E.T.-RNN: Applying Deep Learning to Credit Loan Applications

Dmitrii Babaev
dmitri.babaev@gmail.com
Sberbank AI Lab

Maxim Savchenko
savvvan@gmail.com
Sberbank AI Lab

Alexander Tuzhilin
New York University
atuzhili@stern.nyu.edu

Dmitrii Umerenkov
d.umerenkov@gmail.com
Sberbank AI Lab

ABSTRACT

In this paper we present a novel approach to credit scoring of retail customers in the banking industry based on deep learning methods. We used RNNs on fine grained transnational data to compute credit scores for the loan applicants. We demonstrate that our approach significantly outperforms the baselines based on the customer data of a large European bank. We also conducted a pilot study on loan applicants of the bank, and the study produced significant financial gains for the organization. In addition, our method has several other advantages described in the paper that are very significant for the bank.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing**;

KEYWORDS

credit scoring, recurrent neural networks, card transactions, multivariate time-series

ACM Reference Format:
Dmitrii Babaev, Maxim Savchenko, Alexander Tuzhilin, and Dmitrii Umerenkov. 2019. E.T.-RNN: Applying Deep Learning to Credit Loan Applications. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3292500.3330693

1 INTRODUCTION

Credit scoring is a very important problem for the banking industry because of the huge financial implications for the banks. Banking industry has developed credit scoring models ever since the middle of the XX century and has perfected these models ever since, investing millions of dollars in this process. Traditional credit scoring models rely on a loan application questionaire, applicants credit history and various other aggregate financial information relevant to the customer's application. These models use traditional machine learning methods, such as logistic regression, to compute the credit score of a customer that is indicative if the customer would return the loan or not. Although widespread and useful, these models have certain limitations. First, credit scoring require extensive feature engineering and deep domain knowledge in order to design good features. Second, if the customer does not have significant credit history, it is hard to make reliable scoring decisions regarding that person. Third, the currently existing models do not take full advantage of all the data available about the customer in modern settings.

In this paper we propose a novel approach, called *Embedding-Transactional Recurrent Neural Network (E.T.-RNN),* to compute credit scores of the bank customers by examining history of their credit and debit card transactions. We do it using a deep learning (DL) approach, as opposed to more traditional machine learning methods. Note that this approach is applicable only to those customers that have credit or debit cards with the bank. Since a significant percent of the applicants indeed have credit or debit cards, our method works for a large segment of the applicants. Furthermore our proposed method has the following advantages in comparison to the current credit scoring methods. First, as is shown in the paper, the proposed DL-based method outperforms the baselines, including models currently in use in the bank, resulting in significant financial gains. Second, the proposed DL-based model works directly on the customer transactions and does not need extensive feature engineering requiring deep domain expertise (generating hundreds or thousands of hand-crafted aggregate features). Third, our model works exclusively on the transactional data and, therefore, does not require any additional input from the client. This means that we can make credit loan decisions very fast, ideally in near real-time, because the whole credit scoring process is fully automated. Fourth, information in the transactional data is exceptionally hard to forge. Hence there is no need to check the correctness of the data, unlike the questionnaire and some other data sources used for scoring. Fifth, even the customers without any credit history can be assessed for credit worthiness, their transactional history constituting a source for estimating credit risks. Finally, the proposed method constitutes a *fair* approach to credit scoring, as it does not use information about the individual and therefore cannot be used to discriminate the credit applicants by various demographic factors. These constitute very significant advantages vis-a-vis current loan practices and have a potential to disrupt the retail banking loan industry.

One issue with the proposed approach constitutes the interpretability of the black-box models, such as neural networks. Different organizations across the world have different philosophies regarding applying black-box models to credit scoring problems. While in some countries lack of interpretability is considered as

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330693

clear "no-go", in other parts of the world it is considered more appropriate to use such models for the credit scoring tasks. Also, there has been significant progress in solving the black box interpretation problems in the last few years [9], [15], [23] and we expect this progress to accelerate even faster moving forward. Therefore, we believe that the issue of using the black-box models for credit scoring will be less relevant in the future.

This paper makes the following contributions. First, we propose to use neural networks on the customers fine-grained transactional data for credit scoring applications in the banking industry. Second, we tested our method against the benchmarks on the historical data and achieved superior performance. Third, we conducted a pilot study on loan applicants and produced significant financial gains for the bank.

