recognize that tokenization is a nuanced, domain-specific problem. Tokenizer developments in natural language have significantly enhanced Transformer-based language models by addressing linguistic and engineering challenges [28, 52]. In the same way, tokenization for tabular data demands specialized efforts and meticulous experimentation to optimize its utility and compatibility with Transformer architectures.

Permutation invariance and imputation loss   While the embeddings contextualized by a Transformer are inherently permutation invariant, this invariance is typically explicitly broken by introducing positional encodings [40, 12]. Nevertheless, in particular, LLM-based tabular learning architectures use positional encoding and address the problem, if at all, via permutation augmentation [32, 5, 50]. Positional encodings are not helpful for tabular data due to their invariance against column permutations. Instead, semantic column encodings, e.g. via additive column-specific bias embeddings, can be a useful inductive bias to distinguish between different columns [14, 51].

A possible solution are bidirectional models, such as BERT [10], based on masked token recovery losses, akin to a denoising objective. Note that this is not a natural loss for a language, which is typically constructed in a sequential manner. However, this objective is most natural for table representation learning. As table columns have no natural order, and often suffer from missing values, one can interpret masked cell recovery as an imputation of missing values. In fact, this allows for a natural generalization to a table-generative model using Markov Random Field sampling [41].

Cross-table pretraining   In the supervised case, early works treat tables as images and utilize general-purpose vision backbones [36], whereas recent approaches such as TransTab [44] are limited to tables from similar domains. In a different line of research, prior-fitted networks were introduced, recasting the problem to approximate Bayesian inference learned over a large synthetic prior, dubbed TabPFN [19]. While useful for practitioners and conceptually interesting, TabPFN is limited to small datasets and classification tasks based on purely numeric features and cannot be scaled naively.

Most self-supervised tabular learning approaches are explored in the single-table domain, ranging from autoencoders [48], contrastive approaches [38, 9, 35, 3], to more recent masked autoencoding objectives [1, 27]. In the cross-table setup, some works deal with self-supervised representation learning for tables with partially (or largely) overlapping columns [24, 30]. We are aware of only one non-LLM-based architecture for unconstrained tabular representation learning, namely the recently proposed XTab framework [51]. XTab generalizes the FT-Transformer to multiple tables via table-specific tokenizers and otherwise uses its exact hyperparameter configuration. Notably, XTab's Transformer backbone has less than 1 M trainable parameters.

3   Proposed Approach

We propose a simple Transformer-based architecture and training pipeline for cross-table pretraining that minimizes inductive biases as shown in Figure 1. This way, the proposed approach can be used as a baseline for further experimentation, for example around cross-table tokenization techniques. Our approach builds on the FT-Transformer [14] and is similar to the recently proposed XTab framework [51], with a few important distinctions which we outline in the following.

Tokenization   We employ table-specific tokenizers and use quantile encoding of numerical features combined with look-up embeddings as opposed to quantile transformation with subsequent linear projection embeddings used in FT-Transformer, XTab, and other approaches [13]. That is, instead of transforming the features in order to normalize the column distributions, we encode each value using
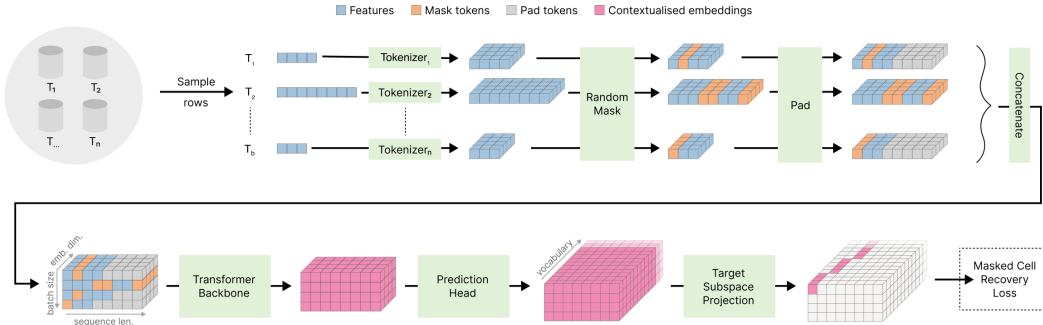
Figure 1: Overview of the proposed cross-table pertaining architecture. Individual tables are tokenized using table-specific tokenizers, including numerical as well as categorical features, and processed by a shared Transformer backbone.

its quantile index. Encoding numericals as quasi-categorical values makes the further treatment of all columns uniform. It simplifies the overall setup and makes the implementation easier to optimize, e.g. via vectorization. As all values are treated equally, there is no need to distinguish between numericals and categoricals at inference. Hence, balancing classification and regression losses is not necessary. However, the gained flexibility and robustness come at the cost of a quantization error and increased numbers of learnable embedding parameters, depending on the number of quantiles chosen. However, combining a low embedding space and linear up-projection can counter this problem, which we plan to address in future work. Furthermore, the ordinal character of the encodings is lost without explicit additional treatment. For categoricals, we use standard integer encoding and embedding via learnable look-up embeddings. Numerical features with less than 20 unique values are treated directly as categoricals. Finally, missing values are encoded as an additional NAN category for both numerical as well as categorical features. Sample statistics needed for the encoding , such as the quantiles, are estimated separately for each dataset using a fixed amount of 10,000 samples each before training.

Finally, we did not add any further additive encodings such as positional encodings and table- or column-specific bias terms, in order to minimize inductive biases and to retain the permutation invariance of the architecture. Hence, the column- and table-specific characteristics have to be learned by each embedding individually. While our current work only uses minimal architectural requirements, we see different types of additive encodings as an interesting prospective research question likely to increase the efficiency of the model training.

Data interleaving To obtain rows from multiple tables, we sample from a large heterogeneous corpus of tables, which we describe in detail in Section 4. We choose to perform stratified sampling, that is, in every batch the occurrence of each dataset is equally likely, regardless of the dataset size. This way, we sample uniformly from tasks and domains instead of sampling uniformly from the union of datasets. As a consequence, smaller datasets are iterated over more often than large ones. To process these samples in a single batch, we add a learnable padding token to each sequence up to the maximum number of tokens per batch. This is vastly different from XTab, which utilizes a federated learning approach, deploying the table-specific tokenizers on individual GPUs. By processing inter-table samples natively, we are able to scale the required hardware independently of the number of tables contained in the pretraining dataset. In fact, we perform all experiments on a single GPU. Our approach can easily be further parallelized using standard techniques from distributed training.

