Fine-grained Attention in Hierarchical Transformers for Tabular Time-series

Raphael Azorin
EURECOM, Huawei Technologies
last@eurecom.fr

Zied Ben Houidi
Huawei Technologies
first.last@huawei.com

Massimo Gallo
Huawei Technologies
first.last@huawei.com

Alessandro Finamore
Huawei Technologies
first.last@huawei.com

Pietro Michiardi
EURECOM
last@eurecom.fr

ABSTRACT

Tabular data is ubiquitous in many real-life systems. In particular, time-dependent tabular data, where rows are chronologically related, is typically used for recording historical events, e.g., financial transactions, healthcare records, or stock history. Recently, hierarchical variants of the attention mechanism of transformer architectures have been used to model tabular time-series data. At first, rows (or columns) are encoded separately by computing attention between their fields. Subsequently, encoded rows (or columns) are attended to one another to model the entire tabular time-series. While efficient, this approach constrains the attention granularity and limits its ability to learn patterns at the field-level across separate rows, or columns. We take a first step to address this gap by proposing Fieldy, a fine-grained hierarchical model that contextualizes fields at both the row and column levels. We compare our proposal against state of the art models on regression and classification tasks using public tabular time-series datasets. Our results show that combining row-wise and column-wise attention improves performance without increasing model size. Code and data are available at https://github.com/raphaaal/fieldy.

ACM Reference Format:
Raphael Azorin, Zied Ben Houidi, Massimo Gallo, Alessandro Finamore, and Pietro Michiardi. 2024. Fine-grained Attention in Hierarchical Transformers for Tabular Time-series. In *ACM SIGKDD'24 – 10th Mining and Learning from Time Series Workshop (MiLeTS), August 26, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Sequential tabular data is widely used in the industry to represent financial transactions recorded in a bank database [11], medical records stored by a hospital [20] or customers purchase history maintained in a CRM system [22], to name a few examples. Such tabular data are composed of rows and columns, each row corresponding to a record which collects values for each column. Different from traditional multivariate time-series, tabular time-series often present categorical variables. Unlike classic tabular data which considers separate rows as distinct input samples for a given downstream task, records in sequential tabular data span multiple rows, a property that can be exploited when time-related fields are present (see Table 1). Common examples of tabular time-series tasks take multiple rows in input and provide some prediction, e.g., detecting fraud from sequences of financial transactions [13], predicting click-through rate from past online behavior [12] or forecasting pollution from historical data [13].

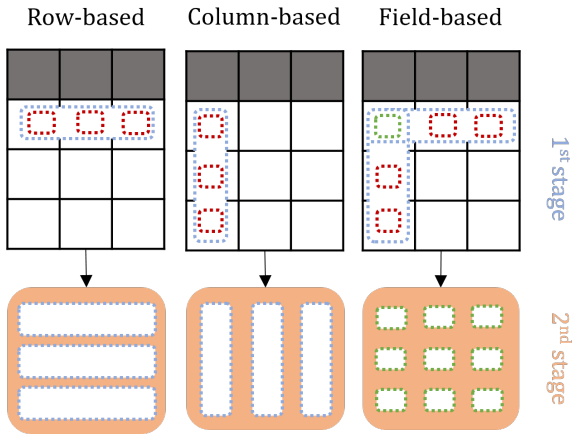

Figure 1: Hierarchical transformers schematic view.

As tabular time-series tasks are intrinsically sequential, the research community started to investigate how to leverage the success of transformer architectures [16] from Natural Language Processing (NLP) within the tabular domain [11, 13, 21]. In a nutshell, by exploiting the attention mechanism, a transformer can relate the tokens that compose a sequence to one another, hence learning relationship patterns between time-steps. This mechanism led to significant improvements in NLP tasks, such as sequence classification (e.g., sentiment analysis), token classification (e.g., entity recognition) or sequence regression (e.g., emotion level prediction). Just as transformers extract rich features from sequences of words, the sequence of records in a table is crucial for extracting meaningful patterns in tabular data modeling.

As tabular data is bi-dimensional, and both rows and columns carry semantics, the transformer architectures available in literature for tabular time-series often present a hierarchical design. In a first stage, each row (or column) is encoded separately by aggregating the outputs of a transformer computing attention across its fields, as shown in Figure 1 – left and center. In the second stage, these encoded rows (or columns) are then passed to another transformer. The result of the second stage is the final encoding of the entire tabular time-series, which is typically processed by additional fully connected layers that specialize the model for solving a given downstream task, e.g., sequence classification. From a tabular perspective, this two-stage process first captures interactions between fields within a given dimension (row or column), and then interactions among those representations.

Table 1: A tabular dataset. Records may be grouped by `Patient` to produce tabular time-series.

