# Customer Shopping Pattern Prediction: A Recurrent Neural Network Approach

Hojjat Salehinejad and Shahryar Rahnamayan, *Senior Member, IEEE*
Department of Electrical, Computer, and Software Engineering University of Ontario Institute of Technology
Oshawa, Ontario, Canada
{hojjat.salehinejad, shahryar.rahnamayan}@uoit.ca

***Abstract*—Customer relationship management is a popular and strategic topic in marketing and quality of service. The availability of big transactions data as well as computing systems have provided a great opportunity to model and predict customer behaviour. However, there is a lack of modern modelling and analytical methods to perform analysis on such data. Deep learning techniques can assist marketing decision makers to provide more reliable and practical marketing strategic plans. In this paper, we propose a customer behaviour prediction model using recurrent neural networks (RNNs) based on the client loyalty number (CLN), recency, frequency, and monetary (RFM) variables. The experiment results show that RNNs can predict RFM values of customers efficiently. This model can be later used in recommender systems for exclusive promotional offers and loyalty programs management.**

***Index Terms*—Customer Behaviour Prediction, Deep Learning, Recurrent Neural Networks, Recency Frequency Monetary (RFM), Recommender System, Shopping Pattern.**

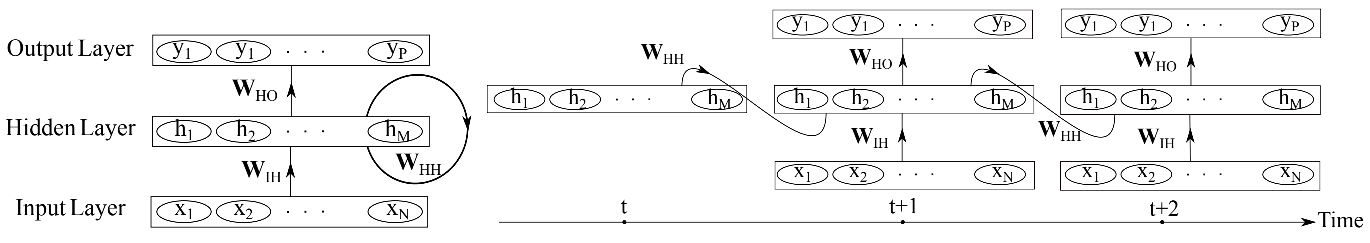
## I. Introduction

Customer relationship management is an important topic in marketing and e-commerce. One of the most popular methods to deal with this challenge is using customer lifetime value (CLV) model. CLV is defined as the value of relationship with the current customers, by considering the future cash flows from the relationship with the customers [1]. Since acquiring new customers is mostly more expensive than keeping current customers, the CLV model attempts to manage the long-term relationships, rather than short-term ones. The decisions are generally made upon classification of customers to a number of loyalty classes. Then, marketing decision makers can manage marketing communication programs through methods such as personalized advertisements and exclusive promotional offers [2].

The CLV models use different strategies for customer behaviour modelling. One of the most reliable ones is using the recency (R), frequency (F), and monetary value (M) variables, called RFM [3], [4], [5]. These variables present some understanding of customer's behaviour and try to answer the following questions: "How recently did the customer purchase?", "How often do they purchase?", and "How much do they spend?" [2]. RFM variables are sufficient statistics for customer behaviour modelling and are a mainstay of the industry because of their ease of implementation in practice [6], [3]. With the explosion of big-data and availability of online and offline transaction data, correct modelings of CLV and prediction of customer behaviour using RFM factors can results in firm revenues, profitability for the market, and more loyalty for customers. Utilizing advanced machine learning techniques is one of the approaches to develop such models.

