Discovering Sequential Patterns by Neural Networks

Jakub Nowak, Marcin Korytkowski, Rafał Scherer
Czestochowa University of Technology
Al. Armii Krajowej 36, 42-200 Czestochowa, Poland
{jakub.nowak, marcin.korytkowski, rafal.scherer}@pcz.pl

***Abstract*—Sequential pattern mining can discover many interesting phenomena such as bank transactions, web page requesting sequences, customer behavior, etc. There have been many frequent itemset mining algorithms proposed so far, yet it is still a challenging task. In this paper, we propose a deep learning architecture for discovering closed sequences. The U-Net network is trained with random, synthetic sequences and, afterward, is able to discover unknown (not seen during training) sequences. The proposed solution is faster than traditional sequential data mining methods for longer sequences.**

***Index Terms*—convolutional neural networks, sequential pattern mining, closed sequences**

I. Introduction

Detecting recurring events is a very interesting and important issue. Based on the collected knowledge, it is possible to determine if, after the occurrence of events, other previously observed events may be repeated. This knowledge can be used to analyze sales data to define associations between products better, to observe natural phenomena, including determining the probabilities of cataclysms or analyzing stock market trends, bank transactions, software events, web page requesting sequences, customer behavior, manufacturing processes [1]–[4] or network traffic [5]–[7].

One of the first works formalizing issues related to sequences was frequent itemset mining (FIM) by Agrawal and Srikant in [8]. It is a different problem as it concerns basket-item data and associations. They introduced three algorithms from the Apriori family to solve the problem. In the research, the most important emphasis was put on the performance of algorithms using their sequence generator (now unavailable). They compared the impact of the minimal sequence support in the database on the calculation time. The Apriori methods require that the searched sequence is closely related in each example (there is no ABC sequence in the ABDC sequence).

The paper was followed by [9], where the Generalized Sequential Patterns algorithm (GSP) was presented. GSP was improved in three ways. There were possible gaps between consecutive elements of the sequence. It was possible to define time constraints. The concept of taxonomy was introduced. Another development was the SPADE algorithm [10]. SPADE searched the database in a vertical way, which means that it looked for associations between individual elements and then created a larger sequence of them, which translated into better performance. A more detailed comparison of SPADE and GSP can be found in [11]. The next was the BIDE algorithm [12] [13], which was designed for closed sequences for improving efficiency.

The detection of anomalies in computer systems is currently a very developed field. In our work, we rely on event sequences represented as numbers (from a dictionary), but this is not the only approach. The work [14] presents a sequential analysis of system logs using natural language processing. No ID number is assigned to a specific event, as in our case, and the information contained in the event description is analyzed. As shown by the concept of [15], the most important in the analysis are the log entries recorded just before the error, both during the attack on the system and during errors caused by the error that occurred during the operation of the system. Using this approach allows limiting the amount of data to be analyzed by the system.

In the paper, we train a neural network to recognize unknown closed sequences. Training is performed with the input data containing sequences mixed with random numbers. The desired output is the clean sequence without the embedded noise. Testing is performed on sequences unseen during training. Moreover, even numbers creating new sequences were from a different interval, not used during training. Thus, the approach presented in the paper is not classification or pattern matching what a typical application of neural networks is. It is also not a regular expression matching (such as the GREP utility). Our initial experiments with LSTM networks [16] were unsuccessful.

The advantages of the proposed approach are speed in the case of long sequences and a possibility to discover transposed sequences. We use the U-Net architecture, described in Section II with one-hot number-level encoding. In the experiments, sequential data for the analysis are generated by the authors. Datasets for supervised neural network training must be prepared to include output target sequences. To train the network, we had to generate a data set where the input data are sequences mixed with random numbers, and the output data are clean sequences to be found (Section III and Fig. 4). We provided the experiments comparing the speed with the traditional algorithms and showing the ability to detect transposed sequences in Section IV.

Through this research, we highlight the following features and contributions of the proposed model.

- We present the first neural network-based detection of unknown closed sequences.

The project financed under the program of the Polish Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019–2022 project number 020/RID/2018/19, the amount of financing 12,000,000.00 PLN.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

- Our work provides new insights, showing that neural networks can detect unseen before closed sequences with nearly 100 % accuracy.
- In the case of long sequences, our approach works faster as the neural network analyses a long window at once.

The remainder of the paper is organised as follows. In Section II, we described the neural network architecture. Section III described data preparation. Section IV describes experiments comparing the method with some standard sequential mining algorithms. Finally, conclusions and discussions of the paper are presented in Section V.