Timestamp	Patient	Disease	Therapy	Temperature
2024-06-01	012	Tuberculosis	A	38.2
2024-01-15	012	Flu	B	38.3
2023-12-28	456	Hemophilia	C	37.5
2023-07-26	012	Angina	B	37.9
2023-01-28	456	Sinusite	B	37.3
2022-02-11	789	Flu	D	38.1

While hierarchical architectures capture all table dimensions, they don't do that *simultaneously*, hence limiting visibility on more subtle cross-field relationships important for the downstream task. In Appendix A, we empirically demonstrate that this shortcoming hinders learning interactions between fields across separate rows, due to the coarse-grained aggregation performed in the second stage. This suggests that a *field-wise attention* mechanism can be an appealing alternative to capture the intricate relationships *between all the fields across the full tabular time-series*, as illustrated in Figure 1 – right. In this paper, we implement this mechanism, introducing Fieldy, a novel architecture that combines row-wise and column-wise transformers in the first stage to learn field representations. These contextualized field representations are then merged, reshaped and passed to the second-stage transformer to produce the final encoding of the entire tabular time-series. Consequently, Fieldy enables fine-grained attention across *all* the fields composing a tabular time-series, while incorporating row-level and column-level information. We compare our solution against both state of the art transformer architectures and Machine Learning (ML) tree-based ensemble algorithms using two popular tabular datasets and show that Fieldy outperforms alternative methods.

The remainder of this paper is structured as follows. First, we review prior work on transformers for standard tabular data and tabular time-series (Section 2). Then, we present Fieldy and highlight its differences with row-based and column-based hierarchical transformers (Section 3) before presenting our evaluation protocol and results (Section 4). Last, we discuss our findings and identify areas for future work (Section 5).

2 RELATED WORK

In this section, we first review prior work on Deep Learning (DL) for standard tabular data. We then focus on the specific case of tabular time-series, using Table 1 as toy example. Finally, we discuss how our contribution fits within the existing literature.

Transformers for tabular data. In traditional tabular data modeling, each row is treated as a distinct input sample for which a prediction or a classification needs to be made. A variety of proposals address this type of data using both DL and ML methods. For instance, a recent study [8] explores DL approaches, including CNNs and transformers, as well as ML tree-based models like XGBoost and Random Forests, applied to tabular data. Although the study shows that gradient-boosted trees outperform deep learning algorithms on most datasets, FT-Transformer [7] emerges as a promising architecture. In particular, FT-Transformer encodes each row by computing attention between its fields and then processes it, with a final fully connected layer. FT-Transformer relies on feature tokenizers to embed both categorical and numerical features that may appear in tabular data, which is an uncommon and challenging scenario for ML approaches. Note that this method is row-based but not hierarchical as each record is encoded separately. Alternatively, Tabbie [9] proposes to encode each table field by averaging representations of its row and its column. To do so, two transformers are tasked to encode each row and each column separately, in order to form "contextualized" fields representations by averaging their intersections. Once the full table has been encoded, each row is processed by a final fully connected layer. Note that Tabbie is not a hierarchical architecture, as it only operates at the field-level granularity, either contextualized by row or by column. While Tabbie is not adapted to tabular time-series, as it does not consider subsets of time-dependent rows, it is the closest to our proposal in the way it attends to row and column contexts.

Transformers for tabular time-series. In their most prevalent form, tabular time-series are a specific family of tabular data where records are interdependent and ordered or timestamped (e.g., with an explicit `Timestamp` field as in Table 1), often presenting an "entity" identifier used for grouping records into samples of interest. For example, Table 1 yields three distinct tabular time-series corresponding to three patients' history, each of which could be the input to a machine learning model. In this case, a classic ML algorithm would simply take as input the concatenation of multiple records, hence returning to the traditional tabular data scenario. Conversely, more recent sequence models open the way for using hierarchical approaches to tabular time-series modeling. In this context, TabBERT [13] is composed of a first-stage transformer that encodes each row in a tabular time-series, and a second-stage transformer that processes the encoded rows to generate a full sequence representation. This final representation is then passed to a fully connected layer, e.g., a classification head, to perform the downstream task. Given its design, we refer to this architecture as "row-based". Alternatively, instead of encoding rows in the first stage, a variation of TabBERT may encode each column separately, and process these encoded columns in a second stage to generate the full tabular time-series representation. We refer to this inversion of TabBERT as "column-based". Several works have extended this hierarchical approach to model tabular time-series, such as [11] that extends it to heterogeneous tabular time-series, i.e., with different number of columns, and [21] that exploits time-deltas to better model time differences in the context of tabular time-series. Our work builds on hierarchical tabular time-series modeling, extending it with the ability to integrate field interactions across both rows *and* columns simultaneously.

Table 2: Transformer-based models comparison.

Model	Architecture	Attention axis
FT-Transformer [7]	Single-stage	Horizontal
Tabbie [9]	Single-stage	Horizontal & vertical
TabBERT *(row-based) [13]*	Two-stage	Horizontal → Vertical
TabBERT *(column-based)*	Two-stage	Vertical → Horizontal
Fieldy *(ours)*	Two-stage	Horizontal & vertical → Fields

