Large-Scale Transformer models for Transactional Data

Fabrizio Garuti[1,2], Simone Luetto[3], Enver Sangineto[2] and Rita Cucchiara[2,4]

[1]Prometeia Associazione, Bologna, Italy
[2]AImageLab, UNIMORE, Modena, Italy
[3]Prometeia SpA, Bologna, Italy
[4]IIT-CNR, Italy

Abstract

Following the spread of digital channels for everyday activities and electronic payments, huge collections of online transactions are available from financial institutions. These transactions are usually organized as time series, i.e., a time-dependent sequence of tabular data, where each element of the series is a collection of heterogeneous fields (e.g., dates, amounts, categories, etc.). Transactions are usually evaluated by automated or semi-automated procedures to address financial tasks and gain insights into customers' behavior. In the last years, many Trees-based Machine Learning methods (e.g., RandomForest, XGBoost) have been proposed for financial tasks, but they do not fully exploit in an end-to-end pipeline all the information richness of individual transactions, neither they fully model the underling temporal patterns. Instead, Deep Learning approaches have proven to be very effective in modeling complex data by representing them in a semantic latent space. In this paper, inspired by the multi-modal Deep Learning approaches used in Computer Vision and NLP, we propose *UniTTab*, an end-to-end Deep Learning Transformer model for transactional time series which can uniformly represent heterogeneous time-dependent data in a single embedding. Given the availability of large sets of tabular transactions, UniTTab defines a pre-training self-supervised phase to learn useful representations which can be employed to solve financial tasks such as churn prediction and loan default prediction. A strength of UniTTab is its flexibility since it can be adopted to represent time series of arbitrary length and composed of different data types in the fields. The flexibility of our model in solving different types of tasks (e.g., detection, classification, regression) and the possibility of varying the length of the input time series, from a few to hundreds of transactions, makes UniTTab a general-purpose Transformer architecture for bank transactions.

Keywords

Deep Learning, Large Scale Model, Representation Learning, Time series prediction, Transactional data

1. Introduction

Transactional data are time-dependent collections of financial transactions. For instance, a bank account can be seen as a time series of transactions, each composed of a tabular data entry with fields specifying the transaction amount, the transaction operation and the receiver type (see Figure 1). These data can be used as training data for different Machine Learning approaches, in a variety of tasks. Some examples are:

- Customer Value Management, to support marketing or commercial actions, for instance via the creation of tailored offers;
- Credit and Liquidity Risk, to assess the initial risk, and to detect early risk signals, for instance, represented by changes in expense patterns or in the regularity of incomes;
- Fraud detection and Anti Money Laundering, to identify potential malicious behaviors.

For these purposes, transactional data have been so far processed with symbolic AI (e.g., rule-based expert systems) and/or with ensembles of **trees-based machine learning models**. RandomForest [1], LightGBM [2], XGBoost [3] and CatBoost [4] are the most frequently used.

Despite the success of Deep Learning methods in other application areas (e.g., Natural Language Processing and Computer Vision), trees-based models seemed to outperform deep learning models on most of the tabular datasets [5]. These datasets are typically composed of tens to hundreds of features and thousands to hundreds of thousands of samples. However, the size of transactional datasets is growing rapidly, now exceeding **millions of transactions** in some cases. Since the performance of deep-learning models improves with dataset size, tree-based models are the best choice only for small and medium-size datasets [6]. In addition, the use of tree-based models for transactional data is limited to constructing **simple aggregated features**, such as calculating the average spending over recent months or determining the total income for the past year. This approach has clear limitations in fully harnessing the precise and timely information that transactional data encapsulates.