II. U-Net Convolutional Neural Network

The U-Net network [17] is a special architecture of convolutional networks pioneered in [18]. It was designed for semantic segmentation of medical images to detect tumors or changes in X-ray images for relatively small sets of training data. This ability was the main rationale behind this choice. It is not possible to generate all possible sequences for training due to a combinatorial explosion. Moreover, for the nature of the problem, there is no point in performing data augmentation. Earlier, we performed experiments with various standard convolutional networks without success.

The U-Net network has an encoder-decoder structure; the encoder that processes the image in the first part of the network is designed to extract the most important elements. A max-pooling operation with size 2×2 is used to reduce the feature map by selecting the elements with the highest value. The reverse process is up-pooling (up-conv) [19]. This operation can be performed in many ways. By creating an additional filter, where the layer is calculated or by diluting the image, describing each value from the input map feature with zeros. In our case, each value is duplicated (reproduced) from the size 1×1 to the size 2×2. The maps from the encoder are additionally copied to the decoding part, creating a single three-dimensional matrix where later convolution is carried out simultaneously on them and the layers after up-pooling.

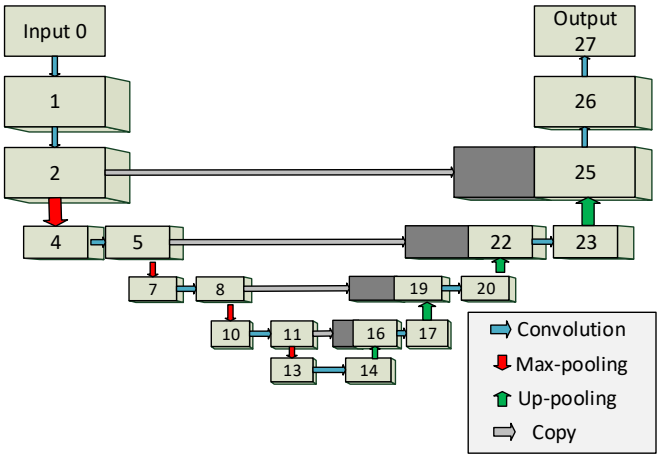

Fig. 1. Architecture of the U-Net network used in the paper.

The use of two-dimensional convolution when analyzing one-dimensional data during sequence discovery is reflected in the operation of convolution networks. When processing NLP data by convolution networks, networks using one-dimensional convolution [20] obtain usually the best effects. In our research, we want to bring out the repetitive shape created in the encoded bitmap. Unlike image analysis, the importance of the encoded information in each pixel is the same [21], and the size of the set of adjacent pixels cannot affect the result, i.e., repetition of multiple values cannot be assigned to a sequence. In our experiments, we accomplish this by creating a set with elements next to each other, where the value of these elements is to be omitted. Another aspect is why the CNN network for image segmentation without a skip-connection (for example U-Net) like [22], or [23] could not solve the task. Although both structures (with and without skip-connections) are suitable for detecting changes in the image, the CNN network was unable to detect sequence elements during the training. We base our assumptions why this was not possible on the fact that the signal during learning was not well propagated back. The U-Net structure has additional skip connections, thanks to which the error signal in such a complicated bitmap can be better propagated to the input layer. Another problem is the analysis of the filters obtained after training the network. It is necessary to analyze whether the set of filters in a given layer is not identical for each feature maps, which would entail unnecessary network load and poor optimization of the entire structure of the convolution network.

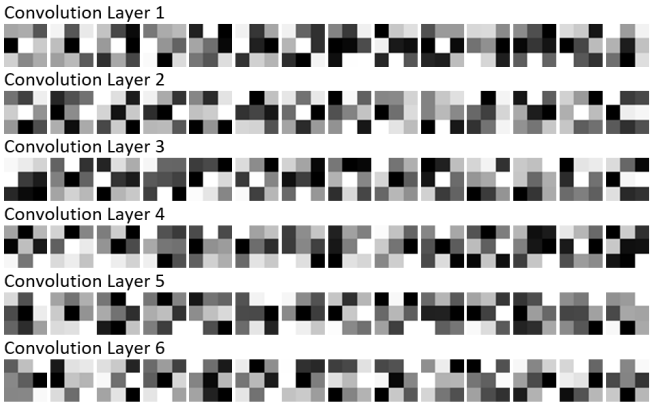

Fig. 2. Visualization of first 12 filters in six convolutional layers of our network trained on sequences.

