Process Trace Classification for Stroke Management Quality Assessment

Giorgio Leonardi, Stefania Montani(✉), and Manuel Striani

DISIT, Computer Science Institute, Università del Piemonte Orientale, Alessandria, Italy
stefania.montani@uniupo.it

Abstract. Stroke is a medical condition where poor blood flow to the brain may result in cell damage, possibly leading to patient's death or disability. Acute stroke care is best performed in dedicated and well-organized centers. Medical process trace classification can support stroke management quality assessment, since it allows to verify whether better-equipped Stroke Centers actually implement more complete processes, suitable to manage complex patients as well. In our previous work, we developed a semantic similarity metric able to compare process traces. In this paper, we adopt such a metric to perform k-Nearest Neighbour (k-NN) classification in the field of stroke management; moreover, we present an alternative classification approach based on deep learning techniques. Experimental results have shown the feasibility of deep learning classification for stroke management quality assessment, which performed better than the application of the semantic similarity metric. Improvements and future research in this direction will therefore be considered. Difficulties in classifying patients treated in less-equipped hospitals also suggest to identify and manage possible organizational problems.

Keywords: K-NN classification · Deep learning · Process traces

1 Introduction

A stroke is a medical condition where poor blood flow to the brain can result in cell death. Approximately 1.1 million inhabitants of Europe suffer a stroke each year and, because of the aging population, the absolute number of stroke is expected to dramatically increase in the near future: by 2025, 1.5 million European people will suffer a stroke each year [5].

Acute stroke care in hospitals is best performed in organized Stroke Units, where patient outcomes are better than those of patients managed in general medical or neurological wards [13].

The European Stroke Organisation (ESO) Stroke Unit Certification Committee has worked on the definition of evidence-based needs for acute stroke care, in order to stimulate the certification of more advanced stroke care facilities. The Committee has thus established 2 certification levels: (1) ESO Stroke Units (SUs) and (2) ESO Stroke Centers (SCs) [28]. ESO Stroke Centers must meet

I. Watson and R. Weber (Eds.): ICCBR 2020, LNAI 12311, pp. 49–63, 2020.
https://doi.org/10.1007/978-3-030-58342-2_4

all the requirements of an ESO Stroke Unit, and additionally should provide more advanced diagnostic and therapeutic equipment, have a larger staff and have expertise on rare or complex stroke subtypes.

In Italy, the Ministry of Health has codified the two levels of stroke care in 2015, along the lines explained above. However, significant organizational problems are still observed not only in SUs, but sometimes also in SCs[1]. Therefore, a thorough analysis of medical processes is needed, in order to verify if the actual performance of an hospital is coherent with its declared level.

In this paper, we propose to tackle the above needs by considering stroke management process traces (i.e., the sequences of activities actually executed on the single patients at the hospital at hand, and logged in the hospital information system). Traces can be interpreted as *cases* [1]; the identification of the k Nearest-Neighbour (k-NN) cases and k-NN classification (distinguishing between the SU class and the SC class) can then be implemented, to verify if the logged activities are coherent with the level assigned to a given hospital, in a quality assessment perspective.

In particular, we have realized classification according to two different approaches:

- in the first approach, we have adopted a trace similarity metric, able to take into account temporal information as well as domain knowledge, that we published in recent years [22,23]. By exploiting this metric, we have implemented k-NN trace classification;
- in the second approach, we have adopted a deep learning strategy [17]. Specifically, we have resorted to an architecture based on Long Short Term Memory (LSTM) networks [11] to extract *deep* features from process traces, and to the Euclidean distance for k-NN classification in this feature space.

The first experimental results obtained by means of the metric in [22,23] were not very satisfactory. We obtained a good improvement by separating the SC class into two subclasses, in order to better distinguish between more complex and simpler patients. The analogous separation, however, did not provide an analogous amelioration in the SU class. Overall, we obtained much better results (in both experiments) by resorting to the deep learning strategy; indeed, deep learning techniques are being increasingly adopted and proving successful in process classification and prediction, as described in Sect. 2. Our first experiments suggest to further investigate in this direction in the future. Difficulties in classifying traces within the SU class (experienced with deep learning as well) also suggest to identify and manage possible organizational problems.

