Banksformer: A Deep Generative Model for Synthetic Transaction Sequences

Kyle Nickerson[1(✉)], Terrence Tricco[1], Antonina Kolokolova[1],
Farzaneh Shoeleh[2], Charles Robertson[2], John Hawkin[2], and Ting Hu[3]

[1] Memorial University of Newfoundland, St. John's, NL, Canada
kln870@mun.ca
[2] Verafin Inc., St. John's, NL, Canada
[3] Queen's University, Kingston, ON, Canada

Abstract. Synthetic data are generated data that closely model real-world measurements, and can be a valuable substitute for real data in domains where it is costly to obtain real data or privacy concerns exist. Synthetic data has traditionally been generated using computational simulations, but deep generative models (DGMs) are increasingly used to create high-quality synthetic data. In this work, we tackle the problem of generating synthetic, multivariate sequences of banking transactions.

A key challenge in modeling transactional sequences with DGMs is that transactions occur at irregular intervals and may depend on timestamp-based features, such as the time of day or day of the week. Relationships between date-based features are often poorly represented in data generated using state-of-the-art sequence DGMs, such as DoppelGANger [17] and TimeGAN [31]. To remedy this, we propose a novel DGM, called Banksformer (Code available at github.com/BigTuna08/Banksformer_ecml_2022), which is able to emulate date-based patterns found in transactional data significantly better than other DGMs. We demonstrate Banksformers' ability to generate high-quality synthetic sequences of banking transactions by conducting a multi-faceted evaluation that compares synthetic data generated by Banksformer to data from other comparable DGMs, across two datasets of banking transactions.

Keywords: Synthetic data · Deep generative models · Transaction sequences

1 Introduction

Synthetic data are becoming an increasingly important component in machine learning systems. Recent work has demonstrated the ability of deep genera-

We wish to acknowledge the support of Mitacs through Accelerate funding for applied research.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-26422-1_8.

M.-R. Amini et al. (Eds.): ECML PKDD 2022, LNAI 13718, pp. 121–136, 2023.
https://doi.org/10.1007/978-3-031-26422-1_8

tive models (DGMs) to produce high-quality synthetic data in domains such as images [10], text [3], and audio [6]. Each of these domains has presented unique challenges, which were addressed by modifying model architectures from previous tasks to be more suited to the target task. Success in these general domains has led to the creation of focused, domain-specific models. One domain that has received considerable recent interest is financial data.

Financial data is a broad category, however most existing work on DGMs in finance focuses on modeling price sequences for stocks and other financial instruments [13,25,29]. Another important type of financial data is transactional data; that is, data that contains sequences of records or transactions recorded at arbitrary intervals. Transactional data is common in finance but also occurs in other domains. For example, both a sequence of purchase records from a credit card and a sequence of entries in electronic health records are transactional. In general, modeling transactional data is more challenging than other time-series data, as we must learn to model the intervals between transactions in addition to the transaction features. This can be particularly challenging in a domain such as banking, where the date and time of a transaction can be strongly related to the transaction type and amount. Further, certain types of dates, such as the weekends or the end of the month, can significantly influence what transactions occur.

Evaluating the quality of synthetic data is a difficult problem without a single clear solution [1,9,26]. Ideally, we would like to measure a distance between the real and synthetic data distributions; however, this is not feasible for multidimensional sequence data. A seemingly general approach would be to use the log-likelihood the generative model assigns to validation data. Unfortunately, this approach is known to have issues [26], and also depends on the model being able to assign likelihood scores, which is possible for transformers but not generative adversarial networks (GANs) [8]. Existing work generating financial time series is limited but commonly evaluates the quality of generated data by comparing univariate features distributions [13,29]. However, these univariate metrics only give a rough picture of the synthetic data quality. These metrics cannot measure how well the synthetic data captures feature interactions and interactions between sequence elements.

The main goal of this work is to produce high-quality synthetic financial transaction sequence datasets, with the same statistical properties as real data upon which they are based. We propose Banksformer (BF), a novel transformer-based DGM designed to model transactional data with date-based patterns. GANs have typically been used as the generative model in previous work generating sequential financial data [13,25,29]. To demonstrate the benefits of our approach, we compare BF against two high-quality GAN models – TimeGAN (TG) [31] and DoppelGANger (DG) [17] – on two datasets of banking transaction sequences.

2 Datasets

We used two datasets of banking transactions to compare the quality of synthetic data produced by BF with data produced by TG and DG. The first is a set of real banking data from the Czech Republic in the 1990s[1] (*czech*), and the second is a synthetic dataset of transactions from the UK in 2017[2] (*uk*). Both datasets contain transaction records from many different bank accounts, with the *uk* dataset containing 5 000 unique accounts, and the *czech* containing 4 500 accounts. Each transaction contains the dollar value of the transaction, multiple categorical codes that have information about the transaction type, and a timestamp indicating when the transaction occurred. To create a uniform representation between datasets, we concatenate together all categorical codes into a single field called the *tcode* (transaction code). In the *czech* data there are 16 unique tcodes, and the *uk* dataset has 44 (Table 1). The timestamp in the *czech* dataset only contains the transaction date, and not the specific time of day. Because of this, we do not use the time of day information in the *uk* dataset and focus only on modeling the transaction dates.