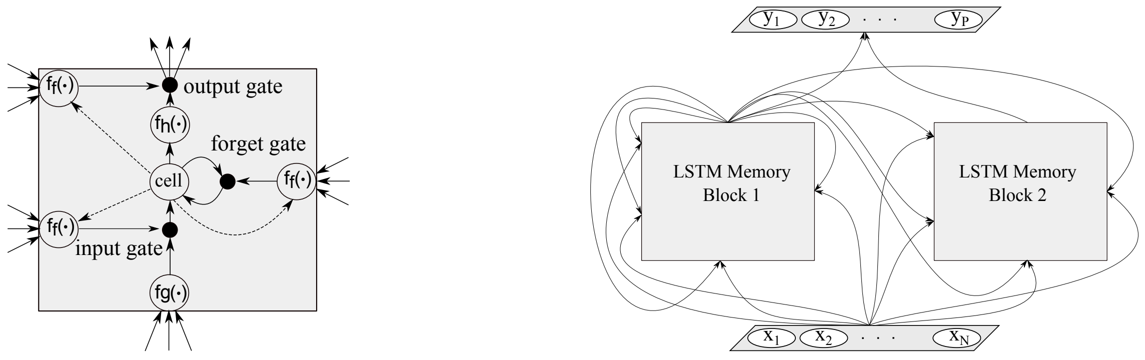
Some attempts toward customer behaviour prediction are proposed recently. For restaurant preference prediction, a model based on artificial neural networks (ANNs) is proposed in [7]. This model incorporates social media location check-ins, historical preferences of the customer, the influence of the customer's social network, and customer's mobility characteristics as inputs to the model. The RFM variables are used in [8] for finding segments of retailers from a large amount of Electronic Funds Transfer at Point Of Sale (EFTPOS) transaction data. In [9], the customer purchase behaviour prediction is made by finding the association among products and exploiting customer′s motivation. Then the customer preferences for product features are learned using probabilistic models to match products to customers. The ANNs are used for RFM prediction of blood donors in [10]. In this approach, a basic version of ANNs is used, which considers time as a separate input variable.

The ANNs have been used for different prediction and classification applications, including traffic prediction [11], pistachio nuts classification [12], and phenotype prediction in genomics [13]. Deep learning models refer to ANNs with more than one hidden layer. It is a representation technique, which gives a learning machine the ability to receive raw data and find its representation for further processing and decision making. Such machines are made from non-linear but simple units, which can provide different levels of data representation through their multi-layer architecture. The higher layers provide a more abstract representation of data and suppresse irrelevant variations [14]. Many naturally occurring phenomena are complex and non-local sequences such as music, speech, or human motion. One of the key challenges in sequence transduction is learning to represent both the input and output sequences in a way that is invariant to sequential distortions such as shrinking, stretching, and translating. The modified version of feed-forward neural networks (FFNNs) by adding recurrent connections is called recurrent neural networks (RNNs), which are capable of modelling sequential data for sequence recognition, sequence production, and time series prediction [15]. The RNNs are made of high dimensional hidden states with non-linear dynamics. The



(a) Folded SRNN. (b) Unfolded SRNN through time.

Fig. 1: A simple recurrent neural network (SRNN) and its unfolded structure through time. To keep the figure simple, biases are not shown.



(a) LSTM block. (b) LSTM network.

Fig. 2: Long short term memory (LSTM) architecture.

structure of hidden states work as the memory of network and state of the hidden layer at a time is conditioned on its previous state. This enables the RNNs to store, remember, and process past complex signals for long time periods and further decision making. In this way, the RNN can map an input sequence to the output sequence at the current time-step and predict the sequence in the next time-step. The handwriting recognition [16] and speech recognition [17] are examples of RNN approaches.

Deep learning methods based on RNNs are practical tools for modelling time-series data. However, such methods have not been discussed in the literature for RFM or customer value prediction, to the best of our knowledge. In this paper, for the first time, a RNN model for customer behaviour prediction based on the R, F, and M variables are presented. The model is consisted of one hidden layer with rectified linear units (ReLU). Since the customer id (also known as client loyalty number (CLN)) is usually a long integer number, we use an auto-encoder inside the model to extract features from client loyalty number and pass the features along with the R, F, and M values to the learning model as a sequence at time $t$. Prediction of the R, F, and M values at time-step $t+1$ is the target of the learning model.