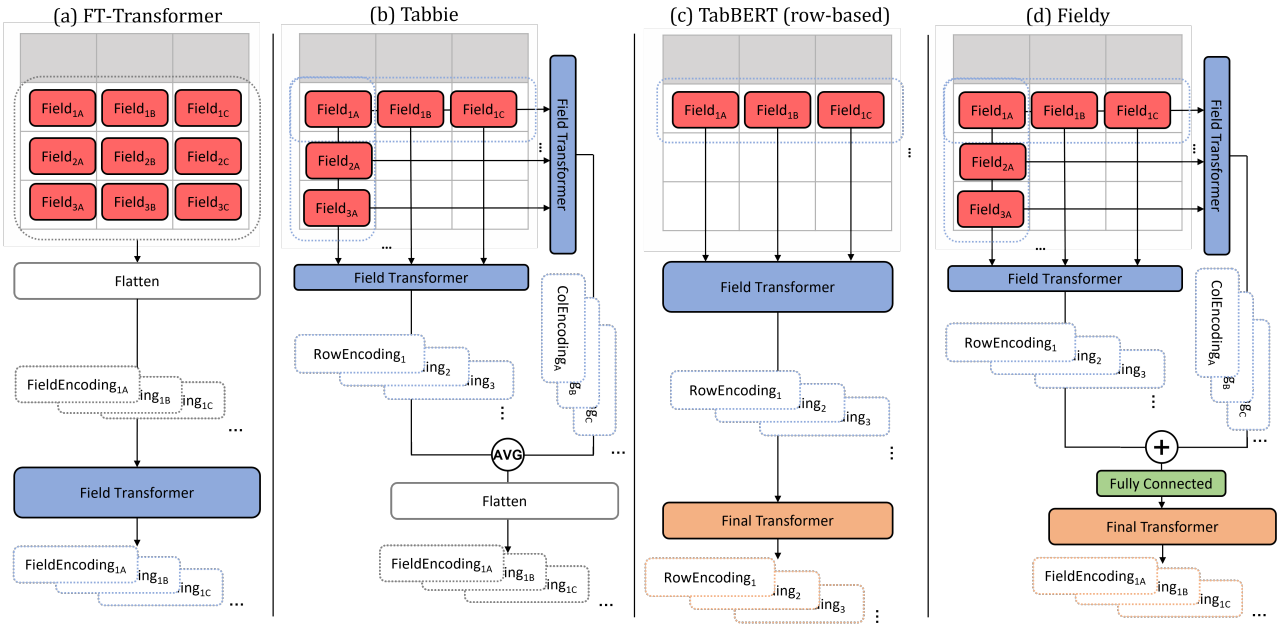

Figure 2: Detailed view of transformer architectures for tabular time-series. The ⊕ operator denotes concatenation.

Our contribution. In this paper we set out to evaluate whether a hierarchical model with a finer cross-field attention provides more appropriate representations of tabular time-series, as opposed to a coarse aggregation of row or column embeddings. In a nutshell, we propose an architecture that simultaneously captures row-wise and column-wise interactions in a first stage, to learn contextualized field representations that are related to one another in a second stage. To properly isolate this effect and focus the comparison on this particular design choice, we compare this field-based attention approach to its row-based and column-based counterparts in similar conditions. In other terms, in this paper, we do not aim to introduce novel tokenizers or novel pre-training tasks which are orthogonal directions of research. Additionally, since our approach relates any field to any other during the second stage, unlike row-based and column-based, we compare it also to a single-stage baseline where all the rows of interest are flattened and fed to a unique transformer. This approach can be seen as an adaptation of FT-Transformer for tabular time-series, effectively resulting in a comparison point that links any field to any other. Finally, to complete the design space, another interesting comparison point we consider consists in constructing contextualized field embeddings that incorporate both rows and columns and then feeding the flattened sequence to a linear layer. This approach could be seen as a straightforward adaptation of Tabbie to the context of tabular time series.

3 METHODOLOGY

In this section, we first dive into the details of current approaches and their limitations. We then present our field-based hierarchical approach that integrates both row-wise and column-wise interactions. We finally introduce the positional embeddings schemes used to encode table structure.

State of the art limitations. Tabular time-series modeling approaches considered in this paper are summarized in Table 2. Non-hierarchical baselines such as FT-Transformer and Tabbie are single-stage architectures designed for traditional (i.e., non sequential) tabular data. When adapting FT-Transformer to tabular time-series, the input sequence is flattened to form a single long row of repeated features at various time-steps, as depicted in Figure 2 (a). As this model computes attention across all the fields composing a single row, it captures relationships between all the fields across all rows and columns as a result of the input flattening. However, the downside of this flattening is that this adaptation of the FT-Transformer model is oblivious to the table structure. We consider this as a baseline transformer attending to all fields. Alternatively, when adapting Tabbie to ingest tabular time-series, we limit its row-wise contextualization to a subset of the table: the rows composing the input sequence. As it relates each field to other fields present in the same row and column, Tabbie is able to capture relationships along both table axes. However, as depicted in Figure 2 (b), this approach is not equipped with a second-stage to relate all the fields outside of their original row and column.

Regarding hierarchical models, recall that two-stage architectures from the literature are either "row-based" or "column-based". Thus, they condition the attention mechanism between distinct fields along a particular table axis in the first stage, as illustrated in Figure 2 (c). While these approaches enforce the tabular time-series row (or column) structure, they fail to capture relationships between fields across separate rows (or columns). For instance, considering Table 1, a row-based architecture cannot explicitly relate, i.e., by means of attention, a `Disease` field to a `Therapy` field if they belong to different rows, nor can it directly relate two `Disease` fields that belong to two different rows. As shown empirically in

Appendix A, this limits the ability of hierarchical models from the literature to learn fine-grained relationships at the field level that might be relevant for a given downstream task.