The size of the input and output two-dimensional matrices must be specified when designing the U-Net architecture. In our architecture, the input and output have the same size $n \times m$, where n is the number of encoded characters (the dictionary size), m is the maximal number of elements in the analyzed data (the window that searches for sequences). As with black-and-white images, we only use one input channel. Our goal is to obtain only numbers that form a repetitive sequence at the U-Net output. A zero-padding operation was used in the network so that the size of the network was not modified after

the convolution [24]. In the experiments we use the following U-Net structure:

1) Conv. padding(3x3), in(1x128x256), out(32x128x256), ReLu activ.
2) Conv. padding(3x3), in(1x128x256), out(32x128x256), ReLu activ.
3) MaxPooling (2x2)
4) Conv. padding(3x3), in(32x64x128), out(64x64x128), ReLu activ.
5) Conv. padding(3x3), in(64x64x128), out(64x64x128), ReLu activ.
6) MaxPooling (2x2)
7) Conv. padding(3x3), in(64x64x128), out(128x32x64), ReLu activ.
8) Conv. padding(3x3), in(128x32x64), out(128x32x64), ReLu activ.
9) MaxPooling (2x2)
10) Conv. padding(3x3), in(128x16x32), out(256x16x32), ReLu activ.
11) Conv. padding(3x3), in(256x16x32), out(256x16x32), ReLu activ.
12) MaxPooling (2x2)
13) Conv. padding(3x3), in(256x8x16), out(512x8x16), ReLu activ.
14) Conv. padding(3x3), in(512x8x16), out(512x8x16), ReLu activ.
15) UpSampling (2x2)
16) Conv. padding(3x3), in(512x16x32[15]+256x16x32[11]), out(256x16x32), ReLu activ.
17) Conv. padding(3x3), in(256x16x32), out(256x16x32), ReLu activ.
18) UpSamplign (2x2)
19) Conv. padding(3x3), in(256x32x64[18]+128x32x64[8]), out(128x32x64), ReLu activ.
20) Conv. padding(3x3), in(128x32x64), out(128x32x64), ReLu activ.
21) UpSamplign (2x2)
22) Conv. padding(3x3), in(128x64x128[20] + 64x64x128[5]), out(64x64x128), ReLu activ.
23) Conv. padding(3x3), in(64x64x128), out(64x64x128), ReLu activ.
24) UpSampling (2x2)
25) Conv. padding(3x3), in(128x128x256[20] + 32x128x256[2]), out(64x128x256), ReLu activ.
26) Conv. padding(3x3), in(64x128x256), out(64x128x256), ReLu activ.
27) Conv. padding(1x1), in(64x128x256), out(1x128x256), ReLu activ.

III. Frequent Sequences Data

The data set for supervised neural networks must consist of training and testing data. In the case of sequential data mining, the datasets focus primarily on checking the efficiency of the algorithm, and it is difficult to find a set that could be used for neural network purposes. Our aim was that the neural network is to recognize sequences that have not participated in the training process. In other words, it must grasp a general idea of repetitive sequences instead of memorizing sequences existing in the data set. Furthermore, every type of sequential data can be transformed into numbers from a dictionary. Thus, training with extensive synthetic data seems to be a perfect solution.

We developed a generator for sequence datasets. Each item was a number from 1 to n, where n is the maximum value we can encode in a given U-Net network. In other words, n is the aforementioned dictionary size. One training example consists of smaller transactions in which a repetitive sequence of characters (green color in Fig. 3) is hidden between them; other numbers were drawn randomly from numbers not used to create the sequence. After each transaction, there is a break created from an additional number of random characters that also did not participate in the creation of the sequence. An example of the sequence is presented in Figure 3.

Fig. 3. Data sequence example generated by the generator.

The parameters for creating samples are:

- The size of the convolutional network input n (dictionary size),
- The maximal space between sequence elements (blue parts in Fig. 3),
- The maximal and minimal length of the sequence itself (numbers in green in Fig. 3),
- The maximal interval between sequences (yellow fields).

Using this type of generator has several key advantages in the case of neural networks. In the paper, we use a one-hot number-level signal encoding. The input signal to the convolutional network (and the output) is a two-dimensional matrix. In algorithms such as GSP or SPADE, we have no limit to the number of elements that a sequence can consist of. In the case of the proposed method, we have to assume it earlier.