The paper is organized as follows: in Sect. 2 we summarize related work. In Sect. 3 we detail our approaches to stroke trace classification for quality assessment. In Sect. 4 we present experimental results, while Sect. 5 is devoted to discussion and conclusions.

[1] https://www.sanita24.ilsole24ore.com/art/medicina-e-ricerca/2017-04-14/stroke-unit-merce-rara-strutture-e-personale-dati-lontani-dm-702015-162809.php?uuid=AEEhud5.

2 Related Work

The management of processes and process traces is nowadays an established area of research within the Case Based Reasoning (CBR) community, as testified by the workshops on Process Oriented CBR (PO-CBR) which have been co-located with the International Conference on CBR, where the most recent one was held in 2019[2].

In particular, process trace comparison has been tackled in, e.g., [12], which introduces a distance definition able to combine a contribution related to activity similarity and a contribution related to delays between activities, and in [22,23], where activity similarity is dealt with in a semantic way (see also Sect. 3.1).

Trace comparison can be adopted to support process prediction/classification [6], a task which exploits the activities logged in process traces to make predictions about the future of a running trace (such as, e.g., the remaining time to complete the work, the next activity to be executed, the needed resources), or to classify the trace on the basis of some categorical or numerical performance properties (as in our work). Process prediction and classification can be useful both for a better planning of the needed resources, and for quality assessment, by means of the identification of non-compliances with respect of the expected performance.

In the literature, most works in this field are focused on the prediction of the next activity in a running process trace. While classical business process management approaches use an explicit model representation such as a state-transition model [16] or an Hidden Markov Model [14], a more recent research direction exploits deep learning.

In particular, several authors rely on Recurrent Neural Networks (RNNs) [27], and more specifically on Long-Short-Term Memory (LSTM) networks [11]. The idea in RNNs is to preserve the results of previous calculations with memories, i.e., with feedback connections that provide a parameter sharing across different parts of the model. In LSTM a cell state, more complex than the memory cell in basic RNNs, is introduced, where information can be added or removed by gated structures [11]; this solution reduces the training time. LSTM can potentially learn the complex dynamics within the temporal ordering of input sequences; therefore, they are well suited to manage the sequential data of process activity logs. Specifically, they can also manage long-distance dependencies between activities. Indeed, in LSTM networks a long-term memory can be implemented, where the information flows from cell to cell with minimal variations, keeping certain aspects constant during the processing of all inputs.

In [32], the authors use LSTM networks to predict the type of the next activity of an ongoing process trace and the time until the next activity (its timestamp). The network architecture consists of a shared LSTM layer that feeds two independent LSTM layers specialized in predicting the next activity and in predicting times, respectively. The experiments show that the LSTM approach outperforms model-based approaches. The work in [8] proposes a different

[2] https://iccbr2019.com/workshops/process-oriented-case-based-reasoning/.

network architecture which comprises two LSTM hidden layers. An empirical evaluation shows that this approach sometimes outperforms the approach of [32] at the task of predicting the next activity. In [7] the authors combine the approach in [8] with the idea of interleaving shared and specialized layers from [32] to design prediction architectures that can handle large numbers of activity types. The paper in [10], on the other hand, is more generally devoted to classification. In this work, RRNs are used in a system designed to solve any classification problem (including next activity prediction) based on activity sequences.

In [21] the authors propose to predict the next activity using a multi-stage deep learning approach. In this approach, each activity is first mapped to a feature vector. Next, transformations are applied to reduce the input dimensionality, by extracting n-grams and applying a hash function; then, the input is passed through two Autoencoder layers. The main idea behind Autoencoders is to reduce the input into a latent space with fewer dimensions and then try to reconstruct the input from this representation. By reducing the number of variables which represent the data, the model is forced to learn how to keep only meaningful information, from which the input is reconstructable. In [21], the transformed input is finally processed by a feed-forward Neural Network responsible for the next activity prediction.