Field-wise attention. Two-stage approaches rely on the assumption that the Field transformer in the first stage is sufficiently powerful to extract expressive row/column representations for the Final transformer, rather than fostering the learning of fine-grained field relationships in the second stage. To capture relevant fields interations that may be beneficial to tabular time-series tasks, we introduce Fieldy: a novel hierarchical transformer that combines both in-row and in-column attention, as well as cross-field attention. As depicted in Figure 2 (d), we propose a two-stage architecture in which the first stage consists of two Field transformers that operate simultaneously: one is responsible to contextualize each field row-wise, and the other is responsible to contextualize each field column-wise. The resulting encoded rows and columns are then concatenated to constitute field representations. These contextualized cell representations are then passed through a fully connected layer to produce rich representations before being processed by a Final transformer, which would attend to all fields. Note that our first stage resembles Tabbie; yet, it differs for two key design choices. First, while Tabbie creates a field embedding through a deterministic simple average of the field's row and column embeddings, Fieldy *learns* how to combine the two embeddings in the first stage. Second, unlike Tabbie which passes the embedded fields to the final fully connected layer (e.g., a classifier head), we adopt a second-stage transformer that relates all the field representations to each other. At last, the entire encoded tabular time-series coming from the Final transformer is processed by a fully connected layer, fine-tuned on a specific downstream task (e.g., sequence classification). In order to fairly compare Fieldy against row-based and column-based architectures, we reduce the size of its Field transformers to reach the same model size. Nonetheless, we note that Fieldy requires increased computational effort, an aspect which we discuss in Section 5.

Positional encoding to capture table structure. Compared to row-based or column-based hierarchical approaches that model the table structure by design, Fieldy requires additional information. In the row-based architecture, the Field transformer implicitly establishes the table's horizontal structure by computing attention horizontally across all the fields of the same row. The Final transformer then enforces the table vertical structure as it ingests a sequence of per-row representations. Conversely, in the column-based architecture, the Field transformer is provided with the table's vertical structure, as its input is a sequence of fields from distinct rows. Then, the Final transformer enforces the table horizontal structure by attending between per-column representations. For these two approaches, it is only the *position* of the rows (or columns) that is unknown from the model, but their delimitation is apparent by design. On the other hand, the design of Fieldy does not incorporate the table structure by default, as the Final transformer ingests a long sequence composed of all the contextualized fields. Thus, without additional information, the Final transformer can only access a bag of fields, being oblivious to the original rows or columns they come from. Therefore, we incorporate row and column positional embeddings [4]. Before being passed to the Final transformer, each contextualized field is augmented (by means of element-wise addition) with two embeddings: one carrying its original row position and one carrying its original column index. To ensure a fair comparison between architectures, we also add these row position and column index embeddings to all the other models. We discuss the effect of these additions in Section 4.

4 EVALUATION

In this section, we first detail the datasets and models configurations considered to compare transformers architectures on tabular time-series. Then, we present the results of our evaluation and expand our analysis with an ablation study.

4.1 Datasets

Pollution – Regression. The UCI Beijing Pollution Dataset [2] consists in predicting air pollution particles from 12 sites located in Beijing. This dataset has been used to evaluate row-based hierarchical transformers in [11, 13]. It is a multi-regression task taking as input 10 features (such as temperature, pressure, etc.) measured on an hourly basis during 10 time-steps. The labels to predict correspond to the $PM_{2.5}$ and PM_{10} concentrations for each time-step, i.e., 20 labels for each input sequence. We replicate the pre-processing described in [13], i.e., we discretize numerical variables using 50 quantiles and normalize the targets. After data exploration, we choose to include 6 additional features not present in the pre-processing from related work. These engineered features correspond to the measurement site name, the hour of the measurement, the day of the month, the weekday, the month, and the year. Finally, we remove 4% of outliers when $PM_{10} > PM_{2.5}$ as mentioned in [18]. As in prior work, the evaluation metric is defined as the RMSE averaged across the concentration targets. In total, this dataset contains 67K tabular time-series. The pre-training dataset consists of the same dataset excluding the labels (more details on data splits in Appendix C).

Loan default – Classification. The PKDD'99 Bank transactions dataset [1] contains real transaction records for 4,500 clients of a Czech bank. It has been used in [11] to evaluate hierarchical transformers. The considered task consists in predicting if a client will default its loan based on his prior transactions. Six input features describe each transaction (amount, type, etc.) and the label is binary, i.e., one for each clients' input sequence. As for the pollution dataset, we pre-process the data similarly to prior work [11], i.e., using 50 quantiles to discretize each numerical feature and splitting the timestamp into three fields: day, month and year. Note that we include an additional weekday feature that proved meaningful during data exploration. As in [11], we segregate the data into 3,818 clients with unlabeled transactions for pre-training, and 682 clients with labeled transactions for fine-tuning. In order to increase the dataset size, we consider any sequence of 10 consecutive transactions for each client, while [11] considered only his last 150 transactions. We thus obtain 5K tabular time-series for fine-tuning, instead of only 682. The evaluation metric is defined as the Average Precision (AP) score to take into account the class imbalance of this dataset (more details on data splits in Appendix C).