The rest of the paper is organised as follows. In Section 2 we discuss the related work. In Section 3 we describe the proposed method. In Section 4 we present our experiments and in Section 5 the results of these experiments. Section 6 and 7 are dedicated to the discussion of our results and conclusions.

2 RELATED WORK

There is a large amount of research on credit scoring problems for the banking industry going back to first half of the XX century [10]. A wide range of methods has been used for this task, including logistic regression [32], decision trees [22], boosting [2], support vector machines (SVM) [16] and neural networks (NN) [30]. Credit scoring methods historically relied on using questionnaire data and applicant's credit history. However new data sources have been utilized more recently to increase scoring quality by using telecom data [4] and transactional data [19], [3], [20], [7], [29].

Most of the previous approaches to credit scoring used *aggregated* transactional data either globally [7] or over some time window, such as a month [19], [3] or a day [20], and most of them relied on the classical ML methods. For example, in [19] authors used generalized classification and regression trees on monthly transactional statistics. In [3] authors used discrete survival models on monthly transactional statistics. Furthermore, some authors used NN-based approaches to credit scoring on the aggregated transactional data. For example in [20] authors applied shallow convolutional neural networks on daily transactional statistics.

Furthermore, [29] has developed some credit scoring models on the *unaggregated* transactional data. However, they used classical ML methods, such as SVMs and weighted-vote relational neighbour classifiers, in their models. Moreover, they focused on the connectivity problem in their work to estimate credit risk and used only information of who transacted with whom, without deploying the full power of the transactional data.

Also, NNs have been applied to the analysis of the transactional data, but in other types of applications. In [31] authors used Long Short Term Memory (LSTM) Recurrent Neural Network (RNN) [13] on individual transaction features for detection of fraudulent transactions. For a review of NNs methods in credit card fraud detection see [1]. In [33] authors applied LSTM RNN for predicting credit scores for a peer-to-peer lending platform.

The main contribution of this work is that we use Neural Network methods for traditional *banking credit scoring* problems on the *unaggregated* transactional data. In this paper, we use an RNN based method in the credit scoring problem. We describe our approach to this problem in the next section.

3 THE METHOD

3.1 Transactional data

Our method computes credit scores using transactional data, each client having multiple credit card transactions, and each transaction having several attributes, both categorical and numerical, and occurring at a certain time. Our data can be described as multivariate time-series data, the schema of which is presented in Table 1. Merchant type field represents the kind of a merchant, such as airline, hotel, restaurant, etc. (note that it is impossible to restore the real merchant organization identifier from this field).

Table 1: Data structure for a single client

Amount	230	5	40
Currency	EUR	USD	USD
Country	France	US	US
Time	16:40	20:15	09:30
Date	Jun 21	Jun 21	Jun 22
Merchant Type	Restaurant	Transport-ation	Household Appliance
Card type	Visa Classic	Visa Classic	Visa Gold
Issuing Branch	90/10735	90/01735	90/01779
N opened debit cards	1	1	1
N opened credit cards	1	1	1

In the next section we describe an architecture of the neural network that computes credit scores using that transactional data.

3.2 Architecture overview

RNNs are used for processing sequential information. In a way, RNNs have "memory" over previous computations and use information from the previous time-steps in addition to the current input in order to produce next output. This approach is naturally suited for many NLP tasks including text classification, machine translation and language modelling [24].

Our *Embedding-Transctional RNN (E.T.-RNN)* architecture is presented in Figure 1 and is inspired by the NLP methods in the context of deep learning [24]. We treated the credit scoring task as a text classification task, using clients as texts and transactions as individual words.

As Figure 1 shows, the E.T.-RNN model consists of three parts: embedding layers, recurrent encoder and classifier. We will explain each of them in the rest of this section. Note that all the parts are trained *simultaneously* in the end-to-end manner.

3.2.1 Embeddings. Credit card transactions are mapped into a latent space before being passed to the encoder RNN. In particular,