Table 1. Dataset Summary. Properties of the *czech* and *uk* data sets. Columns show the number of unique accounts (Accts), total number of transactions (Total Trans), statistics on the number of transactions per account (Trans per Acct), number of unique transcations codes (Tcodes), and the date range.

	Accts	Total trans	Trans per Acct			Tcodes	Date range	
	Count	Count	Min	Max	Mean	Count	Start	End
Czech	4500	1.06×10^6	9	675	235	16	01/01/1993	31/12/1998
UK	5000	10^5	2	50	20	44	01/04/2017	25/05/2017

We are primarily interested in the *czech* dataset, which was initially made available as part of the Discovery Challenge at the 1999 PKKD conference [23]. This dataset is likely to lead to more meaningful results than the *uk* dataset for three main reasons. First, the *czech* dataset contains real banking data. This is in contrast to the *uk* dataset, which is a synthetic dataset. Second, the *czech* dataset contains over 1 M transactions, making it over ten times larger than the *uk* dataset, which has only 100K transactions. Because the datasets have a similar number of unique accounts, this means there are comparatively fewer transactions per account in the *uk* dataset (Table 1). Finally, the *uk* data is also from a much smaller range of dates, containing less than two months' data, whereas the *czech* dataset spans five years. Transactional banking data often contains date-based patterns, which can be difficult for DGMs to emulate. In the *uk* dataset, the most significant date-based patterns are related to the day of the week. In that dataset, transactions never occur on Sunday. Further, certain types of transactions are related to the day of the week, and happen more or less often on certain days. Because the *uk* dataset spans less than two full months,

[1] https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions.

[2] https://pub.towardsai.net/generating-synthetic-sequential-data-using-gans-a1d67a7752ac; this blog post explores using DG to create synthetic data.

we do not consider patterns related to the day of the month. In contrast, the *czech* dataset does not contain any apparent relationships involving the day of the week. However, in the *czech* data there are clear patterns related to the day of the month, with certain types of transactions only occurring at the month's end, and others only happening early in the month.

We transform this dataset of transactions into transaction sequences by grouping together transactions by account, and then sorting the transactions for each account by date (and time in the *uk* dataset). In order to create more uniform datasets, we filtered out sequences shorter than a minimum length parameter l_{min} (5 for *uk* and 20 for *czech*), and split sequences longer than l_{max} (20 for *uk* and 80 for *czech*) into multiple contiguous subsequences, so that all sequences used for training and validation have length in the range $[l_{min}, l_{max}]$. In addition to the features present in each transaction, there is also meta-data information associated with each sequence. This meta-data contains the starting account balance, start date of the sequence, and for the *czech* dataset, the customers' age at the start of the sequence. To preprocess the data for the generative models, continuous features are linearly scaled to have a variance of 1, and categorical features are encoded with a one-hot encoding. In the generic preprocessing step used by all models, we follow the method of [17] and represent time information by providing the start date as meta-data and including a time delta feature with each transaction that indicates the amount of time that has passed between transactions. When using BF, we perform a further preprocessing step (detailed in Sect. 5.1) to create additional date-based features which BF requires.

3 Methods

3.1 Generative Adversarial Networks (GANs)

GANs [8] are a commonly used generative model, and are capable of generating high-quality synthetic data in many domains [3,6,10]. TG [31] and DG [17] are two GAN models that have been successful at generating complex multivariate sequence data. Each of these models has unique innovations that allow them to generate high-fidelity synthetic sequences. In TG, an embedding scheme is used so that the generator and discriminator are operating in an embedded space, and a supervised loss based on predicting the next sequence element is used in addition to the standard GAN training objective. In DG, there are many innovations, including *batch generation* to better capture long-term dependencies, a conditional generation mechanism to deal with relationships between metadata and sequences, and a custom *auto-normalization* scheme that reduces mode collapse.

3.2 Transformers

The *transformer* architecture [27] was designed to perform sequence modeling tasks without a recurrence, instead relying on an attention mechanism and positional encoding scheme to model sequence ordering. While originally proposed as a language model [27], transformers have since been applied to modeling many types of sequences [14,30]. In this work, we use the *transformer-decoder* (TD) [18] variant of the transformer, as this is most appropriate for generating novel sequences. TD is designed as an auto-regressive model that can model probability distributions over sequences. The main innovations in the transformer and TD architectures are *positional encodings* (PEs) and *multi-head attention* (MHA). Since transformers do not use recurrence, and process all sequence elements simultaneously, the PEs are designed to allow the model to learn ordered sequences by adding a PE vector to the initial embedding. While there are many possible options for creating PE vectors, a standard choice for d-dimensional PE vectors is for the i^{th} dimension, corresponding to input position t, to be $\sin(t/10000^{i/d})$ if i is even, and $\cos(t/10000^{i/d})$ otherwise. The MHA mechanism allows the model to create multiple sequence representations by projecting the encoded sequences into multiple sub-spaces. Scaled dot-product attention [27] is then applied separately in each sub-space. When TD models are applied to sequences of discrete symbols, including language, they are trained using the maximum-likelihood objective of minimizing the negative log-likelihood of observed sequences, $-log(P(seq;\Theta))$, with parameters Θ. The probability of a length n sequence, $s = (x_1, ..., x_n)$, is computed using the auto-regressive factorization $p(seq;\Theta) = \prod_{i=1}^{n} p(x_n|x_1, ..., x_{n_1};\Theta)$, which is implemented by the TD.