The rest of paper is organized as follow. Next section provides an overview on a simple RNN (SRNN). The proposed model is discussed in Section III. The experiment results are analyzed in Section IV and the paper is concluded in Section V and some future research challenges are introduced.

## II. Recurrent Neural Networks

A RNN can be seen as a FFNN by unfolding the recurrent cycles over time. As it is demonstrated in Figure 1, a SRNN refers to a one step RNN [18]. Generally in order to train a RNN, we need a training dataset $X$ and a disjoint test dataset $Z$. The sets are consisted of input-target paris, where the objective is to train the network with the training set $X$ and evaluate it with the test set $Z$. During the training procedure, the objective is to train the network (i.e. optimizing the weights) such a way that the error between the input and target pairs is minimized. This error is defined by a loss function as:

$$\mathcal{L}(\mathbf{y},\mathbf{z}) = \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{y}_t,\mathbf{z}_t), \qquad (1)$$

where $\mathbf{y}_t$ is the estimated output.

In general, a SRNN is consisted of input, hidden, and output layers, where each layer is consisted of corresponding units, Figure 1a, [19]. The input layer is consisted of $N$ input units, where its inputs are defined as a sequence of vectors through time $t$ such as $\{..., \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}...\}$ where $\mathbf{x}_t = (x_1, x_2, ..., x_N)$. In a fully connected SRNN, the inputs units are connected to hidden units in the hidden layer, where the connections are defined with a weight matrix $\mathbf{W}_{IH}$. The hidden layer is consisted of $M$ hidden units $\mathbf{h}_t = (h_1, h_2, ..., h_M)$, which are connected to each other through time with recurrent connections. As it is demonstrated in Figure 1b, the hidden units are initiated before feeding the inputs. This initialization should address the network state before seeing the input sequences [20]. It is believed that

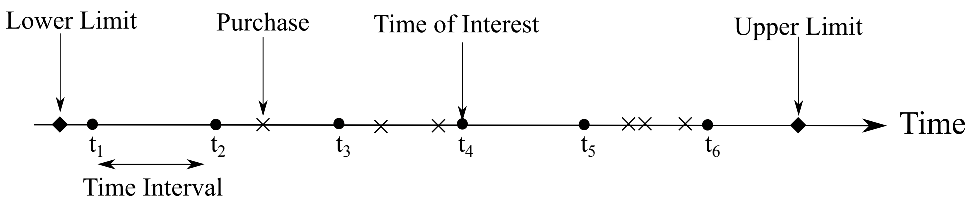


Fig. 3: A sample of shopper's behaviour during different time intervals.

using non-zero elements can improve overall performance and stability of the network [21]. The hidden layer structure defines the state space or "memory" of the system, defined as:

$$\mathbf{h}_t = f_H(\mathbf{o}_t), \tag{2}$$

where

$$\mathbf{o}_t = \mathbf{W}_{IH}\mathbf{x}_t + \mathbf{W}_{HH}\mathbf{h}_{t-1} + \mathbf{b}_h, \tag{3}$$

and $f_H(.)$ is the hidden layer activation function and $\mathbf{b}_h$ is the bias vector of the hidden units [1]. The hidden units are connected to the output layer with weighted connections $\mathbf{W}_{HO}$. The output layer has $P$ units such as $\mathbf{y}_t = (y_1, y_2, ..., y_P)$ which are estimated as:

$$\mathbf{y}_t = f_O(\mathbf{W}_{HO}\mathbf{h}_t + \mathbf{b}_o), \tag{4}$$

where $f_O(.)$ is the activations functions and $\mathbf{b}_o$ is the bias vector in the output layer. Since the input-target pairs are sequence through time, the above steps are repeated consequently over time $t = (1, ..., T)$ as well.