Recently, various deep learning networks have been developed for heterogeneous data, mostly for tabular datasets. Lyu et al. [7] combine different modules and can represent both numerical and categorical features. Borisov et al. [8] use a distillation approach to map decision trees, trained on heterogeneous tabular data, onto homogeneous vectors. Huang et al. [9] represent cate-

Ital-IA 2024: 4th National Conference on Artificial Intelligence, organized by CINI, May 29-30, 2024, Naples, Italy
fabrizio.garuti@prometeia.com (F. Garuti);
simone.luetto@prometeia.com (S. Luetto)

Card	Timestamp	Amount	Use Chip	Merchant Name	Merchant City	Merchant State	Zip	MCC	Errors?
6	2013-09-05 07:19	$23.44	Swipe Transaction	Applebees	Petersburg	VA	23803	7832	
6	2013-09-05 16:08	$84.80	Online Transaction	Frontier Communications	ONLINE			4784	
6	2013-09-06 10:33	$15.66	Swipe Transaction	Green Wholesale	Newport News	VA	23605	5411	Technical Glitch
6	2013-09-06 14:45	$42.38	Online Transaction	Frontier Communications	ONLINE			4784	
6	2013-09-07 10:44	$15.82	Swipe Transaction	Barnes & Noble	West Covina	CA	23605	5541	
6	2013-09-08 14:03	$36.75	Online Transaction	Frontier Communications	ONLINE			4784	Bad CVV
6	2013-09-08 09:56	$80.05	Swipe Transaction	Chevron	Rosemead	CA	91772	4111	
6	2013-09-09 12:34	$223.46	Swipe Transaction	Kelly Auto Repair	Newport News	VA	23606	3393	
6	2013-09-10 08:46	$36.43	Online Transaction	Frontier Communications	ONLINE			4784	
6	2013-09-11 06:09	$48.07	Swipe Transaction	Anwar Grocery	Alhambra	CA	91801	5411	

Figure 1: Some samples of transactions from the Transaction dataset used for fraud detection.

gorical features using an attribute-specific embedding which is used as a prefix, concatenated with the actual field value. Schäfl et al. [10] use a non-parametric representation of the training data, which reminds the use of external networks in Transformers [11]. However, these approaches do not model the **temporal dynamics**: each row in the table is an individual sample. A trivial solution could be to concatenate multiple rows into single samples, but none of the previous work has demonstrated that their architecture can model more than hundreds of fields as an individual sample.

To overcome these problems, we propose a custom Deep Learning architecture based on modern Transformers [12], which can uniformly represent heterogeneous time-dependent data, and which is trained on a large-scale transactional dataset. We call our model UniTTab (Unified Transformer for Time-Dependent Heterogeneous Tabular Data), and we show that it consistently outperforms state-of-the-art approaches based on both Deep Learning and standard Machine Learning techniques.

2. Related works

The heterogeneous nature of transactional data and the lack of large public annotated datasets, due to privacy and commercial reasons, make these data extremely difficult to be handled by deep neural networks. However, in recent years some works have started to address these challenges. For instance, Padhi et al. [13] proposed one of the first deep learning architectures for heterogeneous time series (TabBERT). As a solution to data heterogeneity, the authors quantize continuous attributes so that each field is defined on its finite vocabulary.

Another recent work is TabAConvBERT proposed by Shankaranarayana & Runje [14]. They present an architecture that can deal with both categorical inputs (by using an embedding neural network) and numerical inputs (by using a shallow neural network).

The architecture presented by X. Huang et. al. [9] provides a solution to data heterogeneity, but it cannot handle the temporal component of the data and therefore is unable to solve tasks involving transaction sequences.

A different line of work involves directly or indirectly using natural language-based "interfaces" between the tabular data and a Transformer. For instance, in LUNA [15], numerical values are represented as an atomic natural language string.

Our proposal differs from the aforementioned works in different aspects. On the one hand, we deal with all the variability dimensions of the problem: numerical, categorical and temporal. On the other hand, we train our model using arbitrary length and long-range time series, which can include up to 150 transactions per sample. As a result, it is possible to deal with transactional tasks that require learning the long-term dependencies of the data. Furthermore, the learning phase is enriched with new custom masking techniques, which allow all related fields to be masked simultaneously, making the initial general-purpose training more challenging for our model.

3. Method

3.1. Pre-training and Fine-tuning

The current great availability of data together with the advancement of AI research have opened the possibility of developing larger models with more general purposes. This is already a reality in almost every application regarding unstructured data, like text, images and video. Many general-purpose models have gained fame in recent years: GPT-3 [16], BERT [17], CLIP [18], DALL-E [19]. All these models have been **pre-trained** using large datasets jointly with a self-supervised approach.