A different approach [20] relies on Convolutional Neural Networks (CNNs) [3]. CNNs operate by exploiting multiple convolution operators: a convolution is an operation which takes a filter and multiplies it over the entire area of the input. Convolution layers are then followed by pooling layers, meant to further reduce dimensionality. In particular, in [20] the authors resort to the inception architecture. The inception architecture [31] uses kernels of varied size in a convolution layer to capture features at different levels of abstraction: it processes information at different scales and then aggregates them to efficiently extract relevant features. The authors have obtained better results in predicting the next activity with respect to LSTM architectures in their experiments.

Overall, our approach is thus inserted in a very active research panorama, which is recently focusing on promising deep learning solutions.

Interestingly, deep learning is being progressively considered in CBR research as well (see, e.g., [4,29]), even if - to the best of our knowledge - not yet for trace classification/prediction. Our work can therefore be seen as an innovative contribution in this field.

3 Medical Process Trace Classification

This section presents the technical details of our work.

In particular, Subsect. 3.1 summarizes the main characteristics of the metric, defined in [22,23], which we have used in this paper for stroke trace classification.

Subsect. 3.2 provides a description of the deep learning architecture we have tested as an alternative to this classical approach.

3.1 Classification Through Semantic Trace Comparison

As a first strategy to process trace classification, we have implemented a k-NN classification approach, resorting to the metric we described in [22,23], which is a semantic extension of the edit distance [19].

Indeed, every process trace is a sequence of activities, each one stored with its execution starting and ending times, and an activity is basically a symbol (plus the temporal information).

In the metric in [22,23], thus, we first take into account activity types, by calculating a modified edit distance which we have called **Trace Edit Distance** [22,23]. As the classical edit distance [19], Trace Edit Distance tests all possible combinations of editing operations that could transform one trace into the other one. However, if domain knowledge allows to organize activities in an ontology or a taxonomy, as we have done in the field of stroke (see Fig. 1), the cost of a *substitution* is not always set to 1: indeed, we can adopt a more **semantic** approach, and apply Palmer's distance [26], to impose that the closer two activities are in the semantic structure, the less penalty we introduce for substitution.

Trace Edit Distance $trace_{NGLD}(P, Q)$ is then calculated as the Normalized Generalized Levenshtein Distance (NGLD) [33] between two traces P and Q (interpreted as two strings of symbols). Formally, we provide the following definitions:

Definition 1: Trace Generalized Levenshtein Distance.
Let P and Q be two traces of activities, and let α and β be two activities. The Trace Generalized Levenshtein Distance $trace_{GLD}(P, Q)$ between P and Q is defined as:

$$trace_{GLD}(P, Q) = min\{\sum_{i=1}^{k} c(e_i)\}$$

where $(e_1, \ldots, e_k)$ transforms P into Q, and:

- $c(e_i) = 1$, if e_i is an activity insertion or deletion;
- $c(e_i) = dt(\alpha, \beta)$, if e_i is the substitution of α (appearing in P) with β (appearing in Q), with $dt(\alpha, \beta)$ being Palmer's distance [26] between the two substituted activities.

Definition 2: Trace Edit Distance (Trace Normalized Generalized Levenshtein Distance).
Let P and Q be two traces of activities, and let $trace_{GLD}(P, Q)$ be defined as in Definition 1 above. We define Trace Edit Distance $trace_{NGLD}(P, Q)$ between P and Q as:

$$trace_{NGLD}(P, Q) = \frac{2 * trace_{GLD}(P, Q)}{|P| + |Q| + trace_{GLD}(P, Q)}$$

where $|P|$ and $|Q|$ are the lengths (i.e., the number of activities) of P and Q respectively.