4 Related Work

There are several different approaches to creating and evaluating synthetic financial time series. Here, we give a brief overview of the most relevant works.

4.1 Synthetic Financial Time Series

Traditionally, agent-based models were used to generate synthetic sequences of financial banking data [2,19], similar to the type of data modeled in our work, as well as for generating synthetic stock-market data [4,12,21].

Methods based on DGMs have recently begun to outperform agent-based approaches in generating realistic, univariate financial sequences [11,24,25,29]; however, there is less research on generating multivariate financial data. A GAN model for generating multivariate sequences of stock option prices was proposed in [28]. The work most similar to ours is StockGAN [13], which generates synthetic stock-market order-stream data, where each sequence item contains information about the order price, quantity, type, and date.

Methods based on DGMs have recently begun to outperform agent-based approaches in generating realistic, univariate financial sequences [11,24,25,29]. There is less research on generating multivariate financial sequences; however, [28] introduced a GAN model for generating multivariate sequences of stock option prices, and [13] proposed StockGAN, which generates synthetic stock-market order-stream data.

A critical difference between the banking data we are interested in and the datasets used in these works on financial time series is the transactional nature of our data. The previously mentioned works all aim to model sequences where measurements are taken at regular intervals, such as daily stock prices. In our transactional data, the time between transactions varies, and the timing information plays a critical role influencing the transactional properties. Existing work on modeling transactional data with DGMs is limited, and we are not aware of other works which have solely focused on this task. In the papers which introduced both TG [31] and DG [17], the authors briefly discuss how their models can be used on data with irregular time intervals. In both cases, the authors suggest adding a time delta feature to indicate the time between elements and modeling this like a typical continuous feature. However, neither of these works attempts to show that their models can learn patterns based on dates or times.

To the best of our knowledge, transformers have not yet been applied to the task of generating synthetic financial time-series data. Originally proposed as a language model, transformer models such as GPT-3 can generate novel text with narrative structure [3]. Transformers have also been applied to modeling other types of time series data, including influenza prevalence [30], as well as electricity usage and traffic [7,14,16].

4.2 Evaluation of Synthetic Sequence Data

The evaluation of synthetic data depends upon its planned use. If synthetic data is planned to augment training data, then one approach is to train the model on synthetic data and evaluate its predictive performance on real data [17,31]. If it can achieve comparable accuracy on real data to a model trained on real data, then this is taken as evidence of the quality of the synthetic data. This approach is less valuable when the use of the synthetic data is not known a priori.

Continuous Data. A simple way to evaluate synthetic financial time-series data is to compare univariate distributions, using metrics such as the 1-Wasserstein distance [29] or Kolmogorov-Smirnov distance [2,13]. For multivariate data, these distances can be computed separately for each feature of interest [13]. A limitation is that these metrics do not consider interactions between features, nor sequence order. Due to the limited work in generating multivariate banking data, there are no domain-specific metrics we are aware of. In works on financial sequences of asset prices, such as [13,29], domain-specific metrics were used that focused on well-documented features that occur in real market data known as *stylized-facts* [5].

Categorical Data. [13] studied synthetic financial time-series that generates data with both categorical and continuous-valued features, however, their evaluation only focused on continuous features. [32] use a randomly initialized LSTM model to generate a dataset of discrete sequences that were used to train their sequence generator. The LSTM model was then used to evaluate the likelihood of the data produced by the generator. [15] adopt a similar approach, performing additional validation experiments on real text sequences. To evaluate the quality of the generated text, they use BLEU scores [22], which measure the proportion of N-grams in the generated data that also occur in the real data.

5 Banksformer

We have created a modified TD model, called Banksformer (BF), to generate multivariate sequences of banking transactions. There are two main innovations in the design of BF. First, a preprocessing step allows BF to model sequences of items that contain multiple features of different types, including continuous and categorical features, as well as dates. Second, BF uses a novel method for generating multivariate time series data, in which each field of a transaction is generated sequentially. Our results indicate that this allows BF to better learn the joint distribution, such as $p(amount, tcode)$ as the product of two simpler distributions $p(amount, tcode) = p(tcode)p(amount|tcode)$.