As Eqs. (2) and (4) show, the RNNs are dynamic systems with certain nonlinear state equations, which are iterable through time, [19]. In each timestep, the input vector is received and the current hidden states are updated to provide a prediction at the output layer. The hidden state of RNN is a set of values, which apart from the effect of any external factors, summarizes all the unique necessary information about the past states of the network over many timesteps. This integrated information can be used to define future behaviour of the network and make accurate predictions at the output layer, [22]. As the model presents, the non-linearity structure utilized by each unit is simple; however, this simple structure is capable of modelling rich dynamics, if it is well iterated through time.

## A. *Long-Short-Term Memory (LSTM)*

The LSTM model changes the structure of hidden units from "logistic" or "tanh" to memory cells which their inputs and outputs are controlled by gates, Figure 2, [19]. This modification helps the network to learn and remember long-term dependencies better. Each memory cell is consisted of four inputs but one output. A typical LSTM cell is made of input, forget, and output gates and a cell activation component [19]. These units receive the activation signals from different sources and control the activation of the cell by the designed multipliers. The LSTM gates can prevent the rest of the network from modifying the contents of the memory cells for multiple time steps. The LSTM networks preserve signals and propagate errors for much longer than SRNNs. These properties allow LSTM networks to process data with complex and separated interdependencies and to excel in a range of sequence learning domains. The input gate of LSTM is defined as:

$$\mathbf{g}^i_t = \sigma(\mathbf{W}_{Ig^i}\mathbf{x}_t + \mathbf{W}_{Hg^i}\mathbf{h}_{t-1} + \mathbf{W}_{g^c g^i}\mathbf{g}^c_{t-1} + b_{g^i}), \tag{5}$$

where $\mathbf{W}_{Ig^i}$ is the weight matrix from the input layer to the input gate, $\mathbf{W}_{Hg^i}$ is the weight matrix from hidden state to the input gate, $\mathbf{W}_{g^c g^i}$ is the weight matrix from cell activation to the input gate, and $b_{g^i}$ is the bias of the input gate. The forget gate is defined as:

$$\mathbf{g}^f_t = \sigma(\mathbf{W}_{Ig^f}\mathbf{x}_t + \mathbf{W}_{Hg^f}\mathbf{h}_{t-1} + \mathbf{W}_{g^c g^f}\mathbf{g}^c_{t-1} + b_{g^f}), \tag{6}$$

where $\mathbf{W}_{Ig^f}$ is the weight matrix from the input layer to the forget gate, $\mathbf{W}_{Hg^f}$ is the weight matrix from hidden state to the forget gate, $\mathbf{W}_{g^c g^f}$ is the weight matrix from cell activation to the forget gate, and $b_{g^f}$ is the bias of the forget gate. The cell gate is defined as:

$$\mathbf{g}^c_t = \mathbf{g}^i_t \; tanh(\mathbf{W}_{Ig^c}\mathbf{x}_t + \mathbf{W}_{Hg^c}\mathbf{h}_{t-1} + b_{g^c}) + \mathbf{g}^f_t \mathbf{g}^c_{t-1}, \tag{7}$$

where $\mathbf{W}_{Ig^c}$ is the weight matrix from the input layer to the cell gate, $\mathbf{W}_{Hg^c}$ is the weight matrix from hidden state to the cell gate, and $b_{g^c}$ is the bias of the cell gate. The output gate is defined as:

$$\mathbf{g}^o_t = \sigma(\mathbf{W}_{Ig^o}\mathbf{x}_t + \mathbf{W}_{Hg^o}\mathbf{h}_{t-1} + \mathbf{W}_{g^c g^o}\mathbf{g}^c_t + b_{g^o}), \tag{8}$$

where $\mathbf{W}_{Ig^o}$ is the weight matrix from the input layer to the output gate, $\mathbf{W}_{Hg^o}$ is the weight matrix from hidden state to the output gate, $\mathbf{W}_{g^c g^o}$ is the weight matrix from cell activation to the output gate, and $b_{g^o}$ is the bias of the output gate. The hidden state is computed as:

$$\mathbf{h}_t = \mathbf{g}^o_t \; tanh(\mathbf{g}^c_t). \tag{9}$$

## B. *Rectified Linear Units (ReLU)*

Gradient of standard nonlinear functions (e.g. $sigmoid$ function) is one of the main reasons of gradient vanishing problem. The ReLU is a potential approach to tackle this problem [23]. The ReLU units can have very large outputs in RNNs. This may increase probability of gradient exploding problem comparing to the bounded values. However, by setting biases to zero and using an identity matrix to initialize the recurrent weights, it is likely to get comparative solution to