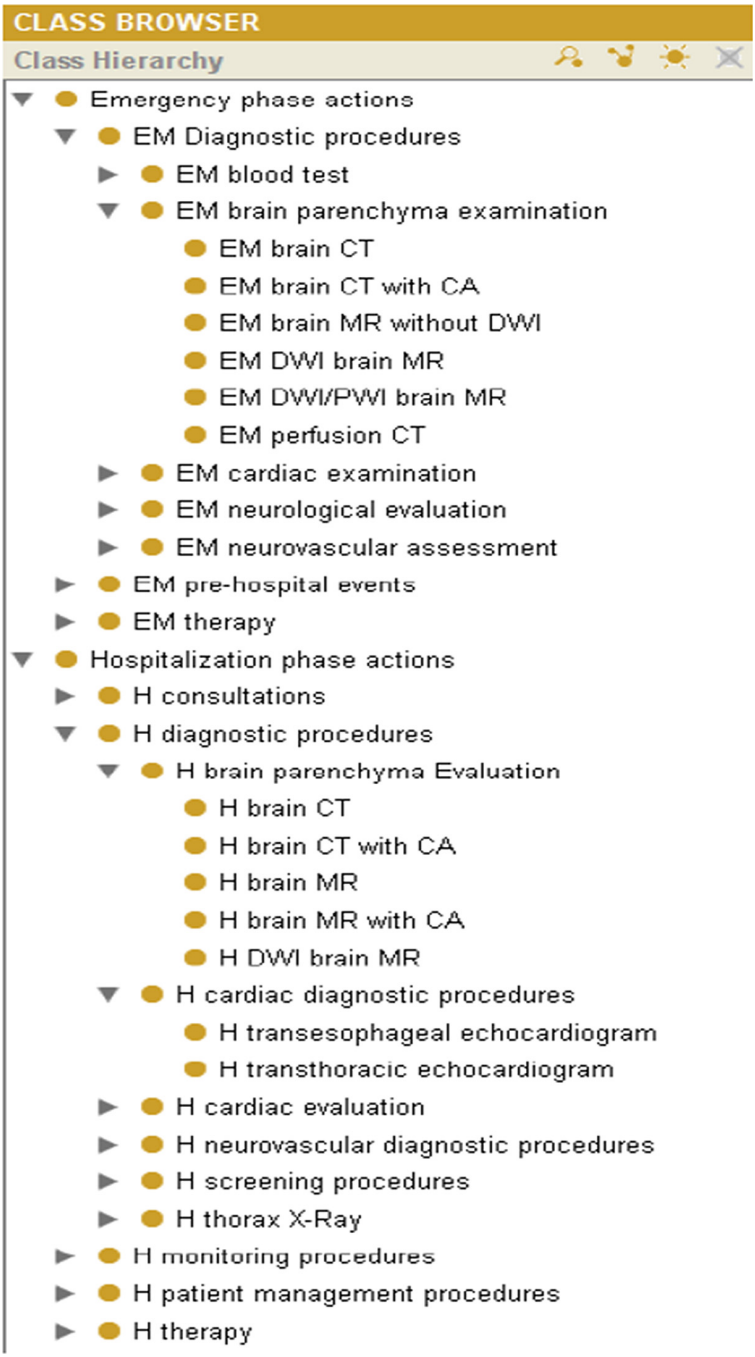

Fig. 1. An excerpt from the domain taxonomy.

The minimization of the sum of the editing costs allows one to find the optimal alignment between the two traces being compared. Given the optimal alignment, we can then take into account temporal information. Indeed, starting and ending times allow to get information about activity duration, as well as qualitative (e.g., Allen's *before*, *overlaps*, *equals* etc. [2]) and quantitative temporal constraints (e.g., delay length, overlap length [15]) between pairs of consecutive activities.

In particular, we compare the durations of aligned activities by means of a metric we called **Interval Distance** [22,23]. Interval distance calculates the normalized difference between the length of two intervals (representing activity durations in this case).

Moreover, we take into account the temporal constraints between two pairs of subsequent aligned activities on the traces being compared (e.g., activity A and B in trace P; the aligned activities A' and B' in trace Q). We quantify the distance between their qualitative constraints (e.g., A and B overlap in trace P; A' meets B' in trace Q), by resorting to a metric known as **Neighbors-graph Distance** [22,23]. If Neighbors-graph Distance is 0, because the two pairs of activities share the same qualitative constraint (e.g., A and B overlap in trace P; A' and B' also overlap in trace Q), we compare quantitative constraints by properly applying Interval Distance again (e.g., by calculating Interval Distance between the two overlap lengths).