Contextualization and learning objective The interleaved batch of tokens are contextualized by a single Transformer backbone. In line with FT-Transformer and XTab, we use the pre-norm variant due to better performance and stability in the natural language context [42]. For self-supervised pretraining, we use the masked cell recovery objective – the tabular analog of masked language modeling (MLM). A random subset of tokens per cell is masked with a learnable Mask token and the training objective is to reconstruct the masked values from the contextualized embedding of the corresponding masked token. We note that this is a natural loss for tabular data, as opposed to MLM in the case of natural language. Masked tokens can be interpreted as missing values, a common occurrence in practical table modeling problems, and the recovery objective is simply the imputation of its value, whereas masking and recovering words in NLP is a less intrinsically meaningful objective. Compared to traditional imputation methods such as univariate mean, median, or mode estimation,

the imputation loss is multi-variate in nature. Hence it can capture richer dependencies between columns and other missing values, that are not able to be captured with standard methods. In the cross-table regime, this loss has been shown to perform better than contrastive pretraining while being more lightweight [51]. As opposed to XTab, we fully replace masked tokens with a single learnable mask embedding instead of random values drawn from the marginalized distribution. We believe this to yield a stronger training signal, but a comparison is left for future works. Note that, in order to obtain a uniform masking rate for all tasks, masking is performed before padding of the tokens.

For the cell recovery, the contextualized masked tokens are projected by a linear layer into the corresponding target probability space. As all values have been effectively encoded into categoricals, we optimize for classification via minimization of the cross-entropy loss. Unlike XTab, we do not use table-specific target heads but perform the target projection into the union of the individual column's target probability spaces. More precisely, given the individual column-specific target probability spaces $\mathcal{C}_{ij}$ for column j of dataset i, the full target probability space is modeled as their direct product, $\mathcal{C} = \prod_{i=1}^{M} \prod_{j=1}^{N_i} \mathcal{C}_{ij}$. However, the calculation of the cross-entropy for each token is restricted to its individual subspace via binary masking corresponding to an orthogonal projection onto $\mathcal{C}_{ij}$.

4 Datasets

Pretraining corpus   In order to perform meaningful scaling experiments, sufficient training data is required. As of now, heterogeneous high-quality tabular training data is not widely available. Instead, we chose to create a large heterogeneous training corpus by utilizing several tabular benchmark datasets of different tasks and sizes. For benchmarking, we then restrict ourselves to a small set of curated datasets as discussed next.

We gather datasets from multiple OpenML collections [39], and only kept datasets with more than 1000 rows and 10 to 50 columns. We discarded categorical columns that have more than 64 unique values. Finally, we manually deduplicated the remaining datasets.

Figure 2: Column and row statistics of the individual datasets contained in our curated pretraining corpus.

In total, we obtain a corpus containing 76 tables including 30 binary and 26 multiclass classification tasks as well as 20 regression datasets of different widths and sizes. Overall, the pretraining corpus contains ca. 135 M tokens in total. Using the previously discussed table-specific tokenization approach, we obtain a token vocabulary size, i.e. the number of unique numerical quantiles and categories to be embedded via look-up, of roughly 66 k. As a comparison, the BERT language model was trained using a vocabulary size of about 30 k, whereas GPT-2 used ca. 50 k. The feature and sample statistics of our pretraining corpus are shown in Figure 2. More detailed information on the datasets and statistics are presented in Appendix C.

Benchmark datasets   Instead of evaluating on a similarly large corpus of datasets or curating a larger set of datasets and splitting it into two folds similar to XTab, we believe a small curated set to be more suitable for investigating these early scaling experiments as opposed to average rank performance across a large benchmark suite. This way, we anticipate gaining more nuanced insights into the performance behavior. For these reasons, we followed the work by Borisov et al. [5] and use five tabular datasets for our evaluation, namely HELOC, California Housing, Adult Income, Cover Type, and Higgs, details of which are shown in Appendix C. These datasets cover a range of tasks (binary and multi-class classification, as well as regression), different numbers and types of columns (from 9 to 55 features), as well as sizes, ranging from roughly 10 k to 10 M samples per dataset. Even in the single-table case, we expect a Transformer-based model to perform severely differently across these five datasets. We split each dataset into 60 % used for pretraining and 40 % evaluated via a 5-fold cross-validation. We describe the pretraining and evaluation procedure in detail next.

5 Experiment Description

We perform scaling experiments for the proposed architecture using self-supervised pretraining in the single-table as well as the cross-table setup. In total, we investigate four different model
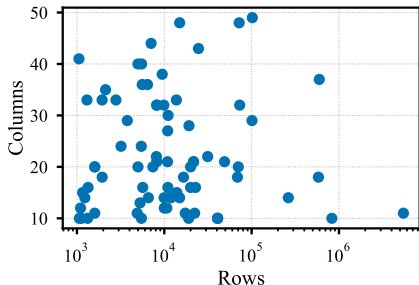
configurations, covering four orders of magnitudes in terms of the backbone model's parameter count, ranging from 13 k to 16 M. Due to limitations with respect to the dataset sizes, for the single-table case, we evaluate models S, M, and L, whereas M, L, and XL are considered in the cross-table case.

Single-table evaluation   Serving as a baseline, we investigate the scaling behavior of our approach in the single-table case. That is, for each table in our benchmark suite, we train a separate model via the imputation loss using the mentioned 60 % pretraining set. We then evaluate the task-specific performance of the pretrained model via linear probing using 5-fold validation on the remaining 40 % of each benchmark dataset. Linear probing is a well-established method to assess the quality of embeddings obtained via self-supervised pretraining and effectively corresponds to learning a linear projection layer supervised on the table-specific task. Hence, linear probing investigates the linear separability of the table representations with respect to a specific downstream task which the model was not explicitly trained on. Note that we evaluate the pretraining performance and do not perform any supervised fine-tuning of the tokenizers or backbone, which we leave for future investigations.

Cross-table evaluation   Secondly, we investigate the cross-table case. Here, each model configuration is pretrained using the imputation loss on the large pretraining corpus. As our architecture uses table-specific tokenizers, the cross-table pretrained models cannot directly be investigated on the benchmark datasets. To this end, we again use the table-specific pretraining portion and train the corresponding tokenizers for the pretrained model. To observe the transferability during training, we checkpoint the pretraining models every 250 M training tokens and evaluate all checkpointed models via linear probing on all benchmark datasets. Importantly, for a direct and fair comparison, we also use the same self-supervised learning objective here as in the single-table case to be able to assess the impact of cross-table pretraining. In this evaluation, we perform two variations: one where the pretrained backbone is frozen and only the tokenizer is trained, and one where the tokenizers and backbone weights are trained jointly. The obtained models are then evaluated via linear probing in full analogy to the single-table case using 5-fold cross-validation on the remaining portion of each dataset. Again, we do not perform any supervised fine-tuning.

We want to point out that in both cases a comparison to baselines is challenging, as existing methods, such as boosted trees, are trained in a supervised fashion on a single table. This is in stark contrast to this work, which uses self-supervised training without labeled targets and simply uses the representation features to train a linear model on top to predict the target. Furthermore, a comparison to other cross-table architectures is difficult, as the only existing approach, XTab, is trained in a federated setup requiring a training cluster of, in our case, 76 GPUs, which is outside our computational budget.