[1]The hidden state model in Eq. 2 is sometimes mentioned as $\mathbf{h}_t = \mathbf{W}_{IH}\mathbf{x}_t + \mathbf{W}_{HH}f_H(\mathbf{h}_{t-1}) + \mathbf{b}_h$, where both equations are equivalent.

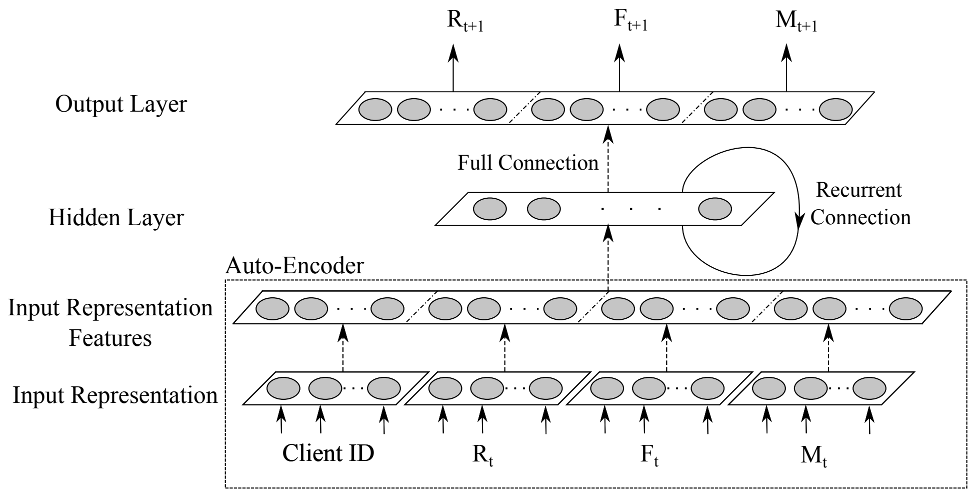


Fig. 4: Proposed RNN model with ReLU activation function and input auto-encoder. Slashed arrows represent full weights connection. Solid arrows represent inputs, output, and full recurrent connection. The weights and biases are not showed for simplicity. The image is not scaled.

LSTM [24]. The ReLU computes the output as:

$$y(x) = max(x, 0), \tag{10}$$

which leads to faster training and sparser gradients [25].

## III. Proposed Model

The customer shopping is generally recorded as a transaction, summarizing purchase at each visit. Since the transactions are happening through time, the customer shopping pattern can be modelled as a time-series discrete phenomena. We develop this model in the next subsection. Then after computing the RFM values for each customer to a desired date, the proposed RNN model in subsection III-B will be trained for RFM prediction.

### *A. Customer Shopping Pattern Model*

We study the customer behaviour though time with equal time steps (intervals) as demonstrated in Figure 3. The time interval can be weekly, bi-weekly, monthly, or etc. Since the first purchase time among customers is different, we define a lower limit and upper limit during the time. The lower limit refers to the start point of our study and the upper limit refers to the end point of our study through time. In this case, we can define some equal time intervals between the lower and upper limits, as shown in Figure 3. The shopper's purchase is then identified in each time interval and the R, F, and M variables are computed with respect to any point of interest. For example, if the point of interest is at $t_4$, the recency is the time difference between the last purchase before the time of interest and the purchase itself. The time difference can be represented in scale of hour, day, week or etc., depending on the application. The frequency is the number of conducted purchases between the lower limit and the time of interest, which is $F = 3$ in this example. The monetary is the value customer has spent on the purchases between the lower limit and the time of interest. The R, F, and M values are computed for each customer with a CLN and for all times of interest.