In the metric in [22,23], these three contributions (i.e., Trace Edit Distance, Interval Distance between durations, Neighbors-graph Distance or Interval Distance between pairs of activities) are finally put in a linear combination with non-negative weights.

3.2 Deep Learning Classification

Inspired by existing literature contributions, we have tested a deep learning approach for stroke trace classification.

In particular, motivated by the successful examples described in Sect. 2 (see, e.g. [8,32]), we have defined and tested an LSTM-based architecture, which is described in Fig. 2.

In this approach, process traces are first pre-processed by converting each activity into a integer by means of an hashing layer; the overall trace is therefore converted into a feature vector. The architecture then exhibits two LSTM block, composed of 32 and 16 units (respectively) with *tanh* activation function and followed by a dropout layer, which randomly forces a fraction of the input units to be ignored at each update during training time, to help prevent overfitting [30].

The *deep* features produced by the following fully connected layer with *Relu* activation function can then be provided as an input to a k-NN classifier. Specifically, we resorted to the open source tool Weka [9] and to the Euclidean distance to perform k-NN classification in this feature space.

All parameters were set experimentally.

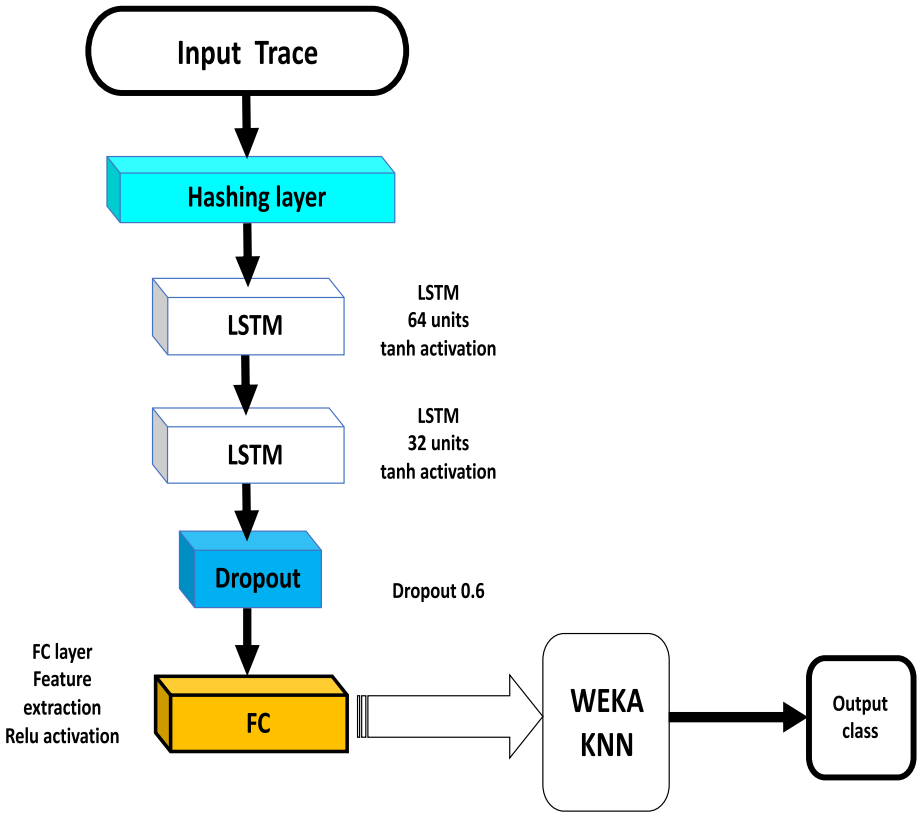

Fig. 2. LSTM-based architecture

4 Experimental Results

Our dataset was comprised of 5013 process traces, composed by a number of activities ranging from 10 to 25 (16 on average). In particular, 2629 traces were generated in a SC, while 2384 were generated in a SU.

The deep learning approach was realized and tested by means of the tool TensorFlow[3].

Details of the results are presented in the following subsections.