Hyperparameters and training   Trainings are performed via mini-batch stochastic gradient descent using the AdamW optimizer [26] with the default parameters. In the single-table experiments, we choose a batch size of 2048 which we reduce to 512 for the cross-table pretraining due to memory constraints. In total, we use 5 M, 10 M, and 25 M samples for pretraining the S, M, and L model in the single-table cases, respectively. In the cross-table case, we train all model configurations using 75 M samples, i.e. rows. The total number of training *tokens* is calculated by summing the number of cells for all samples excluding Padding tokens. For the learning rate, we choose a warmup phase for the first 10 % of training samples, linearly increasing the learning rate from 5×10^{-5} to 10^{-3}, and a cosine decay to 0 for the remaining 90 % of training samples. We employ a global weight decay, i.e. an L_2-norm regularization, of 10^{-2}. Throughout, we use a dropout rate of 10 % during training. For all experiments, we use a masking fraction of 25 %. More details on the used hyperparameters are given in Appendix A. All experiments are conducted using compute nodes with 8 CPU cores, 32 GB of RAM, and a single Nvidia L4 GPU.

Baseline methods   For comparison, we evaluate two baseline approaches. We investigate per-table performance using XGBoost [7], as well as a simple linear model using the raw features as predictors. Naturally, these methods are fitted on each benchmark dataset separately and do not allow for cross-table generalization. In all cases, we do not perform any hyperparameter optimization – including our proposed approach. As we use a different split of the benchmark data, due to the necessity of setting aside a portion for self-supervised pretraining, we cannot directly compare with the many baselines presented in the paper by Borisov et al. [5]. However, we do not expect the results to be fundamentally different on the splits used here as we follow the identical evaluation protocol via five-fold cross-validation.
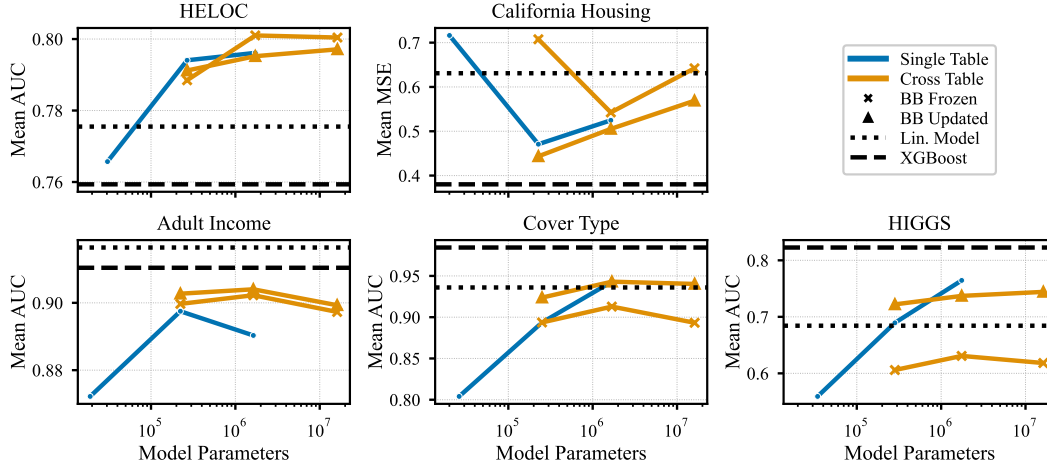
Figure 3: Mean 5-fold cross-validation linear probing results for the considered benchmark datasets in the case of single-table as well as cross-table pretraining with frozen and updated backbones (BB). The (supervised) performance of a linear model as well as XGBoost are shown for comparison.

6   Results

Our main results investigate the scaling behavior of the different models in terms of their linear probing performance on the benchmark datasets and are shown in Figure 3.

Single-table performance   Investigating the single-table case, we make the following observations: First, the imputation objective of recovering masked cell values is indeed informative on the dataset-specific downstream task. Recall that we do not perform any supervised fine-tuning. It indicates that the models are indeed learning multi-variate dependencies to efficiently recover missing values. That is, despite the model not being trained on the task specifics, the obtained contextualized features show good linear separability with respect to the downstream tasks. In most cases, in particular for HELOC and HIGGS, the contextualized features have more predictive power than the unprocessed ones as shown by the comparison with the linear model. Generally, the results are sub-par compared to a non-optimized XGBoost, which, however, is trained in a supervised fashion. With respect to the backbone size, we see slight improvements with scale: The linear probing performance increases with the amount of backbone parameters, as expected. We do, however, observe that in most cases, in particular with smaller datasets, this increase is saturated already with the Medium model configuration while larger datasets, such as Cover Type and HIGGS, do not show this saturation. This is, to some extent, expected as the amount of training data has to be scaled with increasing backbone parameter counts. This supports our claimed need for cross-table approaches in order to be able to scale tabular models towards a much larger scale. The training loss and imputation accuracies for all trained models are provided in Appendix B.

Cross-table performance   Generally, we observe a slight increase in performance when using cross-table pretraining, in particular notable in the HELOC and Adult Income datasets. Typically, the updating of the backbone parameters jointly with the training of the tokenizer, again in a self-supervised fashion, tends to perform better than the frozen weights obtained during pretraining, with the exception of the HELOC dataset. Overall, we do not see a strong increase in performance with scale, which indicates that we might be far from optimal dataset sizes to saturate the models and learn meaningful cross-table contextualization patterns within the backbone. On the other hand, we also observe that scaling does not hurt performance, which could indicate that increasing the dataset sizes can lead to improvements. Slight increases can be observed in the HELOC dataset, whereas increased scale actually leads to worse performance in some instances such as the California Housing dataset. Moreover, we see an interestingly steep increase in imputation accuracy during transfer learning on the benchmark datasets, as shown in Figure 4 in the case of the Adult Income dataset and in Appendix B for the remaining ones. This encourages the usage of the proposed cross-table pretrained model as a multivariate imputation system.
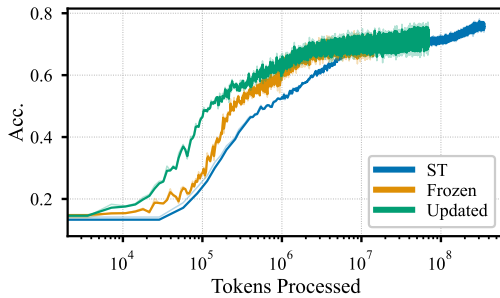
7

Figure 4: Imputation accuracy during transfer learning of the cross-table pretrained backbone onto the Adult Income dataset.
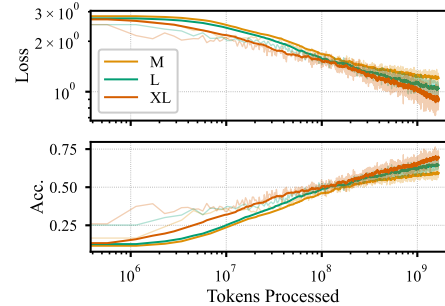
Figure 5: Pretraining loss curves and imputation accuracy of the cross-table models.