The goal of the pre-training phase is to learn a good representation of the input data. As a result, models trained within this scheme show great generalization capabilities even without further training, like the generation of text of GPT models. These capabilities can further improve after a second training phase, called **fine-tuning** over a specific task, for example, ChatGPT's stunning ability to conversate. Taking inspiration from these models, we use a large dataset of transactions to pre-train a Transformer network using self-supervised learning, and then we use a (smaller) labeled dataset to fine-tune the network for a specific task.

During the pre-training stage, we train our UniTTab model using the **Masked Token** pretext task [17]. Some

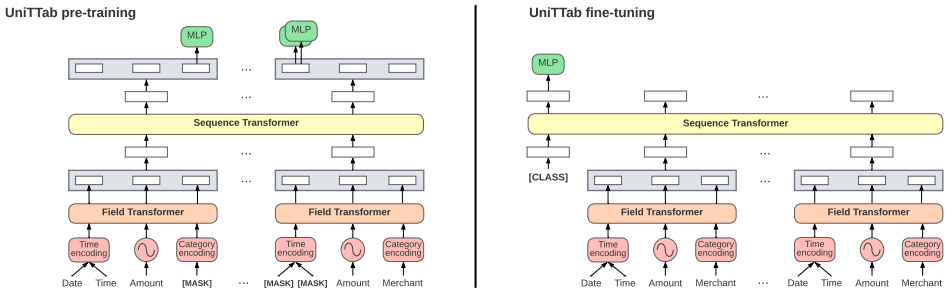

Figure 2: A schematic illustration of the UniTTab architecture for financial data.

of the input features are masked to the model, and the model is trained to predict the masked features as a function of the "visible" ones. This method makes it possible to train a model to automatically learn the semantics and to extract relevant information contained in the sequence of transactions. Specifically, for an input sequence, we randomly replace a field value with the special symbol [MASK]. We use a standard replacement probability value of 0.15 [17, 13]. Moreover, with probability 0.1, we also mask all the fields in a transaction, while, for the fields representing the time stamp, they are always either jointly masked or jointly unmasked. These additional masking strategies, inspired by the block masking of adjacent image patches used in BEiT [20], make the pretext task more challenging for the network.

It is important to remark that, within the pre-training phase, the training of the model solely relies on input features of transactions, eliminating the need for labels. This way we can use the entire transactional dataset, even if it is not fully labeled for the specific downstream task. This is the case of the Czech dataset [21] used for loan default prediction, described in Section 4.1.

3.2. The architecture

We develop a custom model called UniTTab, designed to be suited for sequences of heterogeneous transactional data. Borrowing the techniques used in text analysis in BERT or GPT models, we use input time series with variable length. We vary the sequence length from 10 to 150 transactions, where each transaction is composed of a fixed number of 10 or 6 fields. As a result, each time series can vary in length from 100 to 900 items, a challenging length to manage even for text sentences.

Given the data structure, we propose the hierarchical architecture shown in Figure 2. The architecture is composed of two different Transformers, and it is trained end-to-end. The first Transformer ("**Field Transformer**") takes as input the k features describing a single transaction, like transaction amount, merchant information, transaction date and time. The features are transformed in k final embeddings, which are concatenated in a single embedding vector. This embedding is the representation of a single transaction. Then a sequence of these embeddings, each representing a transaction, is fed to the second Transformer ("**Sequence Transformer**"). The Sequence Transformer models the statistic dependencies between different transactions in the sequence and outputs embedding elements in the latent space. This is the general-purpose latent space where the representation can be potentially exploited for many tasks, such as classification (e.g., to classify the client behavior), detection (e.g., to detect anomalies, frauds), and prediction (e.g., to predict product churn in next few months).

As depicted in Figure 2, during pre-training, for each masked field we use the corresponding output embedding to predict the field value. Instead, during fine-tuning, we add a class token [CLASS] at the beginning of the transaction embedding sequence, and we use the corresponding output embedding to solve the financial tasks. This embedding can attend to all transaction embeddings in the sequence, allowing it to exploit all field information of all transactions.