[3] https://www.tensorflow.org/.

4.1 Semantic Trace Comparison: Classification Results

In our experiments on classification relying on semantic trace comparison, we conducted a 9-NN classification (k = 9 was the optimal parameter setting automatically calculated by Weka [9]. Anyway, we also conducted a sensitivity analysis, which demonstrated that results did not change significantly when changing the value of k). Results are shown in Tables 1 and 2.

Table 1. Results (I) obtained by K-NN classification with semantic trace comparison, by class

Class	Precision	Recall	F-Measure	Specificity
SU	0.73	0.59	0.65	0.69
SC	0.54	0.69	0.60	0.59
Weighted average	0.63	0.64	0.63	0.64

Table 2. Results (II) obtained by K-NN classification with semantic trace comparison

MCC	K-stat	Accuracy
0.27	0.27	0.63

Table 3 also reports the confusion matrix for the LSTM-based classifier, for the sake of completeness.

Table 3. Confusion matrix obtained by K-NN classification with semantic trace comparison

	SU	SC
SU	1745	639
SC	1213	1416

As it can be observed from the tables, results are quite poor, reinforcing the need to test different classification strategies.

Following the suggestion of medical experts, we made a second experiment. In this case, we separated the SC class into two subclasses, in order to distinguish between traces generated on particularly complex patients, and traces generated on simpler patients. Such a distinction was made by experts referring to: (i) clinical data and patient's characteristics, available in the hospital information system (such as, e.g., the presence of co-morbidities), and (ii) the presence of specific activities in the trace (such as procedures for managing uncommon and

problematic stroke types) or of repeated diagnostic/monitoring steps (such as frequent Computer Assisted Tomographies, to monitor the evolution over time of a particularly critical situation). Classification accuracy improved significantly within the SC traces, as shown in Tables 4 and 5.

Table 4. Results (I) obtained by K-NN classification with semantic trace comparison within the SC patients, by class

Class	Precision	Recall	F-Measure	Specificity
SC complex (905 traces)	0.59	0.72	0.65	0.80
SC simple (1724 traces)	0.88	0.80	0.84	0.72
Weighted average	0.78	0.78	0.77	0.75

Table 5. Results (II) obtained by K-NN classification with semantic trace comparison within the SC patients

MCC	K-stat	Accuracy
0.50	0.49	0.78

On the other hand, when implementing the same distinction within the SU class, we did not obtain significantly better results with respect to the initial experiment. Some discussion can be found in Sect. 5.

4.2 Deep Learning: Classification Results

Tables 6 and 7 report the results obtained by the tool Weka in 5-NN classification (in this case, $k = 5$ was the optimal parameter setting automatically calculated by Weka [9]), when *deep* features were provided by the LSTM architecture depicted in Fig. 2. As it can be seen, with respect to the adoption of semantic trace comparison, this approach provided a better classification performance.

Table 6. Results (I) obtained by LSTM + K-NN classification, by class

Class	Precision	Recall	F-Measure	Specificity
SU	0.73	0.72	0.73	0.76
SC	0.75	0.76	0.75	0.72
Weighted average	0.74	0.74	0.74	0.74

Table 8 also reports the confusion matrix for the LSTM-based classifier, for the sake of completeness.