5.1 Date Mechanism

The unique way Banksformer handles dates involves two parts – encoding and prediction. In BF, we create multiple features based on the timestamp to facilitate learning date-based patterns. Specifically, the day of the month (DoM), the number of days until the months' end (DTME), the day of the week (DoW), and the month of the transaction are each represented using two features. The two features are $f_1 = \sin(2\pi i/n_i)$ and $f_2 = \cos(2\pi i/n_i)$, where i is an ordering index and n_i is the number of possible indices (e.g., $i = 0$ and $n_i = 12$ when encoding the month of January). Additionally, BF also models a time delta (Δ_t) feature, as is done in TG and DG.

The way we have chosen to encode the date information helps BF learn date patterns; however, it also clearly contains redundancy. When generating data with BF, we first generate a probability distribution over the result for each date feature, and then create a distribution over the transaction date as

$$p(date) = \frac{1}{Z} \prod_{field \in \{DoM, DTME, DoW, month, \Delta_t\}} p_{field}(date[field]), \tag{1}$$

where Z is a normalizing constant.

We implement this with the following approach. First, a maximum time between transactions is set to make the approach feasible. The distribution over the time delta feature is modeled with a truncated Gaussian distribution, covering the range from 0 to the maximum time. BF outputs two features for the time delta, which are interpreted as the mean and variance to the truncated Gaussian. For each of the other features, BF outputs a categorical distribution over the options, which is created by a softmax layer. To compute the normalizing constant for the distribution, we sum the normalized probabilities of all dates between 0 and the maximum number of days from the current date. We then sample a date from this distribution, and then convert the selected date back into the separate date features.

5.2 Architecture

Figure 1 outlines the architecture of BF, which is composed of 3 main parts. The input layer takes a sequence of multivariate transactions and maps it to a d_{model}

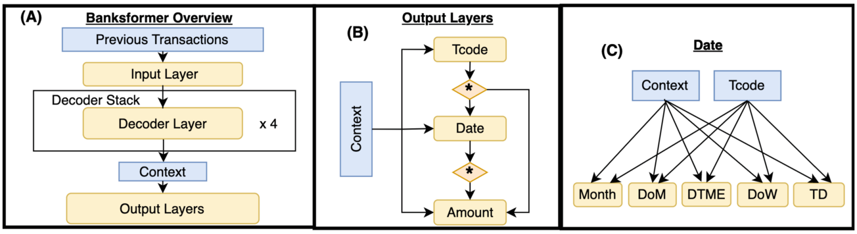

Fig. 1. An illustration of BF. (A) An overview of BFs architecture. (B) A zoomed-in view of the output layers, showing how BF sequentially handles transaction parts. When generating data, the * boxes indicate a sampling operation that samples a value from the input distribution. During training, teacher forcing is used, and the * boxes indicate the true value which should have been produced by the input distribution. (C) A further zoomed-in view of the date layer, showing that each piece of date information is predicted independently from the context, which encodes the sequence of previous transactions and the true value of the tcode for the current transaction.

dimensional sequence to which the positional encoding is added. The decoder stack then processes the encoded sequence and emits a context sequence that encodes predictions about the next element in the sequence. Finally, the output layers process each context and transform them into transaction predictions.

Input Layer. The input layer in BF is fully connected and simply maps the input data with dimension d_{input} to a representation with dimension d_{model}, which is used throughout the decoder stack.

Decoder Stack. After the input layer, BF contains a stack of 4 identical decoder layers, following a similar design as the decoder layers used in [27]. Each decoder layer is composed of two sub-layers. The first is a masked multi-head self-attention layer. This layer allows the network to attend to all sequence positions less than i when predicting the i^{th} element. This design follows the decoder stack in [18]. The final decoder layer emits a vector with size d_{model}. Our BF synthetic datasets were created using $d_{model} = 128$ (see Supplementary materials for a complete list of parameters).

Output Layer. In BF, the output contains multiple important pieces of information. This work focuses specifically on three: a categorical tcode, a transaction date, and a real-valued amount. The output layer of BF contains a conditional generation mechanism, which generates each of these values sequentially, and conditions each value on all previous ones. In the end, our model represents the probability distribution of the k^{th} transaction ($trans_k$) in a sequence as $p(trans_k|hist) = p(tcode_k|hist) \cdot p(date|hist, tcode_k) \cdot p(amount|hist, tcode_k, date)$, where $hist$ is the transaction history up to the k^{th} element of the sequence.

Loss Function. The loss function used for training BF treats each piece of information within a transaction separately, with the overall loss a weighted sum of individual losses. For continuous features, BF outputs predictions as parameters to a normal distribution, and the loss is the negative log probability of the data under the distribution. For categorical features, categorical cross-entropy is used.

5.3 Generating Data

BF generates synthetic data in the following way. The first element of the sequence contains the metadata, transformed into a vector with the same dimensionality as the feature dimension of the training sequences. A sequence of l transactions is then iteratively generated. At each step, the current generated sequence is passed as input, and the next element in the sequence is output. This element is then concatenated to the existing generated sequence. When generating a transaction, BF generates each attribute in a predefined order, conditioning each attribute on all previous attributes. Each generated attribute is output by a unique, fully connected layer. For categorical attributes, the raw output from the associated layer is passed through a softmax function to create a probability distribution over possible values, and the generated value is randomly sampled from this distribution. For continuous attributes, BF outputs two features, which are treated as the mean and variance of a normal distribution, and the generated value is sampled from this distribution. The transaction date is sampled from the distribution in Eq. 1, following the method detailed in Sect. 5.1.