4.2 Models

Architectures. For each dataset, we consider three hierarchical transformers: row-based, column-based and our field-based proposal Fieldy. We use the official code from TabBERT [13] to implement row-based and column-based baselines. Additionally, we evaluate two single-stage baselines: FT-Transformer and Tabbie, using our own implementation to adapt them to tabular time-series. We size all models to amount to the same total number of parameters. As Fieldy requires two Field transformers in its first stage, we reduce their number of layers to make the comparison fair. We keep all the other hyper-parameters related to model capacity (hidden dimensions, number of attention heads, etc.) the same across all models, similar to prior work [11, 13, 21]. Models are pre-trained for 24 epochs (*Pollution*) or 60 epochs (*Loan default*), and fine-tuned for 20 epochs. The best models are selected based on their score on a validation set and evaluated on a held-out test set. Additionally, we include two non-deep learning baselines: XGBoost [3] and a linear model (linear or logistic regression). Note that these two baselines use exactly the same pre-processed input features and labels as the hierarchical transformers, which means that all numerical features are quantized. Both of these baselines take as input a flattened version of the tabular time-series. We select the best linear models after a cross-validated random search of 50 iterations to select their hyper-parameters. Also, regarding the Loan default prediction task, note that only the fine-tuning portion of the Loan dataset is considered for these non-deep learning baselines. More details on hyper-parameters can be found in Appendix B.

Comparability. To ensure a fair comparison across all models, we implement the same pre-training and fine-tuning strategy for all of them. While self-supervised pre-training has proved useful on tabular data [14], we emphasize that our objective is not to design novel pre-training techniques. Thus, we consider a simple field masking pretext task for all models, as from previous literature [13]. In detail, we guide pre-training with a BERT-like token masking pretext task [4], randomly selecting 15% of the tokens, out of which 80% are replaced by a `[MASK]` token, 10% by a random token and 10% left unchanged. Last, as our objective is not to introduce novel fine-tuning mechanisms, we resort to a standard fine-tuning technique popularized in NLP. During fine-tuning, we prepend a `[CLS]` token to the tabular time-series before encoding it with the model (i.e., before the Final transformer for two-stage models). Once the full time-series is encoded, this special token is extracted and passed to a final fully connected layer trained on a specific downstream task as in [4]. We implement this fine-tuning methodology for all models, with a variation for Tabbie. Indeed, in [9], the authors suggest to prepend `[CLS]` tokens to each row and column processed by Tabbie, based on the downstream task to learn. Thus, we consider the version of Tabbie that yields the best results in our experiments.

4.3 Results

We evaluate each model over 5 seeds runs and report their average performance and standard deviation in Table 3. We emphasize that transformer-based models contain the same total number of parameters. On the Pollution dataset, we observe that Fieldy significantly decreases the RMSE demonstrating the effectiveness of the proposed approach. Regarding the Loan default prediction task instead, we observe that the differences in terms of AP are less significant, likely due to the smaller dataset size. Our results for TabBERT (row-based) on the Pollution dataset are similar to the ones reported in [11, 13]. Regarding the Loan default prediction task, given that we reduce the sequence length to any 10 consecutive transactions to generate more fine-tuning samples, we cannot directly compare our results to [11] that only used the last 150 transactions instead, at the expense of generating fewer samples to train on.

Table 3: Results. Average over 5 seed runs, standard deviation in parenthesis. Models have the same number of parameters.

Model	Architecture	Pollution RMSE ↓	Loan Avg. Precision ↑
Linear	Non-DL	59.44 (0.28)	0.31 (0.03)
XGBoost	Non-DL	50.74 (0.59)	0.36 (0.07)
FT-Transformer	Single-stage	26.54 (0.45)	0.44 (0.07)
Tabbie	Single-stage	22.37 (0.31)	0.39 (0.05)
TabBERT *(col-based)*	Two-stage	26.46 (0.32)	0.44 (0.05)
TabBERT *(row-based)*	Two-stage	21.05 (0.22)	0.46 (0.06)
Fieldy *(ours)*	Two-stage	**20.13** (0.34)	**0.48** (0.06)

We first remark that all transformer models outperform the non-deep learning baselines we evaluate.[1] Note that the Pollution prediction task requires to output 20 labels for each input sample, which is implemented with a multi-output regressor wrapped on around these non-deep learning baselines, i.e., fitting one model for each target. Hence, this limits these baselines' ability to capture relationships between targets. Also, for the Loan default prediction task, the performance gap is partially explained by the pre-training advantage the transformers models are given, as the non-deep learning baselines only use the smaller fine-tuning dataset.

Among transformer models, single-stage baselines underperform compared to two-stage architectures, highlighting the benefit of hierarchical representations for tabular time-series. In particular, our field-based proposal ranks first on both datasets. In contrast, TabBERT which conditions attention between fields on a unique table axis, yields worse performance indicating that capturing fields relationships across rows *and* columns is important. Additionally, comparing Fieldy to the flattened FT-Transformer hints at the lack of table structural information for the latter. Last, while Tabbie structures the field embeddings contextualization row-wise and column-wise, its lack of a second-stage fails to relate all of these representations.

4.4 Ablation study

In Table 4, we analyze the effect of various design decisions on the performance of each transformer model on the Pollution and Loan default datasets. Note that, while the results hold qualitatively for both datasets, quantitative analysis on the Loan task might be affected by the smaller dataset size. In the remainder, we derive conclusions paying more attention to the Pollution dataset results.