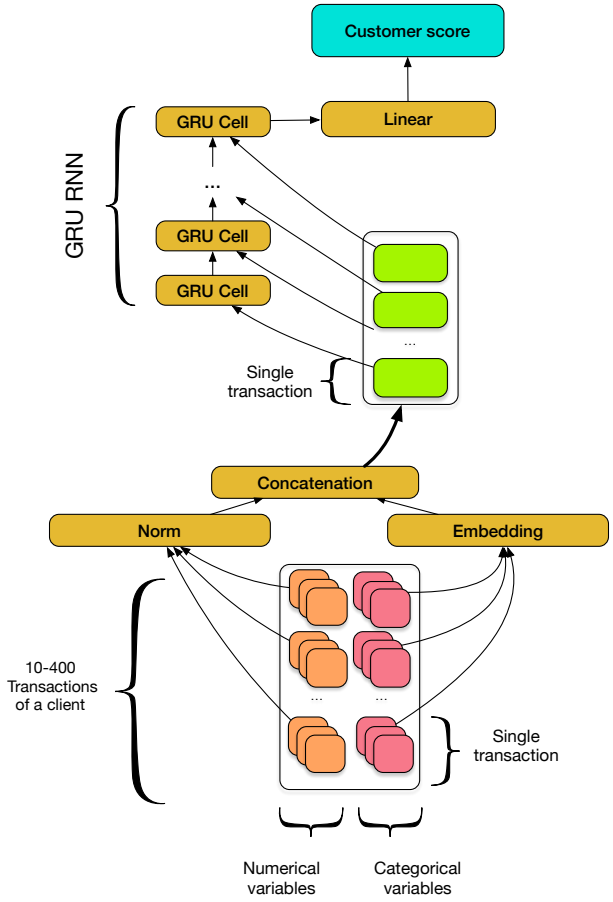
each categorical variable in each transaction is encoded to a low-dimensional vector via a corresponding embedding layer. The embedding layers are randomly initialized and trained simultaneously with the encoder. We have treated the timestamp as a collection of categorical variables each representing a date part (hour, weekday, month). Each transaction is represented as a concatenation of scalar variables and embeddings of categorical variables.

3.2.2 Encoder. We used a single layer RNN based on Gated Recurrent Unit (GRU) [8] as an encoder. The hidden vector from the last time step was used as the representation of the client. Note that this approach is also commonly used for text analysis [28].

3.2.3 Classifier. The hidden vector from the last time step is finally passed to the fully connected classifier sub-network. It turned out that a simple linear classifier outperformed several alternative approaches in our experiments and therefore we used it in our architecture.

More generally, we experimented extensively with different types of deep learning architectures, as explained in Section 4.3.1, and the architecture presented in Figure 1 turned out to be the best for our experiments.

Figure 1: Final architecture



3.3 Loss function

In this work we use the standard area under the ROC curve (ROC AUC) performance measure.

Several loss functions can be used as a proxy for the task of maximising ROC AUC, including the classic binary cross-entry loss: $L_{CE}(p, y) = -\sum_i y_i log(p_i)$ and margin ranking loss: $L_R(p_1, p_2, y) = \max(0, -y * (p_1 - p_2) + margin$ which directly optimizes ROC AUC.

In the final version of our model we decided to use margin ranking loss with margin 0.01, which showed the best results on our data, as presented in Section 4.3.1.

3.4 Ensembling

Ensembling [5] is a way to increase both quality of the model and its stability at the expense of time and computational power. In our case, we have a relative abundance of the negative class samples, as described in Section 4.1. Hence it is possible to use different subsamples of the negative class samples for training each model in the ensemble. The specific parameters will be described in Section 4.3.4.

In the final version of our model, we settled to use mean predictions of an ensemble of six separately trained models, as a practical balance between prediction quality and execution time. Ensemble quality gain and other possible ensembling strategies are further explored in Section 4.3.1.

4 EXPERIMENTS

4.1 Data

The data used for experiments was provided by a large European bank. For our experiments, we took transactional data for the clients who applied for the *retail credits*. Since strict adherence to the requirements of personal data protection laws was one of the key priorities for the Bank when carrying out the project, we cannot describe our data and share it with the readers. Furthermore, we only considered applicants who already used a debit or a credit card product in the bank. If a client has several cards, then transactions from every card was taken into account.

The available transactional data falls into subcategories: transaction-level (such as timestamp, country, amount, merchant type) and card-level (such as issuing branch, card type). Card-level data is duplicated verbatim for each transaction related to the corresponding card. An example of three typical card transactions is presented in Table 1. We also used two derived features calculated from transactional data:

- difference in days between the time of current transaction and the time of previous transaction by this customer
- time in days elapsed from the card issue date until the transaction date

Only the transactions performed before the application date are taken for training and validation.

Our training dataset represented more than 740 thousand clients with approximately 200 million transactions in total. As a target variable, we used the event of default for consumer loan during a year after its disbursement. The period of one year was selected using the performance window attribute, as described in [26].