6 Results

In this section, we present a comparison of synthetic data generated by BF, TG, and DG on both the *czech* and *uk* datasets. Due to space limitations, the figures in this section focus on results from the *czech* dataset; however, we have included additional figures further detailing our results on the *uk* dataset as a supplementary PDF. For BF and TG, all synthetic sequences had an equal number of transactions (20 for *uk*, 80 for *czech*), and the start dates (plus ages for the *czech* data) were randomly sampled from the empirical distribution in the real datasets. In contrast, DG generates sequence lengths and meta-data along with the transaction sequences. To better understand the quality of our generated data, we use a set of metrics to evaluate multiple aspects of our synthetic data.

6.1 Univariate Distributions

The most straightforward metrics are based on comparing univariate feature distributions for the continuous and categorical features. We use the Wasserstein-1 distance for distributions of continuous variables and the Jensen Shannon divergence (JSD) for discrete variables to quantify the difference in univariate distributions. For continuous variables, we compare the distributions of transaction amounts and monthly cash flow. The monthly cash flow of an account is simply the sum of all credits and debits (positively and negatively-valued transaction amounts) in a given month. We are interested in cash flow distributions because, unlike the transaction amount, cash flow is not directly modeled as a variable in the training data. However, cash flow is still an important facet of bank data. If synthetic data can capture the cash flow patterns from the real training data, this will support the claim that the model is learning the actual data distribution. As we can see from Fig. 2, the synthetic data generated by BF best captured the monthly cash flow patterns from the real data. This is supported quantitatively as well (Table 2). The amount distribution produced by BF was quantitatively worse than both TG and DG on the *czech* data (Table 2). However, when viewed on a log scale, BF appears to better capture the three modes of the real amount distribution Fig. 2. On the *uk* dataset, BFs' amount distribution was closest to the real data. The data generated by BF also performs best at emulating the tcode distribution in the *czech* data, which can be seen in Fig. 2 and Table 2. DG does nearly as well as BF at capturing the tcode distribution on the *czech* data, and slightly better than BF on the *uk* data.

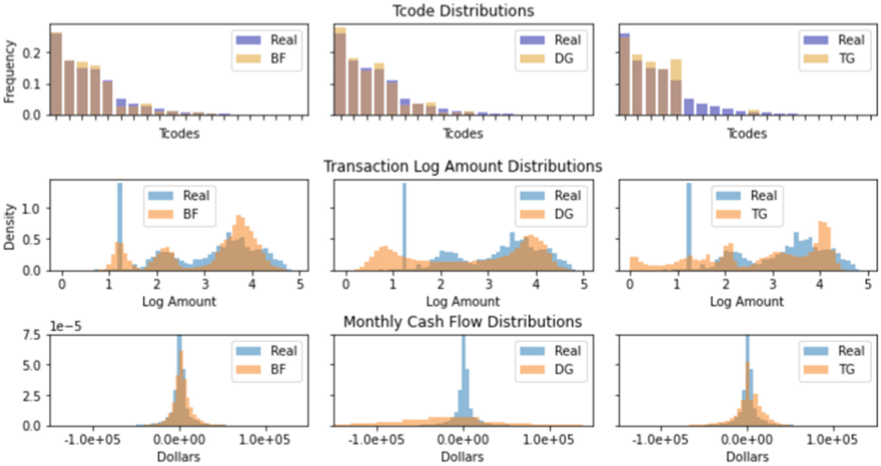

Fig. 2. Comparison of univariate distributions in *czech* data. This figure shows a comparison of the distributions for the tcode (top), log amount (middle), and monthly cash flow (bottom) in the synthetic datasets produced by BF, DG, and TG.

6.2 N-grams

We also compare N-gram distributions for the categorical feature to measure the models' ability to capture sequence orderings. Here we focus on 3-grams, and use the JSD to quantify differences in these distributions. We experimented with other values of N and the results did not change significantly. However, the JSD becomes harder to estimate as N increases because the empirical N-gram distributions become worse estimates of the true N-gram distributions

Table 2. Results Summary. The first 2 score columns are the Wasserstein-1 distances comparing the univariate amount (Amt) and monthly cash flow (CF) distributions respectively. The next two columns are JSD results comparing the univariate distributions of the tcode (Tcode) and transaction day of the month (DoM). The final columns are also JSD results. The Tcode 3G column show the JSD between the distributions of tcode 3-grams. And finally, the (Tcode, Date*) column compares the joint distributions of tcode and the most significant categorical date feature, which is DoM for the *czech* data, and DoW for the *uk* data. The bottom three results for the *czech* dataset show the results of ablation experiments; ablation results for the *uk* dataset can be found in the supplementary PDF.