[1] Although prior work demonstrated in many experiments the superiority of gradient-boosted decision trees compared to transformer-based approaches, in our setting, XGBoost might suffer from using quantized numerical features. This is a consequence of our experimental protocol choices which favor comparability with existing literature (especially for transformer-related works) rather than searching for the global best independently.

Table 4: Ablation study. Average over 5 seed runs, standard deviation in parenthesis. Underline highlights model families best configuration, while **bold** highlights the global best.

Model family	Stage with more capacity	Column ind. emb.	Row pos. emb.	Pollution RMSE ↓	Loan AP ↑
FT-Transf.	Single-stage			28.28 (0.27)	0.43 (0.08)
		✓		28.04 (0.22)	0.42 (0.08)
			✓	27.80 (0.73)	0.44 (0.07)
		✓	✓	26.54 (0.45)	0.42 (0.05)
Tabbie	Single-stage			22.37 (0.31)	0.38 (0.06)
		✓		22.43 (0.14)	0.39 (0.03)
			✓	23.14 (0.23)	0.38 (0.03)
		✓	✓	23.02 (0.14)	0.39 (0.05)
TabBERT *(col-based)*	Field Transf.			27.10 (0.32)	0.44 (0.05)
		✓		27.08 (0.32)	0.43 (0.04)
			✓	26.46 (0.32)	0.40 (0.08)
		✓	✓	26.48 (0.28)	0.42 (0.03)
	Final Transf.			27.85 (0.35)	0.37 (0.03)
		✓		27.88 (0.30)	0.38 (0.02)
			✓	27.19 (0.27)	0.38 (0.04)
		✓	✓	27.23 (0.22)	0.36 (0.04)
TabBERT *(row-based)*	Field Transf.			21.30 (0.28)	0.44 (0.04)
		✓		21.07 (0.15)	0.46 (0.06)
			✓	21.34 (0.28)	0.44 (0.05)
		✓	✓	21.05 (0.22)	0.45 (0.07)
	Final Transf.			22.92 (0.32)	0.46 (0.05)
		✓		22.72 (0.26)	0.44 (0.07)
			✓	22.92 (0.30)	0.45 (0.04)
		✓	✓	22.70 (0.29)	0.45 (0.07)
Fieldy *(ours)*	Field Transf.			20.48 (0.19)	0.46 (0.07)
		✓		**20.13** (0.34)	0.44 (0.09)
			✓	20.42 (0.22)	0.43 (0.08)
		✓	✓	20.32 (0.30)	**0.48** (0.06)
	Final Transf.			24.15 (0.17)	0.41 (0.04)
		✓		24.00 (0.24)	0.42 (0.08)
			✓	23.98 (0.22)	0.40 (0.06)
		✓	✓	24.06 (0.27)	0.40 (0.06)

First, we investigate the partition of model capacity between the first stage, i.e., Field transformer, and the second stage, i.e., Final transformer, for hierarchical models. Namely, we modify the number of encoder layers implemented in each stage to either favor one or the other, while ensuring the total number of model parameters remains the same (cf. Table 7). We observe that hierarchical architectures perform better when favoring the first stage. This is particularly evident for Fieldy with up to +16% performance improvement for the Pollution data set. This common trend indicates that the field representations learned in the first stage are particularly important for hierarchical transformers performance.

Second, we analyze the effect of the table structure encoding mechanisms that we introduced in Section 3, namely using column index embedding and/or row position embedding. Note that, even for architectures that dissect tabular time-series along table axes (i.e., Tabbie and TabBERT), the *ordering* of rows and columns is not preserved by design, due to the permutation-invariant nature of the basic attention mechanism, thus requiring additional positional input. Therefore, we evaluate all table structure encoding combinations for all models. We observe that explicitly indicating table structure information is beneficial to almost all transformer architectures, compared to not including any. However, the type of structural information required (i.e., column index, row position, or both) is dependent on the model family. As expected, FT-Transformer benefits from structure information on both axes. Tabbie shows a similar trend on the Loan default prediction task, although differences are less significant. Regarding two-stage models, positional encodings are particularly important for the table axis along which field representations are aggregated. Hence, column-based TabBERT benefits from row position information, conversely, row-based TabBERT exploits column index information. Fieldy exploits table structure information on both axes, and particularly column index embeddings.

5 CONCLUSION AND DISCUSSION

In this paper we compared transformer architectures for tabular time-series modeling, investigating how attention can be used to simultaneously relate tabular fields across rows *and* columns. We evaluated our approach on tabular regression and classification tasks, showing improvements over existing baselines. However, we highlight that this work can be further expanded. In particular, we envision two possible research directions considering computational costs and more data variety.

Computational requirements. While we ensure that in our experiments all transformer models have the same total number of parameters, their computational requirements differ. In particular, "vanilla" self-attention time complexity is $O(L^2)$ where L is the input sequence length. However, the transformer models we compare do not consider the same input sequence, as they compute attention along various table axes, i.e., row-wise, column-wise or both. In Table 5, we compare attention complexity for each architecture, based on the number of rows and columns composing the tabular time-series. We note that the column-based and row-based hierarchical models are equivalent in terms of attention complexity. In contrast, our field-based proposal is essentially combining Tabbie-like attention in the first stage and FT-Transformer attention between all fields in the second stage. Hence, this finer-grained attention comes at a higher computational cost, which translates into longer training and inference time. Nevertheless, recent techniques for near linear-time attention [15, 17, 19] can be readily used to speed up our approach.