3.3. Feature representation

Our model can effectively represent heterogeneous data, encoding both numerical fields (e.g., the amount), categorical fields (e.g., the type of transaction), and fields with a specific structure (e.g., the date).

The most common approach to tackle this challenge is to reduce all the features to a common representation: usually numerical for ensemble of trees or categorical for deep learning architectures like TabBERT [13]. However, discretizing numerical features into a finite set of values results in a loss of information. For example, it could be important to know if an amount is precisely 20 euros or 20.50 to distinguish between a withdrawal and a grocery expense. For this reason we develop a custom representation to transform numerical values in the input vector. In particular, we represent each numerical value as a feature vector obtained by the concatenation of a battery of

different frequency functions (depicted with a sine wave symbol in Figure 2). Similar representations are used in NeRFs [22] for 3D synthesis. Conversely, we adopt a traditional "category encoding" to represent the categorical features, by using simple embedding neural networks (as used in [13]). Finally, we use a custom "time encoding" method for the timestamp attributes. The value of a timestamp is split using a combination of different field values: the year, the month, the day and, if necessary, the hour. Then each such value is represented as a categorical feature (e.g., with 12 elements for the month).

4. Experimental results

The effectiveness of the model has been tested over two **large size datasets** of transactions: the PKDD'99 Financial Dataset [21] and our Real Bank Account Transaction Dataset (in short, RBAT Dataset). The first dataset is public and is used as a benchmark for predicting loan default. Instead, the second dataset is private and is used to assess how well our model predicts customer churn in comparison to standard industry models. The chosen experiments are binary classification tasks with a large level of unbalance in the statistics of the two target classes.

In all the experiments, the model is first pre-trained using self-supervision (Section 3.1) and then fine-tuned on the classification task, in a standard supervised way, using the labeled data.

4.1. Loan default prediction

The loan default prediction is a classification task defined on the PKDD'99 Financial Dataset, which is a public dataset of real transactions from a Czech bank [21]. This dataset is composed of 1M of transactions from 4500 clients. It also includes customer information, but we use only the transactions, each composed of 6 fields (timestamp, amount, type and channel of the transaction).

The dataset presents a large fraction of unlabeled data, in fact most of the accounts don't have any loans, and they cannot be used for the classification task. This is the perfect example of the potentiality of our model: we perform the pre-training on all the accounts present in the dataset (4500) and then we fine-tune the model only on the labeled ones (478 for training and 204 in test). With such a small number of samples UniTTab has been able to obtain good results, way higher than ensemble of trees, and the possibility to exploit all the data in pre-training is its main advantage.

To evaluate our model's ability to deal with longer sequences and variable lengths, we test **different sequence lengths** of transactions. We define a maximum length value t_{max} (ranging from 50 to 150), and for each account we include the entire sequence of transactions if it is shorter than the maximum. Instead, if the sequence exceeds the maximum length, we only consider the most recent t_{max} transactions. It's important to note that, during pre-training the average sequence length is 232 transactions, whereas during fine-tuning the average sequence length is 80 transactions. This happens because, for fine-tuning, we only take transactions made before the loan begins. For this reason, if we set t_{max} to 150, during fine-tuning almost all transaction sequences are of variable length. It's also interesting to observe that, increasing the length of the sequence, the result of the model improves. This is likely due to the information increase in the input sequence, but it demonstrates that the model is able to deal with long sequences of transactions.

4.2. Effect of Pre-Training

One of the main advantages of using Deep Learning methods over traditional Machine Learning approaches is the possibility to pre-train a large network using a large unsupervised dataset, and then fine-tune the same network on the (usually scarcer) available annotated data of a downstream task. In order to quantify the contribution of the pre-training phase, and to show that this is useful also when the unlabeled dataset is not huge, we use the PKDD'99 Financial Dataset, and we pre-train the models with different portions of the pre-training dataset. Specifically, in Figure 3 we indicate the fraction of the pre-training dataset used for each experiment, where zero corresponds to training the models from scratch directly on the (labeled) downstream task data. The results in the figure show that both Deep Learning methods (i.e., TabBERT and UniTTab) significantly benefit from the pre-training phase, even using only a small portion of the unlabeled data (e.g., 0.25). Furthermore, when pre-training is performed, our UniTTab gets a significantly higher F1 score than traditional tree-based models.