Data	Model	Amt	CF	Tcode	DoM	Tcode 3G	Tcode, Date*
Czech	BF	2102	**2738**	**0.004**	0.011	**0.042**	**0.251**
	DG	1939	57800	0.007	0.090	0.132	0.660
	TG	**1931**	4980	0.075	0.059	0.337	0.638
	BF-ND	3705	4191	0.009	0.059	0.059	0.595
	BF-NC	3580	4775	0.158	**0.006**	0.411	0.542
	TF-V	4726	4138	0.185	0.059	0.445	0.674
UK	BF	**42.6**	**541.8**	0.015	**0.024**	0.156	**0.008**
	DG	179.0	1051	**0.011**	0.034	**0.135**	0.061
	TG	116.0	1460	0.237	0.087	0.622	0.077

due to the curse of dimensionality. We attempted to mitigate this with *additive smoothing* [20], however this did not significantly change the results, so the results we present are based solely on comparing empirical distributions. Figure 3 compares the distributions of the most common N-grams, and shows both BF and DG produce more accurate N-gram distributions on the *czech* data than TG. This is supported by quantitative results in Table 2, which also show that BF outperforms DG in terms of the JSD metric on the *czech* dataset. This metric also shows DG performs slightly better than BF on the *uk* dataset.

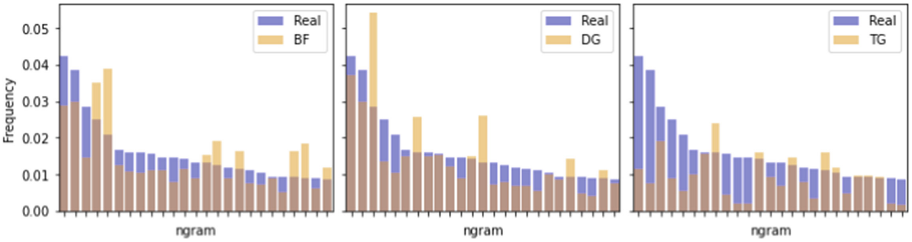

Fig. 3. 3-gram frequency comparison. This figure compares the frequency of the 25 most commonly occurring 3-grams in the real *czech* data, for each of the synthetic datasets.

6.3 Joint Distributions

One limitation of the previous metrics is that they do not account for how well feature interactions are modeled in the synthetic data. To get a sense of the overall joint distribution, we can visually compare the distributions of two-dimensional projections of the datasets (Fig. 4). To create this visualization, we follow the approach of [31], The sequences were first flattened along the temporal dimension and then a PCA model was fit to the real data. All data sets are projected into 2D using this PCA fit. Figure 4 shows that there are multiple peaks in the real *czech* data, and that BF reproduces these peaks on the whole. DG only poorly reproduces the real data, yielding a bimodal distribution, and TG focuses on a single mode.

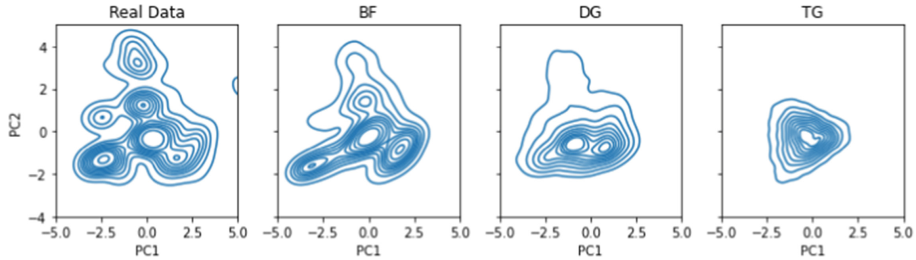

Fig. 4. PCA visualization of *czech* data. The two principal components of the data distributions obained using PCA. The generated data are projected using the PCA model that was fit to the real data.

Figure 5 shows the distribution over the day of the month for two specific tcodes that only occur at specific times of the month. The top row is for interest credited to the account, which only happens on the last day of the month, and the bottom row is a type of debit transaction that only occurred between the 5^{th} and 14^{th} of the month. BF is the only model able to learn the date pattern associated with these tcodes. In particular, our model can correctly generate transactions at the end of the month, even though the last day of the month may occur on days 28 to 31. The JSD may be used to quantify how well the relationship between tcodes and categorical date features were learned in general. Table 2 shows the JSD for the *czech* data, using the joint (tcode, DoM) distributions, and the *uk* data, using joint (tcode, DoW) distributions. For both data sets, BF significantly outperforms both DG and TG.