Further, looking more closely at the linear probing performance at several stages during pretraining, which are shown in Appendix B, we do not see systematic improvements with longer pretraining. This is surprising and suggests that the backbone feature processing does not increase in performance with increased pretraining performance. That is, while we see an increase in pretraining imputation accuracy, this does not directly transfer to improvements with respect to the linear separability of the benchmark tasks, unlike our observations in the single-table case. This is an interesting observation that could be caused by a number of reasons opening several future research directions. First, we note that the cross-table pretraining was limited by our compute budget and that all models, in particular the L and XL variants, show further potential in training as shown in Figure 5. Here, the training loss of the XL model is hardly saturated and we expect further gains with longer training. This is less the case in the single-table training, for which we present the loss curves in Appendix B, which are limited by the individual dataset sizes and saturate much earlier. Second, the approach to using table-specific tokenizers comes at the cost of a comparably large parameter overhead. As previously mentioned, our cross-table pretraining vocabulary contains 66 k tokens and look-up embeddings, resulting in a large number of additional training parameters as detailed in Appendix A. For comparison, GPT-2 uses a 60 k subword vocabulary at a size of 1.5 B parameters and 40 B training tokens, which is orders of magnitudes larger than the ones used here. This imbalance of tokenization and backbone parameters could be a reason for the observed behavior. Continued scaling experimentation is required, while keeping the vocabulary size constant, e.g. by using larger pretraining datasets or improving the tokenization efficiency by using a lower-dimensional embedding space combined with a shared upsampling layer. Finally, we do not investigate supervised fine-tuning here. For one, it would be interesting to observe whether pretraining boosts supervised fine-tuning, similar to the results obtained in the XTab framework [51]. Furthermore, using a supervised objective, either in addition to the self-supervised pretraining or for the benchmark dataset transfer, would allow for introducing a learnable CLS token to aggregate the contextualized embeddings in an adaptive way. Currently, our evaluation protocol uses mean pooling across the contextualized row tokens, excluding Pad tokens, for linear probing. This aggregation might smooth out representations with higher predictive performance and is not task-adaptive. However, in the fully self-supervised case, it is not directly possible to introduce global contextualized representations, e.g. via a learnable CLS token.

Limitations and future work Our current approach offers several limitations, the most technical of which we previously discussed. In addition, our current evaluation protocol is limited in scope. A comparison across more benchmark datasets as well as supervised and unsupervised baselines, such as boosting or LLM-based approaches, is of interest and we plan to address this in the future. Also, performing a hyperparameter optimization should yield better results for both the considered baselines and the proposed approach, e.g. investigating the dropout and masking ratios in detail. Furthermore, we plan to investigate the cross-table tokenization in detail in future works, for example, the impact of row and table encodings as well as the explicit use of the individual table schemas, for example by using a separate learnable schema embedder. Finally, we argue there is a great need for more elaborate tabular training data in order to scale tabular models towards model sizes comparable to, e.g., GPT-2 as a first step. Similarly, benchmarks tailored to the usage of deep learning models need to be further developed and refined.

8

7 Conclusion

We have presented a novel architecture and training pipeline for cross-table pretraining and conducted scaling experiments that showed first interesting results. Generally, we see an increase in the linear probing accuracy across several benchmark datasets with larger model scales in both the single- and the cross-table case. Whereas models trained in a single-table fashion saturated, we saw slight improvements using cross-table pretraining, which was however limited likely due to a lack of training data or compute resources. We have discussed multiple possible reasons for the observed behavior and interesting further research directions.

References

[1] Sercan Ö Arik and Tomas Pfister. "TabNet: Attentive Interpretable Tabular Learning". In: *AAAI Conference on Artificial Intelligence* 35.8 (2021), pp. 6679–6687.

[2] Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. "Transformers for Tabular Data Representation: A Survey of Models and Applications". In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 227–249.

[3] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. "SCARF: Self-Supervised Contrastive Learning using Random Feature Corruption". In: *International Conference on Learning Representations* (2022).

[4] Mikhail Belkin. *Fit Without Fear: Remarkable Mathematical Phenomena of Deep Learning Through the Prism of Interpolation*. May 2021.

[5] Vadim Borisov, Tobias Leemann, Kathrin Sessler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. "Deep Neural Networks and Tabular Data: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 99 (2022), pp. 1–21.

[6] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. "Language Models are Realistic Tabular Data Generators". In: *International Conference on Learning Representations*. 2023.

[7] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, Aug. 2016, pp. 785–794.

[8] Michael Chui, James Manyika, Mehdi Miremadi, Nicolaus Henke, Rita Chung, Pieter Nel, and Sankalp Malhotra. *Notes From the AI Frontier: Insights from Hundreds of Use Cases*. Tech. rep. McKinsey Global Institute, 2018.

[9] Sajad Darabi, Shayan Fazeli, Ali Pazoki, Sriram Sankararaman, and Majid Sarrafzadeh. "Contrastive Mixup: Self- and Semi-Supervised learning for Tabular Domain". In: *ArXiv* 2108.12296 (2021).

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019).

[11] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. "CatBoost: Gradient Boosting with Categorical Features Support". In: *NeurIPS ML Systems Workshop*. 2017.

[12] Alexey Dosovitskiy et al. "An Image is worth 16x16 words: Transformers for image recognition at scale". In: *International Conference on Learning Representations* (2021).

[13] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. "On Embeddings for Numerical Features in Tabular Deep Learning". In: *Advances in Neural Information Processing Systems* (2022).

[14] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. "Revisiting Deep Learning Models for Tabular Data". In: *Advances in Neural Information Processing Systems* (2021).

[15] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. "Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?" In: *Advances in Neural Information Processing Systems* (2022), pp. 507–520.

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked Autoencoders Are Scalable Vision Learners". In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[17] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. "TabLLM: Few-shot Classification of Tabular Data with Large Language Models". In: *International Conference on Artificial Intelligence and Statistics* (2023).

[18] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. "TAPAS: Weakly Supervised Table Parsing via Pre-training". In: *Annual Meeting of the Association for Computational Linguistics* (2020).

[19] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second". In: *International Conference on Learning Representations*. 2023.

[20] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. *TabTransformer: Tabular Data Modeling Using Contextual Embeddings*. arxiv:2012.06678. 2020.

[21] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and others. *An Introduction to Statistical Learning*. Vol. 112. Springer, 2013.

[22] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. arxiv:2001.08361. 2020.

[23] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. 2017.

[24] Roman Levin et al. "Transfer Learning with Deep Tabular Models". In: *International Conference on Learning Representations* (2023).

[25] Xianzhi Li, Xiaodan Zhu, Zhiqiang Ma, Xiaomo Liu, and Sameena Shah. *Are ChatGPT and GPT-4 General-Purpose Solvers for Financial Text Analytics? An Examination on Several Typical Tasks*. arXiv:2305.05862. May 2023.

[26] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations* (2017).

[27] Kushal Majmundar, Sachin Goyal, Praneeth Netrapalli, and Prateek Jain. "MET: Masked Encoding for Tabular Data". In: *NeurIPS Table Representation Learning Workshop* (2022).

[28] Sabrina J. Mielke et al. *Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP*. arXiv:2112.10508. Dec. 2021.