Table 5: Attention complexity. n_{rows} and n_{cols} denote the number of rows and columns per tabular time-series.

Model	Stage	Complexity
FT-Transformer	Single-stage	$O((n_{rows} \times n_{cols})^2)$
Tabbie	Single-stage	$O({n_{rows}}^2 + {n_{cols}}^2)$
TabBERT *(col-based)*	1st – Field Transf.	$O({n_{rows}}^2)$
	2nd – Final Transf.	$O({n_{cols}}^2)$
TabBERT *(row-based)*	1st – Field Transf.	$O({n_{cols}}^2)$
	2nd – Final Transf.	$O({n_{rows}}^2)$
Fieldy *(ours)*	1st – Field Transf.	$O({n_{rows}}^2 + {n_{cols}}^2)$
	2nd – Final Transf.	$O((n_{rows} \times n_{cols})^2)$

Future work. To further characterize the strengths and weaknesses of Fieldy, we require additional evaluations on a more diverse set of tasks and datasets. In particular, the size of such datasets require careful consideration, given that over-parameterized deep learning models typically tend to overfit small datasets. In this regard, click-through rate data [5] may be of interest for their sequential nature and large volume. We did not include such dataset in this work because it is not considered in related literature either. Yet, given its larger size, it would be worth considering in a future work. Also, our findings have only been evaluated in the realm of tabular time-series, however, the wider domain of multivariate time-series might also benefit from field-based hierarchical architectures, combining length and channel signals in a first stage. Finally, as the objective of this paper is limited to the comparison of attention computation axes, we do not include sophisticated tabular data embeddings techniques such as numerical features embeddings from [6, 11] or target-aware pretext tasks for pre-training from [14]. We expect their respective benefits to be portable to our novel field-based architecture, as they impact the prior embedding layer of any transformer architecture. Regarding numerical features embedding, a tangential direction may leverage methods from traditional time-series pre-processing techniques. In this regard, SAX [10] bins continuous time-series into sequences of discrete symbols to capture their trends. Such symbols could then be tokenized before being fed to a transformer-based architecture. This motivates an interesting research direction to compare numerical features embeddings schemes in the context of tabular time-series.

We contribute to the community our codebase and models at https://github.com/raphaaal/fieldy.

6 ACKNOWLEDGMENTS

We would like to thank the authors of [11] for fruitful discussions on the success metrics and pre-processing they used.

REFERENCES

[1] Petr Berka. 1999. PKDD'99 Discovery challenge guide to the financial data set. http://lisp.vse.cz/pkdd99/chall.htm.

[2] Song Chen. 2019. Beijing Multi-Site Air-Quality Data. UCI Machine Learning Repository. https://doi.org/10.24432/C5RK5G.

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

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Diemert Eustache, Meynet Julien, Pierre Galland, and Damien Lefortier. 2017. Attribution modeling increases efficiency of bidding in display advertising. In *Proceedings of the AdKDD and TargetAd Workshop*.

[6] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. In *Advances in Neural Information Processing Systems*. 24991–25004.

[7] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*. 18932–18943.

[8] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data?. In *Advances in Neural Information Processing Systems*. 507–520.

[9] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584* (2021).

[10] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15 (2007), 107–144.

[11] Simone Luetto, Fabrizio Garuti, Enver Sangineto, Lorenzo Forni, and Rita Cucchiara. 2023. One Transformer for All Time Series: Representing and Training with Time-Dependent Heterogeneous Tabular Data. *arXiv preprint arXiv:2302.06375* (2023).

[12] Aashiq Muhamed, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. 2021. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*.

[13] Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. 2021. Tabular transformers for modeling multivariate time series. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3565–3569.

[14] Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. 2022. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*.

[15] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient Attention: Attention With Linear Complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 3531–3539.

[16] 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*.

[17] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: self-Attention with linear complexity. *arXiv preprint arXiv:2006.04768v3* (2020).

[18] Huangjian Wu, Xiao Tang, Zifa Wang, Lin Wu, Miaomiao Lu, Lianfang Wei, and Jiang Zhu. 2018. Probabilistic automatic outlier detection for surface air quality measurements from the China national environmental monitoring network. *Advances in Atmospheric Sciences* 35 (2018), 1522–1532.

[19] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Moo Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A Nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), 14138–14148.

[20] Dongyu Zhang, Jidapa Thadajarassiri, Cansu Sen, and Elke Rundensteiner. 2020. Time-aware transformer-based network for clinical notes series prediction. In *Machine learning for healthcare conference*. PMLR, 566–588.

[21] Dongyu Zhang, Liang Wang, Xin Dai, Shubham Jain, Junpeng Wang, Yujie Fan, Chin-Chia Michael Yeh, Yan Zheng, Zhongfang Zhuang, and Wei Zhang. 2023. FATA-Trans: Field And Time-Aware Transformer for Sequential Tabular Data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. 3247–3256.

[22] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* (2019), 1–38.