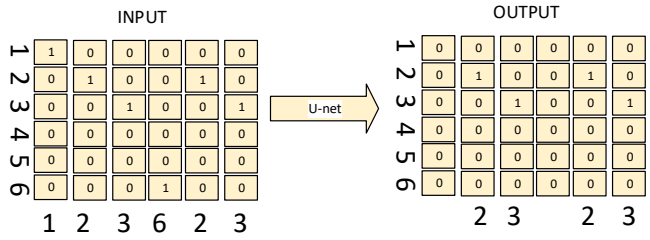

Fig. 4. Data sequence example after one-hot encoding, prepared for the supervised training for the dictionary size $n = 6$. We train the network with sequences intertwined with random noise and the desired output is the clean sequence. We do not use every possible sequence combination. Instead, we train a specific behaviour that allows discovering unknown sequences after training.

The proposed method discover only closed sequential patterns in sequence databases. Standard algorithms, such as BIDE+ has a user-specified threshold named minimum support (a value in [0,1]). In the proposed method, we cannot set this parameter, and the only possibility is to prepare suitable training data.

IV. Experiments

In this section, we present experiments showing the performance of the method and the ability to recognize transposed sequences. We had to prepare datasets with input data and the desired output with the sequence separated from the surrounding noise. Such sequences will be shown at the neural network output when a sequence is identified in the data. An example of such a training pair is presented in Fig. 4. The neural network CNTK script and the data generator is available at https://github.com/DiscoveringSequentialPatterns/.

A. Training

We trained U-Net with the backpropagation algorithm with the Adam optimizer [25], which sets the learning rate in an adaptive manner. That helps to leave training error local minima. Generally, the Adam optimizer is a combination of several earlier deep learning optimizers and allows achieving excellent results in a broad area of tasks. The momentum term was set to 0.9 [26]. The learning rate was set at 1×10^{-4} during the first 2-4 epochs. Then, it remained constant at 1×10^{-5}. Establishing the loss function was an essential element in the case of the U-Net network. We applied the Dice-Sørensen coefficient (DSC) [27], [28] which compares pixel-wise a desired and the actual output image of the sequence (1 means perfect match, 0 vice-versa). It defines the similarity of two samples

$$DSC = \frac{2\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}Y_{ij}}{1+\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}+\sum_{i=1}^{n}\sum_{j=1}^{m} Y_{ij}}, \qquad (1)$$

where X is a training pattern, Y is the output from the U-Net network. Of course, unlike in the case of real-life images, sound, etc., in the paper, during the training, we did not use any data augmentation; the input data will always have the same configuration. Obviously, we have to provide the sequence examples as they are, without any modification. Moreover, as the sequential mining data are very peculiar compared to images, sound, etc., we checked the impact of the minibatch value to the training. We found out that the best results were obtained with minibatch = 1, see Figure 5.

B. Experiment 1

In this experiment, we used the U-Net network with 128×256 character input size ($n = 128$ and $m = 256$). We generated 196×10^4 different random training sequences with injected random numbers (not from a sequence itself) and a dictionary of 128. The size of the search sequences was in the range of [5,10] characters, and there were several sequences in each training input-output matrices (black and white images). For the analysis of the selected algorithms, we used the SPMF

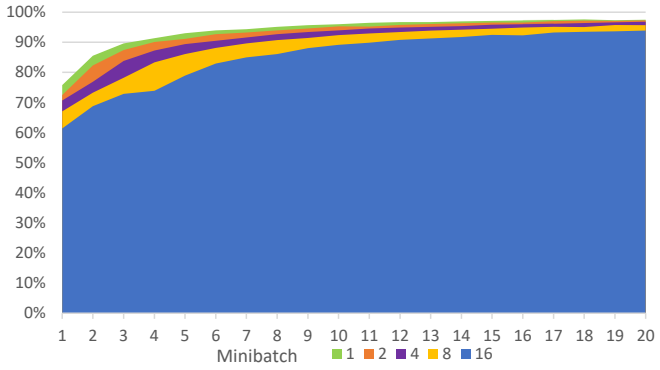

Fig. 5. Accuracy and training time (epochs) for various values of minibatch.