[29] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. "Deep Double Descent: Where Bigger Models and More Data Hurt". In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (2021). Publisher: IOP Publishing and SISSA, p. 124003.

[30] Soma Onishi, Kenta Oono, and Kohei Hayashi. "TabRet: Pre-training Transformer-based Tabular Models for Unseen Columns". In: *ICLR Workshop on Understanding Foundation Models*. arXiv:2303.15747 [cs]. arXiv, 2023.

[31] Xinyu Pi et al. "Reasoning Like Program Executors". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 761–779.

[32] Michał Pietruszka et al. "STable: Table Generation Framework for Encoder-Decoder Models". In: *NeurIPS Table Representation Learning Workshop* (2022).

[33] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. *Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets*. arXiv:2201.02177. 2022.

[34] Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. *Revisiting Pretraining Objectives for Tabular Deep Learning*. arxiv.2207.03208. 2022.

[35] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. "SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training". In: *NeurIPS Table Representation Learning Workshop* (2022).

[36] Baohua Sun, Lin Yang, Wenhan Zhang, Michael Lin, Patrick Dong, Charles Young, and Jason Dong. "SuperTML: Two-Dimensional Word Embedding for the Precognition on Structured Tabular Data". In: *Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[37] Mohamed Trabelsi, Zhiyu Chen, Shuo Zhang, Brian D Davison, and Jeff Heflin. "StruBERT: Structure-aware BERT for Table Search and Matching". In: *ACM Web Conference* (2022), pp. 442–451.

[38] Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. *SubTab: Subsetting Features of Tabular Data for Self-Supervised Representation Learning*. arxiv:2110.04361. 2021.

[39] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. "OpenML: Networked Science in Machine Learning". In: *SIGKDD Explorations* 15.2 (2013), pp. 49–60.

[40] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing System* (2017).

[41] Alex Wang and Kyunghyun Cho. "BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model". In: *NAACL Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. 2019, pp. 30–36.

[42] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. "Learning Deep Transformer Models for Machine Translation". In: *Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1810–1822.

[43] Zifeng Wang, Chufan Gao, Cao Xiao, and Jimeng Sun. *AnyPredict: Foundation Model for Tabular Prediction*. arXiv:2305.12081. 2023.

[44] Zifeng Wang and Jimeng Sun. "TransTab: Learning Transferable Tabular Transformers Across Tables". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 2902–2915.

[45] Shijie Wu et al. *BloombergGPT: A Large Language Model for Finance*. arXiv:2303.17564. 2023.

[46] Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. "TableFormer: Robust Transformer Modeling for Table-Text Encoding". In: *Annual Meeting of the Association for Computational Linguistics* (2022), pp. 528–537.

[47] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. "TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data". In: *Annual Meeting of the Association for Computational Linguistics* (2020), pp. 8413–8426.

[48] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. "VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 11033–11043.

[49] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. *How Well Do Large Language Models Perform in Arithmetic Tasks?* arXiv:2304.02015. 2023.

[50] Tianping Zhang, Shaowen Wang, Shuicheng Yan, Jian Li, and Qian Liu. *Generative Table Pre-training Empowers Models for Tabular Prediction*. arXiv:2305.09696. 2023.

[51] Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. "XTab: Cross-table Pretraining for Tabular Transformers". In: *International Conference On Machine Learning* (2023).

[52] Vilém Zouhar, Clara Meister, Juan Luis Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. *Tokenization and the Noiseless Channel*. arXiv:2306.16842. 2023.

A   Additional Technical Details

Model configuration details   As previously discussed, we investigate four different model configurations. The configurations are depicted in Table 1 with a detailed parameter count, including the tokenizer and projection layers in the case of single-table and cross-table pretraining, given in Table 2.

Hyperparameter details   The full hyperparameter configuration of our approach and pretraining setup is given in Table 3 with model-specific ones detailed in Table 4. For the masking procedure, each value is masked randomly by drawing from the Binomial distribution of the corresponding masking fraction.

Table 1: Investigated Transformer backbone model configurations. The XTab configuration is shown for comparison. The depicted parameters correspond solely to the backbone parameters and do not include the trainable parameters of the tokenizers or projection heads.

Model	Embedding dimension	Number of heads	Number of layers	Parameter count
XTab [51]	192	8	3	740 k
S	16	4	4	13 k
M	64	8	4	200 k
L	128	8	8	1.6 M
XL	192	16	36	16 M

Table 2: Number of trainable parameters of the different investigated model configurations, including backbone, tokenizer, and projection head parameters. Note that, in the single-table case, the number of encoder and projection parameters depend on the datasets specifics, in particular the number of rows and categorical as well as numerical features. Therefore, we state the minimum and maximum number of parameters across the five used benchmark datasets for reference.

Model	Parameter count				
	Backbone	Tokenizer		Projection Head	
		Single-table	Cross-table	Single-table	Cross-table
S	13 k	3 k – 10 k	-	3 k–11 k	-
M	200 k	11 k – 41 k	2 M	11 k–41 k	2 M
L	1.6 M	22 k – 82 k	4 M	23 k–83 k	4 M
XL	16 M	-	6 M	-	6 M

Table 3: Detailed hyperparameter configuration of the proposed approach and pretraining setup.

Hyperparameter	Value
Dropout rate	0.1
Masking rate	0.25
Optimizer	
Type	AdamW
β_0	0.9
β_1	0.999
Weight decay	0.01
Learning rate init	0.000 05
Learning rate peak	0.001
Learning rate final	0
Batch size	$256 - 2048$
Train samples	$5\,\mathrm{M} - 75\,\mathrm{M}$
Encoder	
Numerical encoder	Quantile enc.
Num. quantiles	25
Categorical encoder	Integer enc.
Categorical threshold	20
Embedder	
Numerical embedder	Look-up emb.
Categorical embedder	Look-up emb.

Table 4: Detailed model-specific batch size and training samples used with our approach. Note that the number of training samples corresponds to rows and not the total amount of tokens, which also on the numbers of columns per table.

Model	Single-table		Cross-table	
	Batch size	Samples	Batch size	Samples
S	2048	5 M	-	-
M	2048	10 M	512	75 M
L	2048	25 M	512	75 M
XL	-	-	256	75 M

B   Additional Results

Single-table pretraining   The training loss and training imputation accuracy, including top-3 accuracy, are given in Figure 6. In general, we observe a strong imputation performance (on the training data), in particular for larger model sizes. In fact, the results for the smaller datasets, namely HELOC and California Housing, show a potential of overfitting with the Large model configuration. That is, we observe a double-descent-like training loss. However, we do not observe an additional validation set as the remaining portions of the datasets are used for linear probing. Hence, whether overfitting actually occurs or we are in an interpolation, i.e. grokking, regime, is speculative [4, 29, 33].