### *B. Learning Machine Architecture*

The proposed RNN model is consisted of one input layer, one hidden (recurrent) layer, and one output layer. The input layer is an auto-encoder which extracts features from inputs. The CLN, R, F, and M values for each customer at each time-step $t$ are the input sequence to the RNN model presented in Figure 4. The R, F, and M value of the next time-step $t+1$ is shown to the model as target through predefined time intervals. The time-step $t$ can be set depending on the application. For example, for grocery stores it can be weekly or bi-weekly and for sports wear every season.

In general, the CLNs are provided as large integer digits in transaction data. We use the one-hot encoding method to break the dependencies between integers. For example, if a store has 50,000 customer, with loyalty numbers starting from 100000, the one-hot encoded representation for CLN $u = 100125$ is $\mathbf{u} = [0, ..., 0, 1, 0, ..., 0]_{1\times 50,000}$, where only the element number 125 is one and the rest are zero, Figure 4.

The one-hot encoded CLN and binary representation of $R_t$, $F_t$, and $M_t$ are fed to an auto-encoder, which represents each input vector with a feature representation vector of fixed length. Each input vector is fully connected to the representation layer, where $\mathbf{W}_{VI}$ is the weight matrix to be optimized while training the model. The feature representation is:

$$\mathbf{v} = \mathbf{W}_{VI}\mathbf{u}, \tag{11}$$

where $\mathbf{v}$ is the feature representation of each input parameter CLN, $R_t$, $F_t$, and $M_t$ at time $t$. Then the features are concatenated such as $\mathbf{x}_t = [\mathbf{v}, \mathbf{r}_t, \mathbf{f}_t, \mathbf{m}_t]$ and fed to the recurrent layer. Therefore, the set of weights to be optimized is $\theta = \{\mathbf{W}_{IH}, \mathbf{W}_{HH}, \mathbf{W}_{HO}, \mathbf{W}_{VI-CLN}, \mathbf{W}_{VI-R}, \mathbf{W}_{VI-F}, \mathbf{W}_{VI-M}, \mathbf{b}_H, \mathbf{b}_O\}$ where $\mathbf{W}_{VI-CLN}$, $\mathbf{W}_{VI-R}$, $\mathbf{W}_{VI-F}$, and $\mathbf{W}_{VI-M}$ are the feature representation for the CLN, R, F, and M values, respectively; $\mathbf{b}_H$ and $\mathbf{b}_O$ are biases in the hidden and output layer, respectively.

By using this trick, the models can learn features of CLN,

TABLE I: Parameters setting of the recurrent neural network model.

| Parameter | Value |
|---|---|
| Number of hidden units | 250 |
| Number of epochs | 1000 |
| Number of auto-encoder features | 60 |
| Number of outputs | 24 |
| Number of cross validations | 10 |
| Time Interval | Weekly |
| $L_1$ | 0.0001 |

TABLE II: Performance results of the SRNN, LSTM-RNN and ReLU-RNN models for the test dataset. The outstanding result is in bold.

| Model | R | F | M | Total |
|---|---|---|---|---|
| SRNN | 68% | 64% | 69% | 67% |
| LSTM-RNN | 72% | 79% | 80% | 77% |
| ReLU-RNN | 78% | 82% | 79% | **80%** |

$R_t$, $F_t$, and $M_t$ automatically through training procedure. Since the number of extract features from auto-encoder is identical for each input parameter, the model provides equal opportunity for each input representation.

This model is extendable in deeper layer for more abstract representation. The input vector $\mathbf{x}_t$ is used as in Eq. 3 to go through the hidden layer and then the output layer. The output sequence are the predicted binary R, F, and M variables of the future time-step which are compared with the target sequence $\mathbf{z}_{t+1} = [R_{t+1}, F_{t+1}, M_{t+1}]$ to compute the loss value.