Table 7. Results (II) obtained by LSTM + K-NN classification

MCC	K-stat	Accuracy
0.48	0.48	0.74

Table 8. Confusion matrix obtained by the LSTM + K-NN classification

	SU	SC
SU	1713	671
SC	620	2009

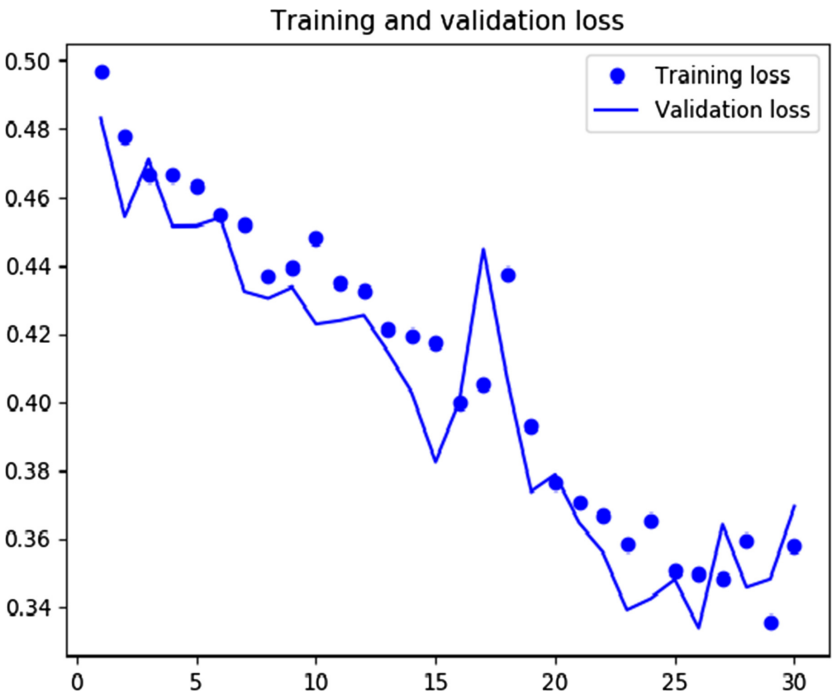

Fig. 3. LSTM loss per epoch

Figure 3 shows the evolution of the loss, depending on the number of epochs.

As it can be observed from the figure, experimentally, working at 30 epochs represented a good compromise, able to reduce the loss value without increasing too much the computational effort (15 min were required for computation at 100 epochs, on Intel Xeon E3 - 2.70 GHz 4 processors with 4 GB RAM).

Also in this case, we repeated the experiment by distinguishing between more complex patients and simpler patients. The deep learning strategy provided a

Table 9. Results (I) obtained by LSTM + K-NN classification within the SC patients, by class

Class	Precision	Recall	F-Measure	Specificity
SC complex (905 traces)	0.90	0.90	0.90	0.95
SC simple (1724 traces)	0.95	0.95	0.95	0.90
Weighted average	0.93	0.93	0.93	0.92

Table 10. Results (II) obtained by LSTM + K-NN classification within the SC patients

MCC	K-stat	Accuracy
0.85	0.85	0.93

much higher accuracy when working within the SC class patients (see Tables 9 and 10), while results did not improve much within the SU class patients.

5 Discussion and Conclusions

In this paper, we have proposed two very different approaches to stroke trace classification. The first approach relies on a semantic similarity metric, followed by k-NN classification. The second approach adopts deep learning techniques.

Our first experimental results have shown that the more traditional approach, based on the semantic similarity metric, is not very successful, while the deep learning strategy has performed better.

The rather poor results obtained by the semantic metric have improved significantly when focusing on the SC class, and distinguishing between more complex patients and simpler patients, suggesting that two types of processes are actually carried out, depending on the patient condition - which makes sense from the medical viewpoint; however, an analogous improvement was not observed when working within the SU traces. Interestingly, an analogous output was observed also when applying deep learning. We thus make the hypothesis that SUs are much more heterogeneous than SCs, and more affected by organizational problems, which may limit their capacity to apply the right protocol to the right patient. Further experiments will be needed to support this claim.

As a more general consideration on our experimental results, we can conclude that deep learning, which is nowadays frequently chosen by the business process management community as a tool for trace prediction and classification, is indeed a promising approach, to be further investigated in the future. To this end, we will make other experiments, and consider different architectures as well, such as, e.g., convolutional inception modules [20,31].

Since deep learning methods operate as black boxes, and it can difficult to provide a meaning for the abstracted *deep* features, or to justify misclassification, we will also consider the issue of explainability. To this end, we will investigate whether it is possible to adapt the knowledge-based strategy we adopted in [18].