Due to the risk of data non-stationarity, we have opted to use out-of-time validation strategy, as in [20], instead of out-of-sample validation as used in [19] and [3]. Note that our results for the out-of-fold validation were consistently higher than that for the out-of-time validation for a range of architectures and hyperparameters, which is the common situation, as discussed in [14].

We have used a subset of credit applications from 16-month period for training and four-month period for the out-of-time validation. Training and validation sets were the same for each considered model and baseline.

Due to a large disparity between number of positive and negative cases (because of the low default rate at the bank), we settled on the following undersampling strategy: before each experiment we selected all the positive cases and 10 times as much randomly selected negative cases. On each training epoch we used all positive cases and an equal number of negative cases, selected from the pool of negative cases.

All models in this paper where trained on the last 800 transactions for each customer when available, padding by zero was applied when the actual transaction count for a client was lower.

4.2 Baselines

To compare our model with other approaches, we have implemented a logistic regression based model. We have also implemented an additional model that is based on the Gradient Boosting Machine (GBM) method [12].

Both logistic regression and GBM methods require a *large* number of *hand-crafted* aggregate features produced from the transactional data as an input to the classification model. An example of an aggregate feature would be an average spending amount in some category of merchants, such as hotels of the entire transaction history.

We used LightGBM[18] implementation of GBM algorithm and created nearly 7000 hand-crafted features for the application.

Similarly, for the logistic regression we manually designed about 400 aggregate features. Weight of evidence coding and binning of predictors [6] was used to transform categorical features.

4.3 Offline execution of our method

4.3.1 Encoder architecture selection. We have experimented with a different architectures of encoders, using Long Short Term Memory (LSTM), Bidirectional Recurrent Cells [25] and Gated Recurrent Units (GRU). The results of this comparison are presented in Table 2. Based of this comparison we decided to use one-layer GRU because the difference with the best performing bidirectional model was not statistically significant, while increasing complexity of the model and incurring a noticeable computational price.

Table 2: Encoder architecture comparison

Encoder	Valid ROC-AUC (STD)
GRU 1-layer	0.8155 (0.0015)
GRU 1-layer Bidirectional	0.8160 (0.0004)
LSTM 1-layer	0.8055 (0.0022)
LSTM 1-layer Bidirectional	0.8058 (0.0027)

4.3.2 Loss function and learning rate. We used a batch size of 32 for the training and the batch size of 768 for validation for all the experiments. When using ranking loss, we introduced the new hyperparameter *loss margin size*. We found that loss margin size of 0.1 gives the best results among all the loss hyperparameters that we tried, as shown in Table 3.

Table 3: Loss comparison

Loss	Valid ROC-AUC (STD)
BCE Loss	0.8124 (0.0016)
Hinge 0.5	0.8104 (0.0026)
Hinge 0.1	0.8168 (0.0017)
Hinge 0.01	0.8155 (0.0016)
Hinge 0.01 + BCE	0.8144 (0.0030)

Learning rate and learning rate reduction schedule is one of the most sensitive hyperparameters which can dramatically change the performance of the model. Note, that the optimal learning rate schedule depends heavily on loss function used, batch size and overall number of parameters in the model. We tried several learning rates and several learning rate reduction regimes and found that for both BCE loss and ranking loss the most effective strategy was an aggressive linear learning rate reduction with gamma=0.5, as shown in Table 4. We also tried unsuccessfully instead of monotonically decreasing the learning rate to vary it cyclically as proposed in [27].

Table 4: Learning rate schedules

Loss	Valid ROC-AUC (STD)
gamma = 1	0.8042 (0.0026)
gamma = 0.8	0.8144 (0.0015)
gamma = 0.5	0.8155 (0.0016)
gamma = 0.5, 2 cycles	0.8145 (0.0006)
gamma = 0.5, 3 cycles	0.8111 (0.0027)

4.3.3 Regularization methods. Due to the low number of positive classes, all models exhibit propensity for overfitting. Therefore we tried various types of dropout regularisation, such as:

- *Transaction dropout* that randomly drops some of the client transactions with defined probability
- *Transaction shuffle* that randomly permutes the order of client transactions
- *Dropout after embedding* that randomly zeroes some components after embedding layer

Note that, none of the aforementioned regularization methods proved effective against overfitting, as shown in Figure 2.