## IV. Experiments

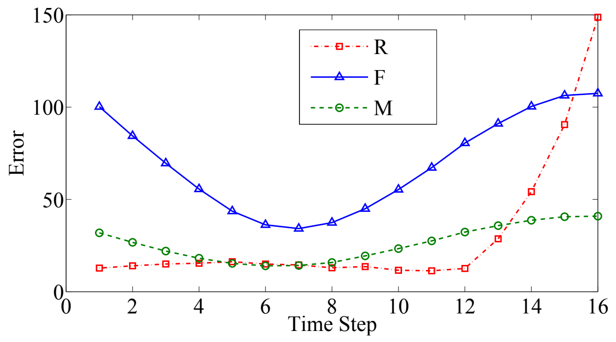
### A. Data Set

The data set used in our experiments is ta-feng dataset[2], containing 817,741 transactions belonging to 32,266 users and 23,812 items.

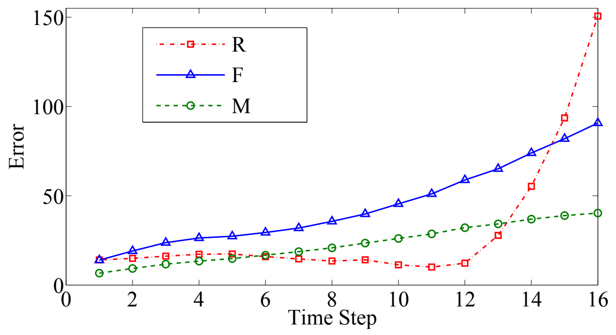
### B. Settings

Before training the models, the R, F, and M variables are computed for each time interval (time-step) and for each CLN. The data is divided into training (50%), validation (25%), and test (25%) partitions. The data is shuffled for each experiment. In order to have the R,F, and M values in same scale, the data is normalized. The experiments are cross-validated 10 times. A summary of model parameters is in Table I. The hyper-parameters of the models are selected based on the cross validation. The regularization coefficient is set to 0.0001. Each binary representation has 8 bits. Therefore, the number of target is $3 \times 8$=24. Each input parameter is represented with 20 features in auto-encoder. To break the symmetry in weights, the initial value of connections weights are set to small random values with uniform distribution. The optimization method is stochastic gradient descent (SGD) and the mini-batch size is 120.

[2]http://recsyswiki.com/wiki/Grocery_shopping_datasets



(a) LSTM.



(b) ReLU.

Fig. 5: Error of testing dataset for recency (R), frequency (F), and monetary (M) variables over different time steps for the ReLU and LSTM methods.

### C. Performance Analysis

In this subsection we analyse performance of the proposed RNN with ReLU activation function (ReLU-RNN) model with LSTM-RNN and SRNN. The performance results on the test dataset in Table II show that the RNN models have competitive performance for RFM recommender system. The ReLU activation function has slightly better performance than LSTM and SRNN with a total success rate of 80%. The ReLU-RNN has better performance (i.e. 78%) for the Recency and (i.e. 82%) the Frequency parameters. This model has a 79% performance for the Monetary while the LSTM-RNN performs slightly better with 80%.

Error rate of the LSTM-RNN and ReLU-RNN regarding long-term dependency through time-steps is presented in Figure 5. Both plots are scaled for sake of comparison. The plots show that as the number of time-steps increases, we observe more error in each time-step. This is mostly due to the long-term dependency between the data through time. As the training proceeds toward more time-steps and receives more new data, it forgets more about past and the total error increases. We observe that both models are competitive, but the ReLU-RNN has slightly better error rate. The LSTM shows a curve-shape behaviour for the Frequency parameter which should be due to the nature of this parameter through time.

## V. Conclusion And Further Challenges

The recency (R), frequency (F), and monetary (M) values are widely used in industry and academia to model history of customer behaviour and plan for future marketing strategies. This paper proposes a new model for RFM prediction of customers based on recurrent neural networks (RNNs) with rectified linear unit activation function. The model utilizes an auto-encoder to represent features of input parameters (i.e. customer loyalty number, R, F, and M).