Temp.	...	Pressure	Hour
2.8	...	992.2	13
4.6	...	992.9	14
8.4	...	994.1	15
16.3	...	998.3	16
16.7	...	1001.3	17
[MASK]	...	[MASK]	?
[MASK]	...	[MASK]	?
[MASK]	...	[MASK]	?
[MASK]	...	[MASK]	?
[MASK]	...	[MASK]	?

Figure 3: Illustrative input sample for the field-wise attention toy task. Predicting missing tokens in the `Hour` column requires field-wise attention across rows.

Appendix A FIELD-WISE ATTENTION

In this section, we investigate the hypothesis that coarse-grained attention resulting from the typical hierarchical learning of tabular time-series representations might miss important cross-field relationships. More precisely, we compare a pre-trained TabBERT (row-based architecture) and a pre-trained Fieldy (field-based architecture) on a simple prediction task that specifically requires the attention mechanism to focus on field dependencies *across* rows. For this, we utilize the Pollution dataset introduced in Section 4, sampling 100 input sequences of 10 rows each. We then `[MASK]` all values in the last five rows of each sample, and task the models to predict the masked tokens in the `Hour` column, as depicted in Figure 3. Note that values in the `Hour` column are always incremented by +1, as the Pollution measurements are hourly-based and the sequences are ordered. Hence, predicting the five missing hour values requires to specifically attend to `Hour` fields across separate rows. From this simple experiment, considering the top-1 predicted token, Fieldy (field-based) achieves an accuracy of 56%, significantly outperforming TabBERT (row-based) which only scores 9%. This indicates that the aggregation performed by hierarchical models such as TabBERT limits their ability to relate fields along their second-stage axis.

Appendix B HYPER-PARAMETERS

Transformers architectures hyper-parameters are reported in Table 6. Note that we increase dropout and decrease the hidden dimension for the Loan prediction task, as it is composed of fewer samples and models tend to overfit. We adjust the model size by selecting an appropriate number of layers as reported in Table 7. We keep all the other hyper-parameters related to model capacity the same across all models for a fair comparison.

Regarding XGBoost, we run random searches to find hyper-parameters, whose possible values are reported in Table 8. We use a 2-fold cross-validation on the Pollution dataset and a 10-fold one on the Loan dataset that is significantly smaller. Both datasets were granted a sampling budget of 50 iterations for each seed run.

Table 6: Hyper-parameters for transformer-based models.

Setup	Pollution	Loan
Pre-training epochs	24	60
Fine-tuning epochs	10	20
Optimizer	AdamW	AdamW
Learning rate	5e-05	5e-05
Batch size	64	100
Dropout	0.1	0.3
Hidden dimension	800	500
Number of attention heads	10	10
Number of parameters	≈106M	≈36M

Table 7: Transformer-based models layers statistics.

Model family	Stage with more capacity	Num. of layers Pollution	Num. of layers Loan
FT-Transformer	Single-stage	14	8
Tabbie	Single-stage	4	4
TabBERT *(col-based)*	Field Transf.	6 / 10	6 / 6
	Final Transf.	1 / 12	1 / 8
TabBERT *(row-based)*	Field Transf.	6 / 10	6 / 6
	Final Transf.	1 / 12	1 / 8
Fieldy *(ours)*	Field Transf.	8 / 4	5 / 4
	Final Transf.	2 / 10	2 / 6

Parameter count per layer depends on the architecture. For hierarchical architectures, values reflect 1^{st}-stage layers / 2^{nd}-stage layers.

Table 8: Hyper-parameters for XGBoost random search.

Parameter	Pollution	Loan
Objective	MSE	Logistic
Max # trees	5,000	5,000
Early-stopping	50	50
Max depth	[1, 2, ..., 20, None]	[1, 2, ..., 20, None]
Learning rate	LogUniform(1e-05, 0.7)	LogUniform(1e-05, 0.7)
Min child weight	LogUniform(1e-08, 100)	LogUniform(1e-08, 100)
Subsample	Uniform(0.5, 1)	Uniform(0.5, 1)
Col. sample	Uniform(0.5, 1)	Uniform(0.5, 1)
Gamma	LogUniform(1e-08, 100)	LogUniform(1e-08, 100)
Reg. alpha	LogUniform(1e-08, 100)	LogUniform(1e-08, 100)
Reg. lambda	LogUniform(1e-08, 100)	LogUniform(1e-08, 100)

Appendix C DATASETS

In Table 9, we describe the two datasets used for evaluation. For the Pollution task, the same dataset is used for pre-training (unlabeled) and fine-tuning. For the Loan default prediction task, the pre-training dataset is composed of 4,500 clients with 232 transactions on average. At each pre-training step, random sequences of 10 consecutive client transactions are sampled for masking. The fine-tuning dataset for this task corresponds to transactions from 682 clients. As in prior work [11], splitting between training, validation and test sets is performed on a client-basis to avoid target leakage.

Table 9: Datasets statistics.

Item	Pollution	Loan
Time-series # rows	10	10
Time-series # columns	16	10
# categorical columns	7	7
# numerical columns	9	3
Pre-training samples	67K	≈999K
Pre-training split (train-val-test)	60-20-20	80-20-0
Fine-tuning samples	67K	5K
Fine-tuning split (train-val-test)	60-20-20	60-20-20