4.3.4 Ensembling methods. We tried several different types of ensembling methods:

- Simple averaging of model results. Averaging predictions of different models trained with distinct negative class examples leads to both increased accuracy and reduced variabilty of results, as shown in Figure 3

Figure 2: Regularization methods

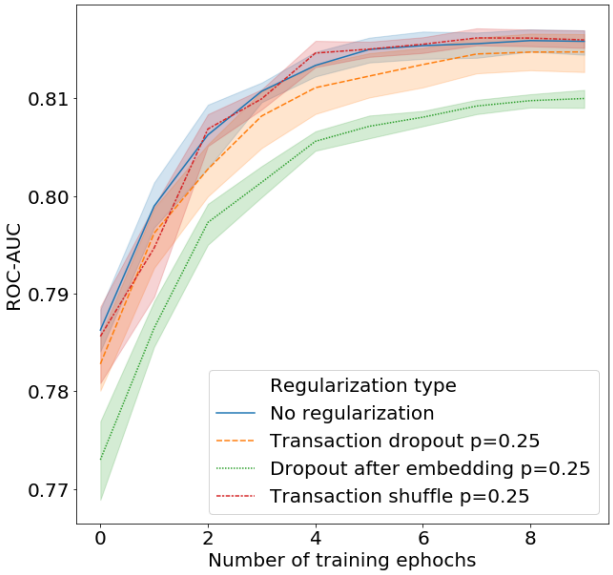
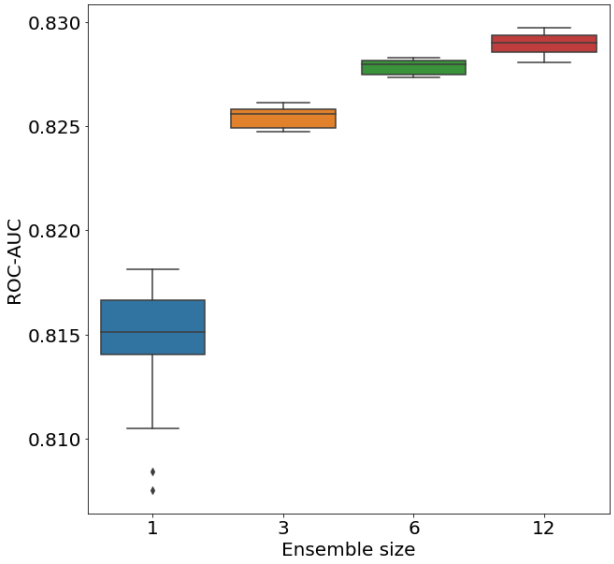


Figure 3: Ensemble quality comparison



- Stochastic Weight Averaging (SWA) [21]. Averaging the weights of ensemble models can significantly reduce inference time since only one model with averaged weights is used instead of the whole ensemble. But in our case averaging of weights of different models led to noticeable reduction in quality.
- Snapshot ensembling [17]. Using snapshots of the same model in the final ensemble can significantly reduce training time since only one model should be trained. Unfortunately this approach does not benefit from using distinct negative class examples
- SWA + snapshot ensembling. We found that combining SWA with snapshot ensembling for single model training by taking snapshots after a set epoch and averaging the weights leads to some reduction of variability, but the results were inconclusive and we opted for not using these advanced ensembling methods in our production model.

We opted to use a size six averaging ensemble for our production model, providing a reasonable compromise between model quality and training/inference times. As mentioned in Section 3.4, each model of the ensemble is trained on different subsamples of the negative class samples. As described in Section 4.1, we use undersampling procedure to reduce the number of negative samples. The negative samples are selected independently for each model of the ensemble, hence each model of the ensemble is trained on slightly different subset of negative samples.

4.4 Moving to production

We performed massive field test of our neural scoring model in a bank's production pipeline. We used the model trained on the same dataset as discussed in Section 4.1 to pre-calculate scores for each client with a debit or credit card. Training of a full six model ensemble took about 4 hours on a Tesla P100 GPU. It took about 17 minutes to score 1 million customers on an Tesla P100 GPU. And the inference time scales linearly with the number of clients.

These scores were used to make decisions about credit applications for tens of thousands of applicants during one month and the early results are very promising.

The potential financial gain was measured for the case if our model is used instead of the current scoring model for the applicants with enough transnational data. The preliminary financial results are measured in the millions of dollars per year, which constitutes a very significant result for the bank of this type and size.

5 RESULTS

Table 5 presents the main results of the experiments described in Section 4.