Cross-table pretraining   The linear probe accuracy in the cross-table pretraining case for different amounts of processed training tokens is depicted in Figure 7. That is, the cross-table pretrained models were checkpointed every 250 M tokens. All checkpointed models were then transferred to the individual benchmark dataset via self-supervised learning using the imputation loss in order to train a new tokenizer for each set, in full analogy to the evaluation of the final checkpoints depicted in the main text. The obtained models are then evaluated via linear probing as previously discussed. Again, we generally see that updating the backbone models during transfer on the specific downstream tasks is beneficial. This is not to be confused with a supervised fine-tuning as we update the backbone model weights jointly with the encoder via the self-supervised imputation loss. However, we observe, that regardless of the model size, the final downstream task performance does not significantly increase with longer pretraining. This is somewhat in line with the observations made in the main paper, i.e. that the cross-table pretrained models seem to be bottlenecked in their performance, likely due to a lack of pretraining and transfer learning data. This needs to be further investigated in future works.

A similar observation can be made from the transfer learning loss and imputation accuracy curves as depicted in Figure 8. We see that the imputation accuracy increases steeply with only few transfer learning steps, underlining the generalization capability of the models as an imputation system. However, further (self-supervised) transfer learning does not further increase imputation accuracy. Assuming that the imputation accuracy is an informative proxy for the downstream task performance, this would point towards a similar problem as previously discussed.
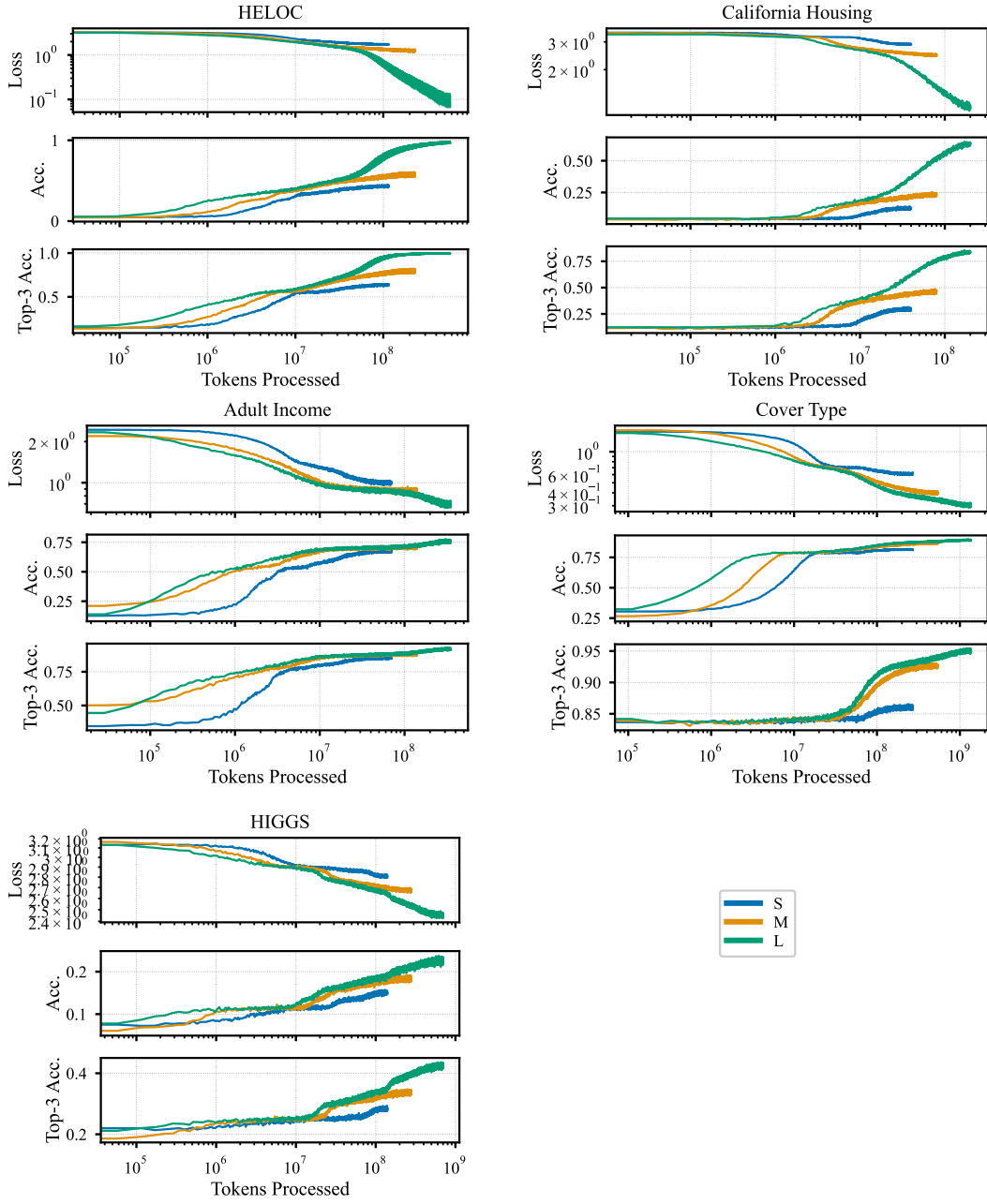
Figure 6: Training loss and imputation accuracy for the individual self-supervised single-table training using the five investigated benchmark datasets.
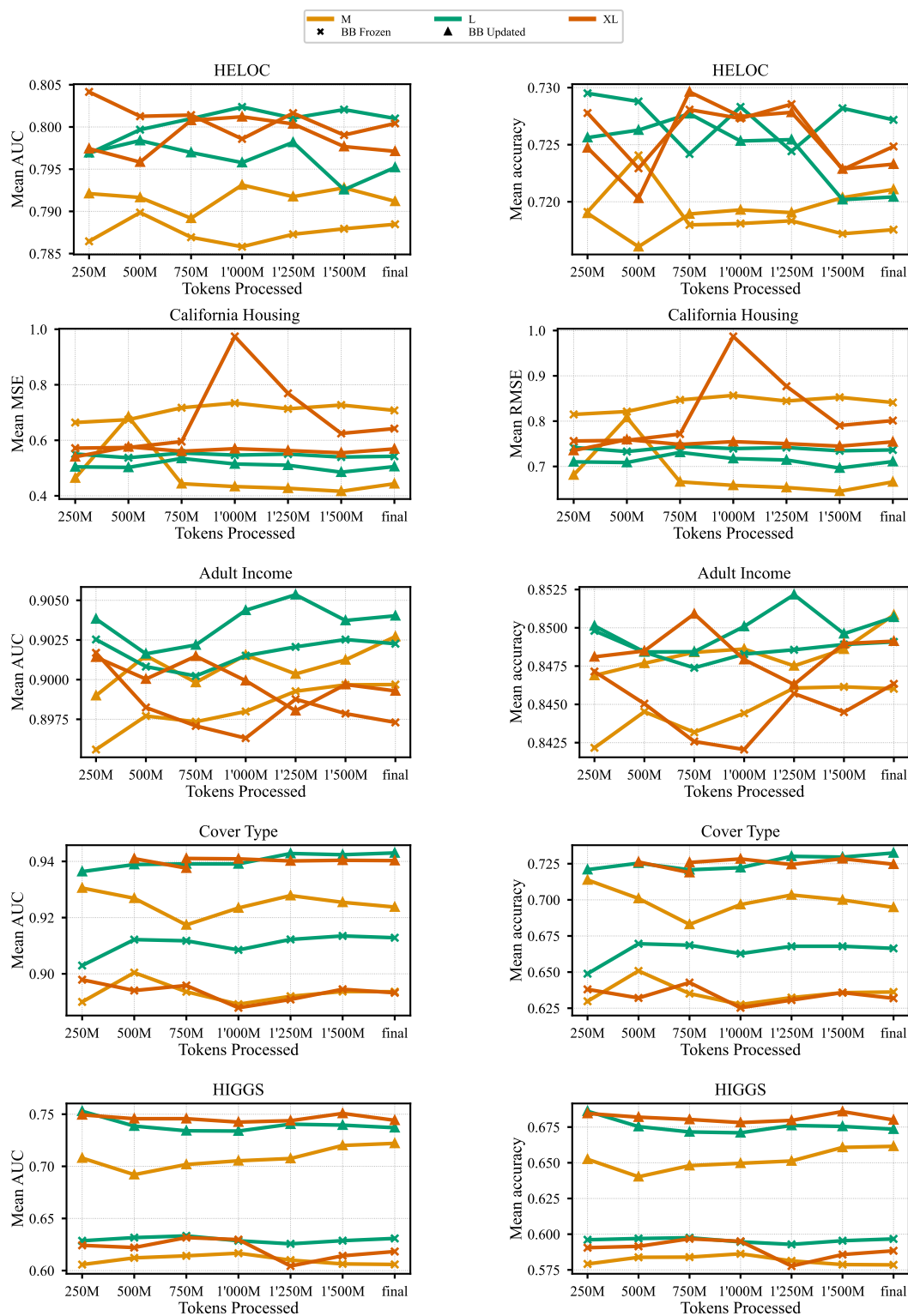
Figure 7: Linear probe evaluations using the pretrained model at different stages of pretraining for all considered benchmark datasets.
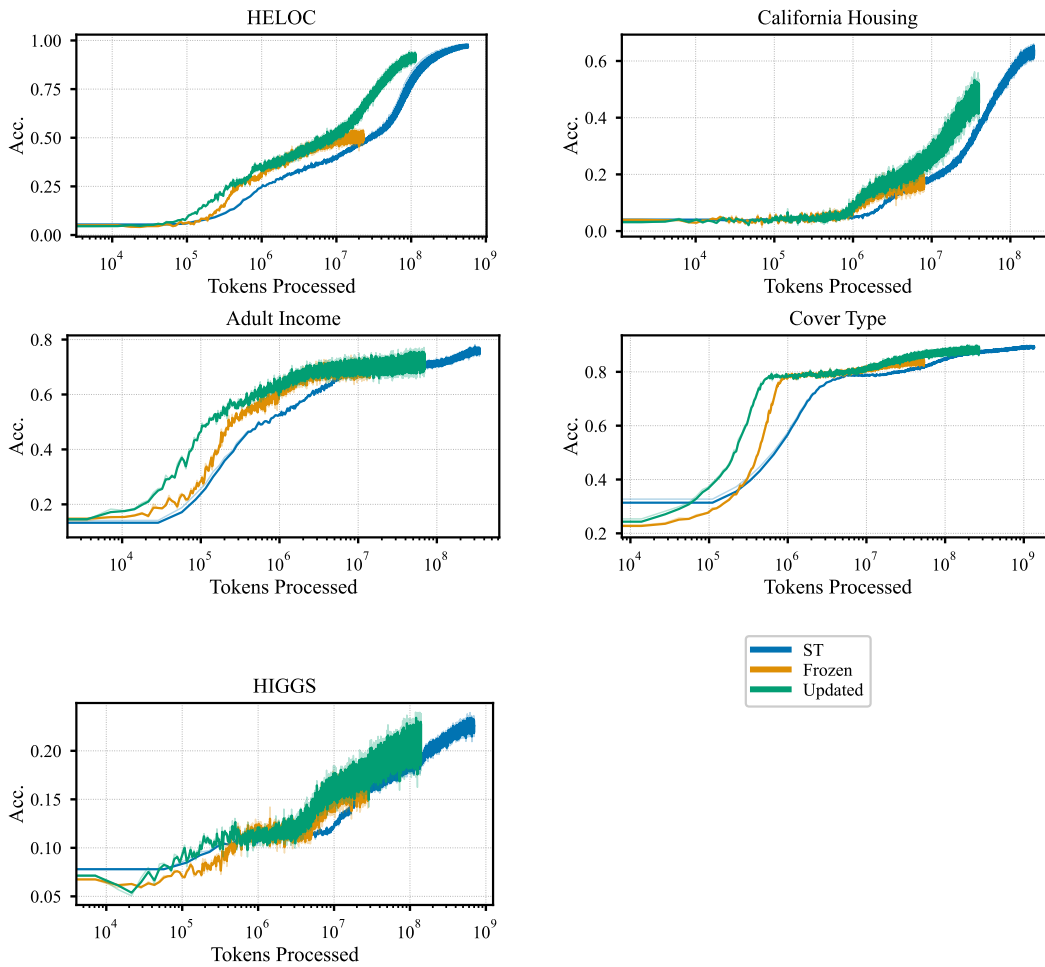
Figure 8: Imputation accuracy during the transfer learning of the cross-table pretrained models onto each benchmark dataset.