4.3. Churn prediction: comparison with industry standards

We also compare our model with a custom transactional tree-based pipeline on a churn rate prediction task. The task is defined on the private RBAT dataset, which is provided by an international bank and is composed of several hundred million real transactions of bank customers.

The churn prediction task is defined as whether or not a customer churns in the next 3 months, given a 6-month history sequence of transactions performed by that customer. The history sequences provided to the models are of variable length, with an average of 192 transactions and up to a maximum of 500 transactions. Each sequence is associated with a binary target that represents the presence of the churn event for that customer in

Table 1
Loan default prediction task: average and standard deviation results obtained with 5 random seeds.

t_{max}	Model	F1 score	Average Precision	ROC AUC	Accuracy
50	TabBERT [13]	0.611(±0.032)	0.594(±0.031)	0.827(±0.048)	90.7(±1.6)
	LUNA [15]	0.604(±0.048)	0.613(±0.048)	0.869(±0.030)	92.5(±1.7)
	UniTTab (ours)	0.619(±0.011)	0.574(±0.017)	0.882(±0.021)	90.2(±1.5)
100	TabBERT [13]	0.636(±0.024)	0.625(±0.036)	0.874(±0.019)	91.6(±0.9)
	LUNA [15]	0.624(±0.075)	0.601(±0.018)	0.846(±0.025)	92.5(±1.7)
	UniTTab (ours)	0.654(±0.032)	0.653(±0.033)	0.903(±0.006)	91.4(±1.2)
150	TabBERT [13]	0.620(±0.024)	0.603(±0.016)	0.857(±0.026)	91.6(±1.1)
	LUNA [15]	0.637(±0.043)	0.589(±0.017)	0.851(±0.030)	92.6(±1.2)
	UniTTab (ours)	**0.673**(±0.038)	0.690(±0.030)	**0.912**(±0.018)	92.3(±1.1)
-	Random Forest [23]	0.2667	-	0.6957	89.27
	XGBoost	0.608(±0.079)	**0.700**(±0.040)	0.894(±0.019)	**92.8**(±1.8)
	CatBoost	0.527(±0.065)	0.617(±0.079)	0.866(±0.043)	92.0(±1.1)

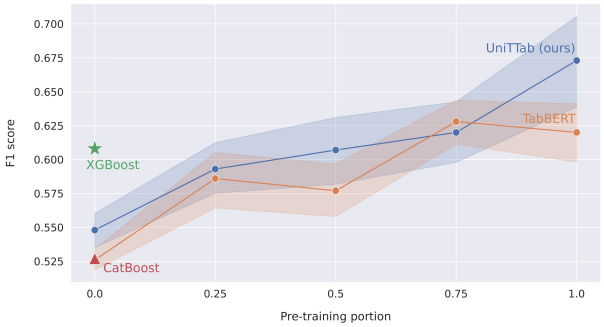

Figure 3: Loan default prediction task: impact of different portions of the pre-training dataset.

the following 3 months. Initially, our model has been pre-trained on a random sample of 1M untargeted accounts, corresponding to approximately 300 million transactions. Then, we evaluate the performance of our model and industry standards using fine-tuning datasets of different sizes, ranging from 50K transaction sequences up to 1 million sequences.

Figure 4 shows that our UniTTab model significantly outperforms industry standards for every training dataset size. It also demonstrates the scalability of our model through an increased number of fine-tuning samples: increasing the number of training accounts yields considerably improved AUC on the churn prediction task.

5. Conclusion

The UniTTab project presented in this paper is a step towards the creation of general-purpose architectures for bank transactions. The empirical results show that our model drastically outperforms both deep learning and standard machine learning based predictive models on different benchmarks. We believe that our work and our results can stimulate this research field and the adoption of self-supervised deep learning in banking data.