Table 5: Experiment comparison

	ROC AUC	N Features
Logistic regression	0.78	~ 400
LGBM	0.81	~ 7000
E.T.-RNN	0.83	12

As shown in Table 5, E.T.-RNN significantly outperformed the baselines on our data. Moreover, one of the crucial features of our approach is that we did not have to do feature engineering for our method, unlike the classical methods which rely heavily on the hand-crafted features (e. g. 400 features for Logistic regression and 7000 features for LGBM).

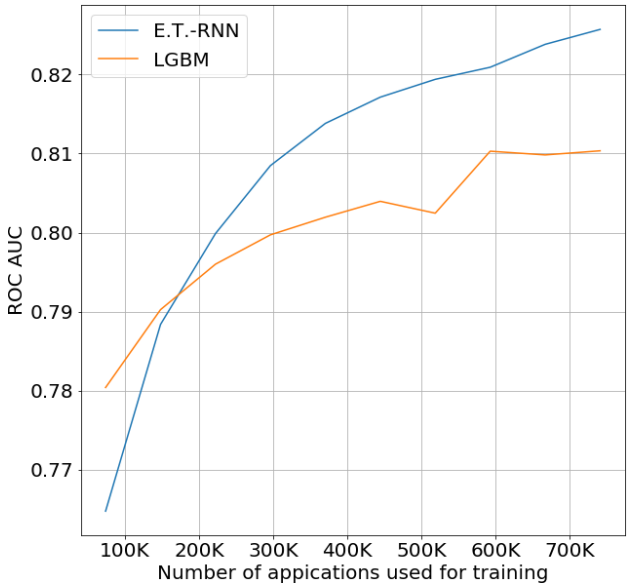
5.1 Training dataset size

Note that the results presented in Table 5 were achieved on the full dataset described in Section 4.1. We also conducted a series of experiments to estimate model performance for different dataset

sizes. As Figure 4, shows LGBM outperforms our approach for *small* volumes of data, as measured in terms of the number of applications (on the X axis). However, given enough data, E.T.-RNN method significantly outperforms the classical approaches. This observation is in line with the well-known understanding that neural networks outperform classical methods on large datasets.

Also note that E.T.-RNN has steeper learning curve than LGBM. Hence the performance gap would increase even further with more available data.

Figure 4: E.T.-RNN has steeper learning curve than LGBM.



5.2 Transaction count

Performance of our model depends heavily on the number of available transactions per client. As Figure 5 shows, scoring quality increases untill we reach around 350 transactions. Beyond this level, performance increase due to additional transactions is insignificant enough to be overshadowed by statistical variations in the data. Furthermore the share of clients having more than 350 transactions is about 50 percent for our dataset. This means that our model achieves significant hit rate when scoring clients of the bank. On the other hand, our method is still effective even for the applicants with a low number of transactions. For clients with more than 25 transactions (about 95 percent of total number of clients), we reach 82.5 ROC-AUC.

6 DISCUSSION

Our method worked well for the following reasons:

- Reasonably large number of customers in training dataset. Neural networks have lots of learnable parameters comparing to the classical approaches and, hence, require more data than classical methods. This is also true in our case as presented on Figure 4.

Figure 5: Classification quality vs number of transactions

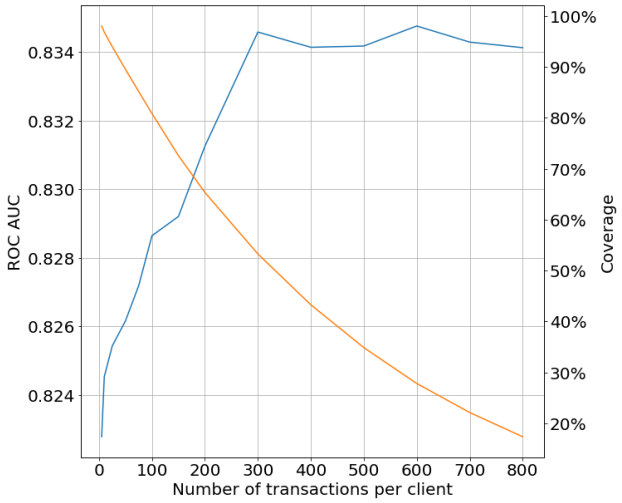
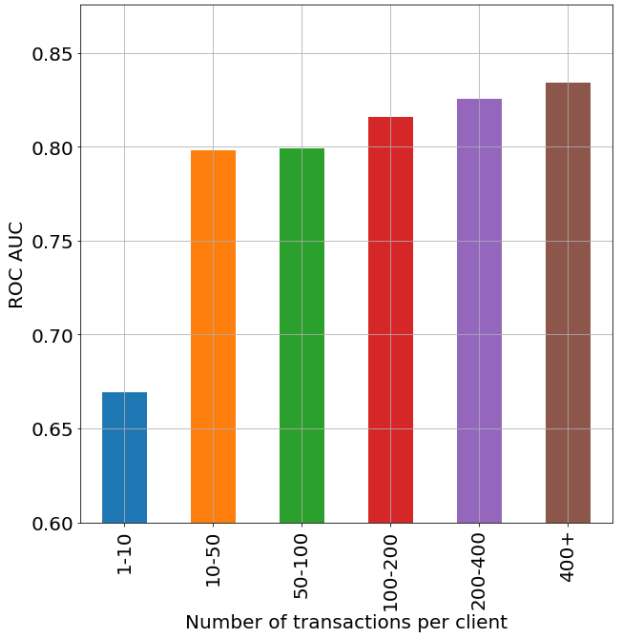