TABLE I
SPEED COMPARISON OF THE PROPOSED METHOD TO THE STANDARD SEQUENCE MINING ALGORITHMS. SPADE WAS IMPLEMENTED IN THE PARALLELIZED VERSION. IN THE CASE OF U-NET, MIN_SUPPORT WAS NOT TAKEN INTO ACCOUNT. THE MOST IMPORTANT IS THE COMPARISON WITH BIDE+ AS BOTH METHODS FIND CLOSED SEQUENCES.

min support	GSP time[ms]	SPADE time[ms]	BIDE+ time[ms]	U-Net time[ms]
0.6	9712	1118	7	364.8
0.3	9802	1119	8	364.8
0.2	10465	1169	8	364.8
0.1	9817	1169	9	364.8
0.05	9730	1272	23	364.8
0.01	10967	1299	204	364.8
0.005	13023	1431	238	364.8
0.003	17866	1159	365	364.8
0.002		2126	787	364.8
0.0005		3093	901	364.8

open-source data mining library [29]; it requires that each transaction is separated by number -2. This element has been added after each last element belonging to the sequence. In the case of the U-Net network used in this experiment, we could encode up to 256 characters in one input-output image (as described earlier, the applied encoding scheme resulted in data similar to two-dimensional images). The first step was to check whether each of the sequences was correctly found by the algorithms and the U-Net network. In the experiment, all the algorithms returned the intended sequences. Then, we could proceed to check the neural network performance against the standard algorithms. We studied the impact of the min_support parameter on the performance of the algorithms on the tested hardware. In the case of the neural network, it is not possible to provide this parameter; therefore, the processing time is constant for each case. We can observe that the network detects the sequence in each case faster than the GSP and SPADE algorithms. However, in the case of the BIDE+ algorithm [12], after changing the min_support parameter to 0.003 (discovering longer sequences) it works slower than the U-Net network, what means that it is faster only for very short sequences. For longer sequences, the U-Net sequence discovery is faster, and the speed gain increases with the sequence length.

TABLE II
EXPERIMENT WITH FINDING TRANSPOSED SEQUENCES BY TWO U-NET NETWORKS OF DIFFERENT SIZES

	U-Net 1	U-Net 2
Number of elements (dictionary)	128	256
Max. number of elements in a seq.	256	512
Percent of elements found correctly	94.8%	95.5%
Percent of elements found	99.7%	99.8%
Time	3.8 ms	19.5 ms

TABLE III
EXAMPLE OF FINDING SEQUENCES TRANSPOSED FROM THE TRAINING SEQUENCES

Network output for the transposed seq.	35 60 83 30 16 60 16 30 35 83 16 30 35 60 35 60 16 30 16 35 60 30 30 16 60 35 83 30 60 35 16 30 35 16 60 60 16 35 30 60 35 16 30 60 30 83 16 35 83
Desired network output or the transposed seq.	35 60 83 30 16 60 16 30 35 83 16 30 35 60 83 35 60 16 30 83 83 16 35 60 30 83 30 16 60 35 83 30 60 35 16 83 30 35 16 60 60 16 83 35 30 60 35 16 30 83 60 30 83 16 35 35 16 60 83
The same sequence but not transposed on the network output	35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 30 35 16 83 30 35 16 60 83
Desired network output for not transposed seq.	35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83 30 35 16 60 83

C. Experiment 2

The next experiment was to check how the trained network behaves when the sequence elements are recorded in the analyzed transactions in a different order. In other words, we test the trained network with transposed sequences. Example testing string: 1 9 2 4 3, 2 7 1 6 3, 3 5 1 8 2, and the searched items are: 1 2 3. The experiment was carried out on two network architectures. Network 1 was used in the previous experiment with the previously described architecture — Network 2 with the same layout of layers with the increased input size. Increasing the input size is reflected in the entire architecture by increasing the size of all feature maps by two. We generated data for each U-Net network according to its architecture, with 5×10^5 test strings containing a hidden sequence. The data for the Network 1 contained 128 characters in the dictionary, and the sequence length was 256. Network 2 analyzed data from the 256-character dictionary, and the data had length 512. In the case of the data we generated, we knew what elements we were looking for; thus, we could determine what percentage of the transposed elements was found. In this respect, each network was able to find over 99 % of the transposed items. Both Networks 1 and 2 can perfectly assess which elements belong to the sequence. The significant difference is the network processing speed. Network 2 needs a lot more time to calculate one example, but it gains 1 % on the accuracy in the transposed sequence detection.

V. CONCLUSION

We presented a method for discovering frequent closed sequences in data. The method uses the U-Net architecture trained with the desired sequences for given input data. We checked various types of convolutional neural networks, and only U-Net was able to detect sequences. We used one-hot number-level encoding for the input data and the output (found) sequences. We also tried to use one-dimensional encoding without success. The trained network detects sequences unseen during training, and even consisting of a new set of numbers. The method can efficiently analyze large datasets. Moreover, the workload can be distributed into many GPU devices. The method is less efficient than standard algorithms (e.g., GSP, SPADE) only in the case of very short sequences, usually impractical in real-life applications. Generally, the proposed method is more efficient for real-life usage as it analyses a long window at once. It should be mentioned that the paper concerns finding any sequence, not matching patterns from the training data sets. The disadvantage of the method is the necessity of declaring the dictionary size and the size of the window that analyses data. In the current version of the system, we cannot determine the minimum support or time constraints.