References

[1] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[2] G. M. Ke, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems (2017).

[3] T. . Chen, Xgboost: A scalable tree boosting system, Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (2016) 785–794.

[4] L. G. Prokhorenkova, Catboost: unbiased boosting with categorical features, Advances in neural

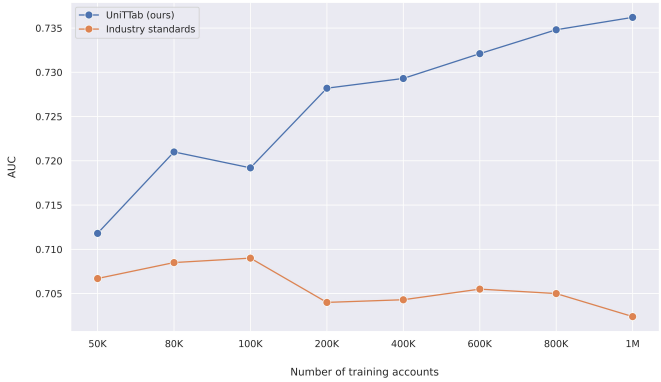

Figure 4: Customer churn rate prediction task: comparison with industry standard on different portions of the fine-tuning dataset.

information processing systems (2018).

[5] V. e. Borisov, Deep neural networks and tabular data: A survey, IEEE transactions on neural networks and learning systems (2021).

[6] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on tabular data?, 2022. arXiv:2207.08815.

[7] F. Lyu, X. Tang, H. Zhu, H. Guo, Y. Zhang, R. Tang, X. Liu, OptEmbed: learning optimal embedding table for click-through rate prediction, in: M. A. Hasan, L. Xiong (Eds.), Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022.

[8] V. Borisov, K. Broelemann, E. Kasneci, G. Kasneci, DeepTLF: robust deep neural networks for heterogeneous tabular data, Int. J. Data Sci. Anal. 16 (2023) 85–100.

[9] X. Huang, A. Khetan, M. Cvitkovic, Z. S. Karnin, TabTransformer: Tabular Data Modeling Using Contextual Embeddings, arXiv:2012.06678 (2020).

[10] X. Huang, A. Khetan, M. Cvitkovic, Z. S. Karnin, Tabtransformer: Tabular data modeling using contextual embeddings, arXiv:2012.06678 (2020).

[11] Y. Wu, M. N. Rabe, D. Hutchins, C. Szegedy, Memorizing transformers, in: ICLR, 2022.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is All you Need, in: NeurIPS, 2017.

[13] I. Padhi, Y. Schiff, I. Melnyk, M. Rigotti, Y. Mroueh, P. L. Dognin, J. Ross, R. Nair, E. Altman, Tabular Transformers for Modeling Multivariate Time Series, in: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2021.

[14] S. M. Shankaranarayana, D. Runje, Attention Augmented Convolutional Transformer for Tabular Time-series, in: 2021 International Conference on Data Mining, ICDM 2021 - Workshops, 2021.

[15] H. Han, J. Xu, M. Zhou, Y. Shao, S. Han, D. Zhang, LUNA: Language Understanding with Number Augmentations on Transformers via Number Plugins and Pre-training, arXiv:2212.02691 (2022).

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, arXiv:2005.14165 (2020).

[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL„ 2019.

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning Transferable Visual Models From Natural Language Supervision, 2021. arXiv:2103.00020.

[19] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, Zero-shot text-to-image generation, in: ICML, 2021.

[20] H. Bao, L. Dong, F. Wei, BEiT: BERT pre-training of image transformers, ICLR (2022).

[21] P. Berka, Workshop notes on Discovery Challenge PKDD'99, 1999. URL: https://sorry.vse.cz/~berka/challenge/pkdd1999/berka.htm.

[22] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, NeRF: representing scenes as neural radiance fields for view synthesis, in: ECCV, 2020.

[23] Z. Xu, Loan default prediction with Berka dataset, 2020. https://towardsdatascience.com/loan-default-prediction-an-end-to-end-ml-project-\with-real-bank-data-part-1-1405f7aecb9e.