Figure 6: Classification quality for customers grouped by number of transactions



- Low-level, granular data. Our data can be described as a series of events and each event consists of several variables. Note that if data structure is relatively simple, our method may not work better than the traditional approaches. For example, for the data from the application questionnaire there is no need for sophisticated neural network models. Even classical ML approaches, like logistic regression would work reasonably well on the data with simple table-like structure.

- High-frequency data (as discussed before, more than 80 percents of customers have at least 100 transactions.

Our method worked because we applied sophisticated neural network method (as discussed in Section 4.3 we tried numerous other DL-based approaches, and many of them did not work that well) on the data exhibiting aforementioned characteristics.

To summarize, our E.T.-RNN approach would possibly work better than classical methods in cases where data is in low-level, granular form and there is enough data to train complex neural net based model.

7 CONCLUSIONS

In this paper we proposed a novel E.T.-RNN method which allows to use fine-grained transactional data for credit scoring. We tested our method against the benchmarks on the historical data and achieved superior performance. We also conducted a pilot study on banking customers and produced significant financial gains for the bank.

The significant advantage of our approach is that even complex multivariate time-series data can be directly used for training without any need for feature design. As was demonstrated in [11], the neural network learns meaningful internal representations of the input data during training, and this drastically reduces the need to generate hundreds or even thousands of hand-crafted aggregate features, as is typically done in credit scoring applications. This means that our method does not require any significant domain-specific expertise for feature design. Also, our model works exclusively on the transactional data and therefore does not require any additional input from the client that means that we can make credit loan decisions very fast, ideally in nearly real-time, because the whole credit scoring process is fully automated. Moreover information in the transactional data is exceptionally hard to forge. Hence, there is no need of costly checks for the correctness of such data, unlike data provided by the client or obtained from some other sources. Still another advantage of our method is that even a person without any credit history can be reliably accessed for credit-worthiness, his or her transactional history constituting a source for estimating credit risks. Finally, this method provides a *fair* approach to credit decision making because it *does not* rely on personal demographic information of an individual and, therefore, cannot discriminate applicants based on various demographic factors. For all these reasons, we believe that the proposed credit scoring approach has a potential to disrupt current loan practices in the retail banking industry.

One issue with our method is lack of interpretability. Neural networks constitute black-box models by their nature. The ability to produce rich models on top of raw data representation is the main strength of neural networks. But this ability also leads to significant interpretability problem, which is the main weakness of complex models. Also note that this issue is applicable not only to our method, but also to most other advanced machine learning methods, such as Gradient Boosting Machine, since they also suffer from the lack of interpretability.

Different organizations around the worlds have different philosophies regarding applying black-box models in credit scoring. In some countries, lack of interpertablilty is considered less appropriate while in other parts of the world it is considered more appropriate to do so. We believe that this issue will be less relevant moving forward because of the significant progress in solving the black-box interpretetaion problem, including that of neural networks, that have been achieved over the past few years. [9], [15], [23] constitute some examples of the recent work. Based of this progress, the black-box interpretation problem should be successfully addressed moving forward.

As a future work, we plan to study more effective method of regularization, which would allow us to use the data we have available more effectively. Furthermore, we plan to focus on even more effective ways to integrate time into our model. In particular our model is not sensitive to shifting all the customers transactions in time (e.g. shifting by one month back), and we plan to work on this problem. Finally we also plan to work on other types of loans, such as mortgage loans, which differ from retail loans in several respects.

REFERENCES

[1] Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. 2016. Fraud detection system: A survey. *Journal of Network and Computer Applications* 68 (2016), 90–113.