Different transaction types also have different associated amount distributions. In Fig. 6, we compare the conditional amount distributions for the two most common tcodes in the *czech* dataset. In this figure, we can see that BF, TG and DG all appear to have approximately learned the relationship between amount and tcode. Additionally, this figure also shows qualitative differences in the conditional amount distributions produced by the different models. BF tends to produce narrower, symmetric distributions, which are centered near the

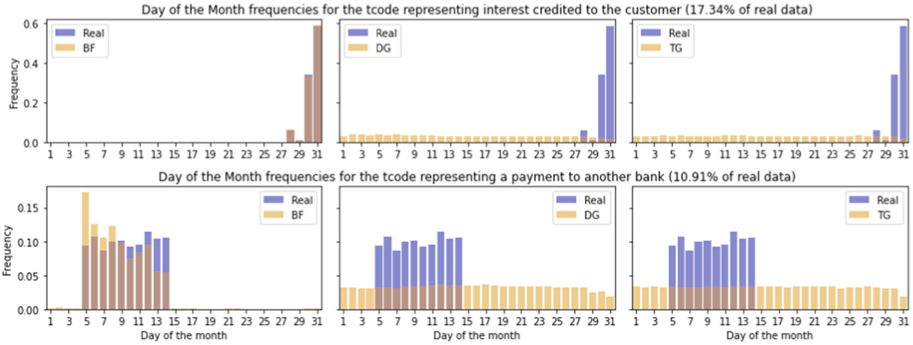

Fig. 5. Date, tcode relationship in *czech* data. This figure shows the conditional distribution of the transaction day of the month, given the tcode, for two tcodes that are strongly related to the date. This figure shows that BF (left) is the only model which has learned the relationship between these tcodes and the date.

mean of the real data; whereas both DG and TG tend to produce much wider, asymmetric distributions.

6.4 Ablation

To illustrate the impact of the innovations behind BF, we perform ablation experiments on conditional generation and date generation mechanisms. Specifically, we create the following three ablated versions of BF:

- A version without the date mechanism (BF-ND). In this implementation, we model the date using only the time delta feature, as is done in TG and DG.
- A version without conditional generation (BF-NC). In this implementation, we generate all transaction fields simultaneously.
- A basic transformer model with neither mechanism (TF-V).

The results from these experiments are shown in Table 2, which validate that both mechanisms introduced to the architecture of BF led to improved performance on most metrics. This was particularly true for the metrics that measured joint distributions, as well as metrics related to the amount, where BF scored much better than the ablated versions, and TF-V scored noticeably worse. For other metrics, different ablations had different impacts. The BF-NC version did worse than BF-ND on comparisons of both the tcode and tcode 3-gram distributions, with BF-NC being comparable to TF-V on these metrics. Similarly, BF-ND does worse than BF-NC and is comparable to TF-V on the DoM metric, which compares the distributions of transaction day of month.

Overall, these results are in line with expectations. It is somewhat surprising that the conditional generation mechanism improved the distributions of tcodes and tcode 3-grams, as the tcode is the first feature produced when generating conditionally. It may be that without conditional generation, the other features

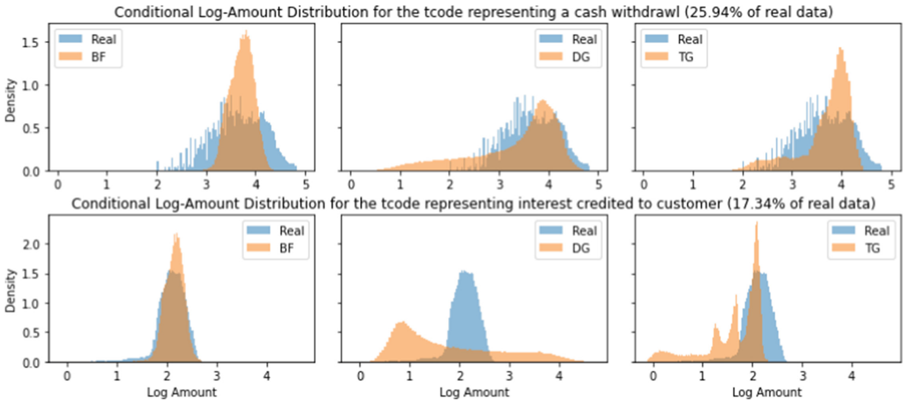

Fig. 6. Amount, tcode relationship in *czech* data. This figure shows a comparison of the conditional distributions $p(amount|tcode)$ produced by BF, DG and TG, against real data for the two most common tcodes in the real data.

become more difficult to model, causing the model to spend more effort learning those relationships, and less on the tcodes. We plan to investigate this further in future work.

7 Discussion

Our experiments show that the design of BF led to a clear improvement over TG and DG in modeling financial transactional sequences. Qualitatively, the most significant area of gain is in modeling the joint relationship between dates and transaction types, as only BF was able to learn these. Quantitatively, BF also created data that better matched the statistical properties of real data, according to the majority of the metrics we considered. Through ablation experiments, we demonstrated that both of BFs' innovations, the date mechanism and conditional generation for the individual transaction fields, improved synthetic data quality. We believe a promising future direction for this work is to explore hybrid models and combine innovations from BF, TG, and DG. There are multiple approaches we have in mind to explore this idea, including adapting the date mechanism from BF to GAN models based on TG and DG, and adding an adversarial training step to BF.