C Additional Dataset Details

Benchmark datasets   Detailed properties of the used benchmark datasets HELOC, California Housing, Adult Income, Cover Type, and HIGGS are stated in Table 5.

Pretraining corpus   As previously described, we collect a total of 76 datasets from various sources as our pretraining corpus. The column and row statistics of the used datasets are shown in Figure 9. Further details of all used datasets are given in Table 6.

Table 5: Properties of the used benchmark datasets. Throughout, evaluation is performed using a 5-fold cross-validation on the eval split.

Task	HELOC	California Housing	Adult Income	Cover Type	HIGGS
Task type	binary	regression	binary	multiclass	binary
Samples	7519	16 512	39 073	464 809	8 800 000
Features (all)	24	9	15	55	29
Features (numerical)	16	8	1	10	24
Missing values	0	0	5259	0	0
Pretrain split	60 %	60 %	60 %	60 %	60 %
Eval split	40 %	40 %	40 %	40 %	40 %

Figure 9: Row and column statistics of the datasets used in our curated pretraining corpus.

Table 6: Details of the datasets used in our curated pretraining corpus.

Name	Collection	Task	Classes	Rows	Columns	Numer. features	Cat. features	Unique cat.	NANs in %
wine-quality-white	automl_benchmark training	multiclass	7	4898	11	11	0	0	0.0
wine-quality-red	automl_benchmark training	multiclass	6	1599	11	11	0	0	0.0
wind	automl_benchmark training	binary	2	6574	14	14	0	0	0.0
waveform-5000	automl_benchmark training	multiclass	3	5000	40	40	0	0	0.0
wave_energy	openml_ctr23	regression	71 993	72 000	48	48	0	0	0.0
wall-robot-navigation	automl_benchmark training	multiclass	4	5456	24	24	0	0	0.0
video_transcoding	openml_ctr23	regression	10 960	68 784	18	16	2	8	0.0
texture	automl_benchmark training	multiclass	11	5500	40	40	0	0	0.0
steel-plates-fault	automl_benchmark training	binary	2	1941	33	33	0	0	0.0
spoken-arabic-digit	automl_benchmark training	multiclass	10	263 256	14	14	0	0	0.0
solar_flare	openml_ctr23	multiclass	8	1066	10	2	8	27	0.0
satimage	automl_benchmark training	multiclass	6	6430	36	36	0	0	0.0
sarcos	openml_ctr23	regression	11 414	48 933	21	21	0	0	0.0
rl	automl_benchmark training	binary	2	31 406	22	8	14	89	4.3
ringnorm	automl_benchmark training	binary	2	7400	20	20	0	0	0.0
qsar-biodeg	automl_benchmark training	binary	2	1055	41	41	0	0	0.0
pumadyn32nh	openml_ctr23	regression	8191	8192	32	32	0	0	0.0
porto-seguro	automl_benchmark training	binary	2	595 212	37	12	25	102	3.8
pol	automl_benchmark training	binary	2	15 000	48	48	0	0	0.0
pokerhand	automl_benchmark training	multiclass	10	829 201	10	5	5	20	0.0
pendigits	automl_benchmark training	multiclass	10	10 992	16	16	0	0	0.0
pc2	automl_benchmark training	binary	2	5589	36	36	0	0	0.0
pbcseq	automl_benchmark training	binary	2	1945	18	12	6	13	3.2
parity5_plus_5	automl_benchmark training	binary	2	1124	10	0	10	20	0.0
page-blocks	automl_benchmark training	multiclass	5	5473	10	10	0	0	0.0
nyc-taxi-green-dec-2016	automl_benchmark regression	regression	1811	581 835	18	9	9	22	0.0
naval_propulsion_plant	openml_ctr23	regression	51	11 934	14	14	0	0	0.0
mv	automl_benchmark training	binary	2	40 768	10	7	3	7	0.0
mushroom	automl_benchmark training	binary	2	8124	22	0	22	117	1.4
mofn-3-7-10	automl_benchmark training	binary	2	1324	10	0	10	20	0.0
microaggregation2	automl_benchmark training	multiclass	5	20 000	20	20	0	0	0.0
miami_housing	openml_ctr23	regression	2111	13 932	15	15	0	0	0.0
mc1	automl_benchmark training	binary	2	9466	38	38	0	0	0.0
letter	automl_benchmark training	multiclass	26	20 000	16	16	0	0	0.0
led24	automl_benchmark training	multiclass	10	3200	24	0	24	48	0.0
kings_county	openml_ctr23	regression	4028	21 613	21	17	4	45	0.0
kick	automl_benchmark training	binary	2	72 983	32	14	18	134	6.4
jm1	automl_benchmark training	binary	2	10 885	21	21	0	0	0.0

Name	Collection	Task	Classes	Rows	Columns	Numer. features	Cat. features	Unique cat.	NANs in %
hypothyroid	automl_benchmark_training	multiclass	4	3772	29	7	22	47	5.5
house_sales	automl_benchmark_regression	regression	4028	21 613	21	20	1	0	0.0
house_16H	automl_benchmark_training	binary	2	22 784	16	16	0	0	0.0
health_insurance	openml_ctr23	regression	75	22 272	11	4	7	21	0.0
grid_stability	openml_ctr23	regression	10 000	10 000	12	12	0	0	0.0
fried	automl_benchmark_training	binary	2	40 768	10	10	0	0	0.0
fps_benchmark_	openml_ctr23	regression	2675	24 624	43	30	13	96	6.6
fifa	openml_ctr23	regression	133	19 178	28	27	1	0	0.0
fars	automl_benchmark_training	multiclass	8	100 968	29	14	15	93	0.0
eye_movements	automl_benchmark_training	multiclass	3	10 936	27	24	3	6	0.0
elevators	automl_benchmark_training	binary	2	16 599	18	18	0	0	0.0
eeg-eye-state	automl_benchmark_training	binary	2	14 980	14	14	0	0	0.0
delays_zurich_transport	tabular_benchmark_ categorical_regression	regression	4082	5 465 575	11	8	3	12	0.0
cpu_activity	openml_ctr23	regression	56	8192	21	21	0	0	0.0
compas-two-years	automl_benchmark_training	binary	2	5278	13	7	6	12	0.0
colleges_usnews	automl_benchmark_training	binary	2	1302	33	32	1	51	18.2
colleges_aaup	automl_benchmark_training	binary	2	1161	15	13	2	56	1.5
colleges	automl_benchmark_regression	regression	4502	7063	44	32	12	206	33.5
cjs	automl_benchmark_training	multiclass	6	2796	33	31	2	68	73.8
churn	automl_benchmark_training	binary	2	5000	20	16	4	17	0.0
cardiotocography	automl_benchmark_training	multiclass	10	2126	35	35	0	0	0.0
baseball	automl_benchmark_training	multiclass	3	1340	16	15	1	7	0.1
bank32nh	automl_benchmark_training	binary	2	8192	32	32	0	0	0.0
autoUniv-au7-1100	automl_benchmark_training	multiclass	5	1100	12	8	4	10	0.0
Traffic_violations	automl_benchmark_training	multiclass	3	70 340	20	1	19	268	0.2
PhishingWebsites	automl_benchmark_training	binary	2	11 055	30	0	30	68	0.0
Moneyball	automl_benchmark_regression	regression	374	1232	14	8	6	66	20.9
MagicTelescope	automl_benchmark_training	binary	2	19 020	10	10	0	0	0.0
JapaneseVowels	automl_benchmark_training	multiclass	9	9961	14	14	0	0	0.0
Gesture Phase Segmentation Processed	automl_benchmark_training	multiclass	5	9873	32	32	0	0	0.0
GAMETES Heterogeneity 20atts 600_Het_0.4_0.2_7	automl_benchmark_training	binary	2	1600	20	0	20	59	0.0
GAMETES Epistasis 3-Way 20atts 0.2H EDM-1_1	automl_benchmark_training	binary	2	1600	20	0	20	60	0.0
Diabetes130US	automl_benchmark_training	multiclass	3	101 766	49	13	36	130	0.0
Brazilian_houses	automl_benchmark_regression	regression	5751	10 692	12	8	4	44	0.0
Bike_Sharing_Demand	tabular_benchmark_ categorical_regression	regression	869	17 379	11	6	5	14	0.0
BachChoralHarmony	automl_benchmark_training	regression	102	5665	16	2	14	102	0.0
Ailerons	tabular_benchmark_ numerical_regression	multiclass	35	13 750	33	33	0	0	0.0
2dplanes	automl_benchmark_training	binary	2	40 768	10	10	0	0	0.0