[2] João Bastos. 2008. Credit scoring with boosted decision trees.

[3] Tony Bellotti and Jonathan Crook. 2013. Forecasting and stress testing credit card default using dynamic models. *International Journal of Forecasting* 29, 4 (2013), 563–574.

[4] Daniel Björkegren and Darrell Grissen. 2017. Behavior revealed in mobile phone usage predicts loan repayment. *arXiv preprint arXiv:1712.05840* (2017).

[5] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[6] Novi MI Bruce Lund, Independent Consultant. 2016. Weight of Evidence Coding and Binning of Predictors in Logistic Regression. *MidWest SAS Users Group conference proceedings* (2016).

[7] Bo-Wen Chi and Chiun-Chieh Hsu. 2012. A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. *Expert Systems with Applications* 39, 3 (2012), 2650–2661.

[8] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014). arXiv:1406.1078 http://arxiv.org/abs/1406.1078

[9] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. 2016. RETAIN: Interpretable Predictive Model in Healthcare using Reverse Time Attention Mechanism. *CoRR* abs/1608.05745 (2016). arXiv:1608.05745 http://arxiv.org/abs/1608.05745

[10] David Durand. 1941. *Credit-Rating Formulae.* NBER, 83–91. http://www.nber.org/chapters/c12952

[11] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal* 1341, 3 (2009), 1.

[12] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[13] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).

[14] Dennis Glennon, Nicholas M Kiefer, C Erik Larson, and Hwan-sik Choi. 2008. Development and validation of credit scoring models. (2008).

[15] Pankaj Gupta and Hinrich Schütze. 2018. LISA: Explaining Recurrent Neural Network Judgments via Layer-wIse Semantic Accumulation and Example to Pattern Transformation. *arXiv preprint arXiv:1808.01591* (2018).

[16] Cheng-Lung Huang, Mu-Chen Chen, and Chieh-Jen Wang. 2007. Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications* 33, 4 (2007), 847 – 856. https://doi.org/10.1016/j.eswa.2006.07.007

[17] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, get M for free. *CoRR* abs/1704.00109 (2017). arXiv:1704.00109 http://arxiv.org/abs/1704.00109

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *NIPS*.

[19] Amir E Khandani, Adlar J Kim, and Andrew W Lo. 2010. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance* 34, 11

(2010), 2767–2787.
[20] HÃĕvard Kvamme, Nikolai Sellereite, Kjersti Aas, and Steffen Sjursen. 2018. Predicting mortgage default using convolutional neural networks. *Expert Systems with Applications* 102 (2018), 207 – 217. https://doi.org/10.1016/j.eswa.2018.02.029
[21] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic Gradient Descent with Restarts. *CoRR* abs/1608.03983 (2016). arXiv:1608.03983 http://arxiv.org/abs/1608.03983
[22] Paul Makowski. 1985. Credit scoring branches out. *Credit World* 75, 1 (1985), 30–37.
[23] R. Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. 2019. RNNs implicitly implement tensor-product representations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJx0sjC5FX
[24] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.
[25] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.
[26] Naeem Siddiqi. 2005. Credit Risk Scorecards: Developing And Implementing Intelligent Credit Scoring. (2005).
[27] Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 464–472.
[28] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, Cambridge, MA, USA, 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173
[29] Ellen Tobback and David Martens. 2017. *Retail credit scoring using fine-grained payment data*. Working Papers. University of Antwerp, Faculty of Business and Economics. https://EconPapers.repec.org/RePEc:ant:wpaper:2017011
[30] David West. 2000. Neural network credit scoring models. *Computers & Operations Research* 27, 11-12 (2000), 1131–1152.
[31] Bénard Wiese and Christian Omlin. 2009. *Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time with LSTM Recurrent Neural Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 231–268. https://doi.org/10.1007/978-3-642-04003-0_10
[32] John C. Wiginton. 1980. A Note on the Comparison of Logit and Discriminant Models of Consumer Credit Behavior. *Journal of Financial and Quantitative Analysis* 15, 03 (1980), 757–770. https://EconPapers.repec.org/RePEc:cup:jfinqa:v:15:y:1980:i:03:p:757-770_00
[33] Yishen Zhang, Dong Wang, Yuehui Chen, Huijie Shang, and Qi Tian. 2017. Credit Risk Assessment Based on Long Short-Term Memory Model. In *International Conference on Intelligent Computing*. Springer, 700–712.