Another critical area for future work is to examine the privacy implications of these models. One major motivation for studying synthetic banking transaction data is to minimize reliance on real private data. However, before these models can be used to generate synthetic data to replace real data with genuine privacy concerns, users must be aware of any potential information which could be leaked through synthetic datasets.

Reproducible Research Statement. Code for Banksformer is available at github.com/BigTuna08/Banksformer_ecml_2022, as well as copies of the datasets and links to the other models used in this work. The synthetic datasets we have generated are available upon request.

References

1. Alaa, A.M., van Breugel, B., Saveliev, E., van der Schaar, M.: How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. CoRR abs/2102.08921 (2021)
2. Assefa, S., Dervovic, D., Mahfouz, M., Balch, T., Reddy, P., Veloso, M.: Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls. InfoSciRN: Data Protection (Topic) (2020)
3. Brown, T., Mann, B., Ryder, N., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
4. Byrd, D., Hybinette, M., Balch, T.H.: ABIDES: towards high-fidelity market simulation for AI research. ArXiv abs/1904.12066 (2019)
5. Cont, R.: Empirical properties of asset returns: stylized facts and statistical issues. Quant. Finan. **1**, 223–236 (2001)
6. Engel, J., Agrawal, K.K., Chen, S., Gulrajani, I., Donahue, C., Roberts, A.: Gansynth: adversarial neural audio synthesis. arXiv:1902.08710 (2019)
7. Farsani, R.M., Pazouki, E.: A transformer self-attention model for time series forecasting. J. Electr. Comput. Eng. Innov. (JECEI) **9**(1), 1–10 (2021)
8. Goodfellow, I., et al.: Generative Adversarial Networks. ArXiv abs/1406.2661 (2014)
9. Jordon, J., Yoon, J., van der Schaar, M.: Measuring the quality of synthetic data for use in competitions. arXiv:1806.11345 (2018)
10. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. CoRR abs/1912.04958 (2019)
11. Koshiyama, A., Firoozye, N., Treleaven, P.: Generative adversarial networks for financial trading strategies fine-tuning and combination. Quant. Finan. **21**(5), 797–813 (2021)
12. LeBaron, B.: Chapter 24 agent-based computational finance. In: Handbook of Computational Economics, vol. 2, pp. 1187–1233. Elsevier (2006)
13. Li, J., Wang, X., Lin, Y., Sinha, A., Wellman, M.: Generating realistic stock market order streams. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, pp. 727–734 (2020)
14. Li, S., Jin, X., Xuan, Y., et al.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
15. Li, Z., Xia, T., Lou, X., et al.: Adversarial discrete sequence generation without explicit neural networks as discriminators. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 3089–3098. PMLR (2019)
16. Lim, B., Arık, S., Loeff, N., Pfister, T.: Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. Int. J. Forecast. **37**(4), 1748–1764 (2021)

17. Lin, Z., Jain, A., Wang, C., Fanti, G., Sekar, V.: Using GANs for sharing networked time series data: challenges, initial promise, and open questions. In: Proceedings of the ACM Internet Measurement Conference, IMC 2020, pp. 464–483. Association for Computing Machinery, New York (2020)
18. Liu, P.J., Saleh, M., Pot, E., et al.: Generating Wikipedia by Summarizing Long Sequences. ArXiv abs/1801.10198 (2018)
19. Lopez-Rojas, E.: Applying Simulation to the Problem of Detecting Financial Fraud. Ph.D. thesis, Blekinge Institute of Technology (2016)
20. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
21. Panayi, E., Harman, M., Wetherilt, A.: Agent-based modelling of stock markets using existing order book data. In: MABS (2012)
22. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 311–318 (2002)
23. Berka, P., Sochorova, M.: PKKD 1999 Discovery Challenge (1999). Accessed 01 Apr 2022
24. Silva, B.D., Shi, S.S.: Towards Improved Generalization in Financial Markets with Synthetic Data Generation (2019)
25. Takahashi, S., Chen, Y., Tanaka-Ishii, K.: Modeling financial time-series with generative adversarial networks. Physica A: Stat. Mech. Appl. **527**, 121261 (2019)
26. Theis, L., Oord, A.V.D., Bethge, M.: A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844 (2015)
27. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, pp. 6000–6010. Curran Associates Inc., Red Hook (2017)
28. Wiese, M., Bai, L., Wood, B., Buehler, H.: Deep hedging: learning to simulate equity option markets. arXiv preprint arXiv:1911.01700 (2019)
29. Wiese, M., Knobloch, R., Korn, R., Kretschmer, P.: Quant GANs: deep generation of financial time series. Quant. Finan. **20**(9), 1419–1440 (2020)
30. Wu, N., Green, B., Ben, X., O'Banion, S.: Deep transformer models for time series forecasting: the influenza prevalence case. CoRR abs/2001.08317 (2020)
31. Yoon, J., Jarrett, D., van der Schaar, M.: Time-series generative adversarial networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
32. Yu, L., Zhang, W., Wang, J., Yu, Y.: SeqGAN: sequence generative adversarial nets with policy gradient. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI 2017, pp. 2852–2858. AAAI Press (2017)