REFERENCES

[1] E. Rafajłowicz, M. Wnuk, and W. Rafajłowicz, "Local detection of defects from image sequences," *International Journal of Applied Mathematics and Computer Science*, vol. 18, no. 4, pp. 581–592, 2008.

[2] P. Jurewicz, W. Rafajłowicz, J. Reiner, and E. Rafajłowicz, "Simulations for tuning a laser power control system of the cladding process," in *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2016, pp. 218–229.

[3] E. Rafajłowicz and W. Rafajłowicz, "Testing (non-) linearity of distributed-parameter systems from a video sequence," *Asian Journal of Control*, vol. 12, no. 2, pp. 146–158, 2010.

[4] W. Wei, X. Fan, M. Woźniak, H. Song, W. Li, Y. Li, and P. Shen, "H∞ control of network control system for singular plant," *Information Technology And Control*, vol. 47, no. 1, pp. 140–150, 2018.

[5] S. A. Ludwig, "Applying a neural network ensemble to intrusion detection," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 3, p. 177–188, 2019.

[6] W. Wei and Y. Qi, "Information potential fields navigation in wireless ad-hoc sensor networks," *Sensors*, vol. 11, no. 5, pp. 4794–4807, 2011.

[7] W. Wei, H. Song, H. Wang, and X. Fan, "Research and simulation of queue management algorithms in ad hoc networks under ddos attack," *IEEE Access*, vol. 5, pp. 27 810–27 817, 2017.

[8] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*. IEEE, 1995, pp. 3–14.

[9] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *International Conference on Extending Database Technology*. Springer, 1996, pp. 1–17.

[10] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast vertical mining of sequential patterns using co-occurrence information," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014, pp. 40–52.

[11] M. Verma and D. Mehtac, "Sequential pattern mining: A comparison between gsp, spade and prefix span," *International Journal of Engineering Development and Research*, vol. 2, no. 3, pp. 3016–3036, 2014.

[12] J. Wang and J. Han, "Bide: Efficient mining of frequent closed sequences," in *Data Engineering, 2004. Proceedings. 20th International Conference on*. IEEE, 2004, pp. 79–90.

[13] J. Wang, J. Han, and C. Li, "Frequent closed sequence mining without candidate maintenance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1042–1056, Aug 2007.

[14] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4739–4745.

[15] M. Landauer, M. Wurzenberger, F. Skopik, G. Settanni, and P. Filzmoser, "Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection," *Computers & Security*, vol. 79, pp. 94 – 116, 2018.

[16] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, p. 235–245, 2019.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[19] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Advances in Neural Information Processing Systems*, 2014, pp. 1790–1798.

[20] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[21] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 4898–4906.

[22] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. De Vries, M. J. Benders, and I. Išgum, "Automatic segmentation of mr brain images with a convolutional neural network," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1252–1261, 2016.

[23] J. Xu, X. Luo, G. Wang, H. Gilmore, and A. Madabhushi, "A deep convolutional neural network for segmenting and classifying epithelial and stromal regions in histopathological images," *Neurocomputing*, vol. 191, pp. 214–223, 2016.

[24] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in neural information processing systems*, 2014, pp. 2042–2050.

[25] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *the 3rd International Conference for Learning Representations, San Diego*, vol. 5, 2015.

[26] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[27] Z. Zhu, Y. Xia, W. Shen, E. Fishman, and A. Yuille, "A 3d coarse-to-fine framework for volumetric medical image segmentation," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 682–690.

[28] M. Amrehn, S. Gaube, M. Unberath, F. Schebesch, T. Horz, M. Strumia, S. Steidl, M. Kowarschik, and A. Maier, "Ui-net: Interactive artificial neural networks for iterative image segmentation based on a user model," *arXiv preprint arXiv:1709.03450*, 2017.

[29] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The spmf open-source data mining library version 2," in *Machine Learning and Knowledge Discovery in Databases*, B. Berendt, B. Bringmann, É. Fromont, G. Garriga, P. Miettinen, N. Tatti, and V. Tresp, Eds. Cham: Springer International Publishing, 2016, pp. 36–40.