The proposed model is the first of its kind in the literature and has many opportunities for further improvement. The model can be improved by using more training data. It is interesting to explore deeper structures of the model in auto-encoder and recursion levels. Clumpiness is another variable which can be studied as an additive to R, F, and M (i.e. RFMC) variables. Another pathway is considering other parameters of user (e.g. location, age, and etc.) for automatic feature extraction and further development of recommender systems.

## References

[1] S. Gupta, D. R. Lehmann, and J. A. Stuart, "Valuing customers," *Journal of marketing research*, vol. 41, no. 1, pp. 7–18, 2004.

[2] P. W. Farris, N. T. Bendle, P. E. Pfeifer, and D. J. Reibstein, *Marketing metrics: The definitive guide to measuring marketing performance*. Pearson Education, 2010.

[3] Y. Zhang, E. T. Bradlow, and D. S. Small, "Predicting customer value using clumpiness: From rfm to rfmc," *Marketing Science*, vol. 34, no. 2, pp. 195–208, 2014.

[4] V. Kumar, *Customer lifetime value: the path to profitability*. Now Publishers Inc, 2008.

[5] P. Fader, *Customer centricity: Focus on the right customers for strategic advantage*. Wharton digital press, 2012.

[6] P. S. Fader, B. G. Hardie, and K. L. Lee, "Rfm and clv: Using iso-value curves for customer base analysis," *Journal of Marketing Research*, vol. 42, no. 4, pp. 415–430, 2005.

[7] B. Zheng, K. Thompson, S. S. Lam, S. W. Yoon, and N. Gnanasambandam, "Customers' behavior prediction using artificial neural network," in *IIE Annual Conference. Proceedings*. Institute of Industrial Engineers-Publisher, 2013, p. 700.

[8] A. Singh, G. Rumantir, A. South, and B. Bethwaite, "Clustering experiments on big transaction data for market segmentation," in *Proceedings of the 2014 International Conference on Big Data Science and Computing*. ACM, 2014, p. 16.

[9] J. Qiu, Z. Lin, and Y. Li, "Predicting customer purchase behavior in the e-commerce context," *Electronic Commerce Research*, vol. 15, no. 4, pp. 427–452, 2015.

[10] Q. K. Al-Shayea and T. K. Al-Shayea, "Customer behavior on rfmt model using neural networks," in *Proceedings of the World Congress on Engineering*, vol. 1, 2014.

[11] H. Salehinejad and S. Talebi, "A new ant algorithm based vehicle navigation system: a wireless networking approach," in *Telecommunications, 2008. IST 2008. International Symposium on*. IEEE, 2008, pp. 36–41.

[12] S. Mahdavi-Jafari, H. Salehinejad, and S. Talebi, "A pistachio nuts classification technique: An ann based signal processing scheme," in *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*. IEEE, 2008, pp. 447–451.

[13] F. Pouladi, H. Salehinejad, and A. M. Gilani, "Recurrent neural networks for sequential phenotype prediction in genomics," in *Developments of E-Systems Engineering (DeSE), 2015 International Conference on*. IEEE, 2015, pp. 225–230.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[16] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in neural information processing systems*, 2009, pp. 545–552.

[17] B. Schrauwen and L. Buesing, "A hierarchy of recurrent networks for speech recognition," in *Deep Learning for Speech Recognition and Related Applications*, 2009.

[18] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5528–5531.

[19] H. Salehinejad, "Learning over long time lags," *arXiv preprint arXiv:1602.04335*, 2016.

[20] A. Graves, "Supervised sequence labelling," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, pp. 5–13.

[21] H. Zimmermann, R. Grothmann, A. Schaefer, and C. Tietz, "Identification and forecasting of large dynamical systems by dynamical consistent neural networks," *New Directions in Statistical Signal Processing: From Systems to Brain*, pp. 203–242, 2006.

[22] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[23] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8624–8628.

[24] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[25] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean *et al.*, "On rectified linear units for speech processing," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3517–3521.