Last but not least, we also believe that further improvements of classification results, by both the approaches evaluated in this paper, might be obtained by resorting to a trace abstraction technique, such as the one described in [24,25]. Such an approach can hide irrelevant details, that could lead to misclassification, while keeping the most important information in the trace. This research direction will be considered in our future research as well.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations and systems approaches. AI Commun. **7**, 39–59 (1994)
2. Allen, J.F.: Towards a general theory of action and time. Artif. Intell. **23**, 123–154 (1984)
3. Alom, M.Z., et al.: A state-of-the-art survey on deep learning theory and architectures. Electronics **8**(3), 292 (2019)
4. Amin, K., Kapetanakis, S., Althoff, K.-D., Dengel, A., Petridis, M.: Answering with cases: a CBR approach to deep learning. In: Cox, M.T., Funk, P., Begum, S. (eds.) ICCBR 2018. LNCS (LNAI), vol. 11156, pp. 15–27. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01081-2_2
5. Bejot, Y., Bailly, H., Durier, J., Giroud, M.: Epidemiology of stroke in Europe and trends for the 21st century. La Presse Medicale **45**(12, Part 2), e391–e398 (2016). QMR Stroke
6. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Quart. **40**, 1009–1034 (2016)
7. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_19
8. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**, 129–140 (2017)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18 (2009)
10. Hinkka, M., Lehto, T., Heljanko, K., Jung, A.: Classifying process instances using recurrent neural networks. In: Daniel, F., Sheng, Q.Z., Motahari, H. (eds.) BPM 2018. LNBIP, vol. 342, pp. 313–324. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11641-5_25
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Kapetanakis, S., Petridis, M., Knight, B., Ma, J., Bacon, L.: A case based reasoning approach for the monitoring of business workflows. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS (LNAI), vol. 6176, pp. 390–405. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14274-1_29
13. Kjellstrom, T., Norrving, B., Shatchkute, A.: Helsingborg declaration 2006 on European stroke strategies. Cerebrovasc. Dis. **23**, 229–241 (2007)
14. Lakshmanan, G.T., Shamsi, D., Doganata, Y.N., Unuvar, M., Khalaf, R.: Markov prediction model for data-driven semi-structured business processes. Knowl. Inf. Syst. **42**(1), 97–126 (2015)
15. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Proceedings of BMMDS/EMMSAD, pp. 94–107 (2010)

16. Le, M., Gabrys, B., Nauck, D.: A hybrid model for business process event prediction. In: Bramer, M., Petridis, M. (eds.) SGAI 2012, pp. 179–192. Springer, London (2012). https://doi.org/10.1007/978-1-4471-4739-8_13
17. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**(7553), 436–444 (2015)
18. Leonardi, G., Montani, S., Striani, M.: Deep feature extraction for representing and classifying time series cases: towards an interpretable approach in haemodialysis. In: Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference, FLAIRS 2020, Miami, Florida, AAAI Press (2020)
19. Levenshtein, A.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. **10**, 707–710 (1966)
20. Di Mauro, N., Appice, A., Basile, T.M.A.: Activity prediction of business process instances with inception CNN models. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019. LNCS (LNAI), vol. 11946, pp. 348–361. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35166-3_25
21. Mehdiyev, N., Evermann, J., Fettke, P.: A multi-stage deep learning approach for business process event prediction. In: Loucopoulos, P., Manolopoulos, Y., Pastor, O., Theodoulidis, B., Zdravkovic, J., (eds.) 19th IEEE Conference on Business Informatics, CBI 2017, Thessaloniki, Greece, 24–27 July ,2017, Volume 1: Conference Papers, pp. 119–128. IEEE Computer Society (2017)
22. Montani, S., Leonardi, G.: Retrieval and clustering for business process monitoring: results and improvements. In: Agudo, B.D., Watson, I. (eds.) ICCBR 2012. LNCS (LNAI), vol. 7466, pp. 269–283. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32986-9_21
23. Montani, S., Leonardi, G.: Retrieval and clustering for supporting business process adjustment and analysis. Inf. Syst. **40**, 128–141 (2014)
24. Montani, S., Leonardi, G., Striani, M., Quaglini, S., Cavallini, A.: Multi-level abstraction for trace comparison and process discovery. Expert Syst. Appl. **81**, 398–409 (2017)
25. Montani, S., Striani, M., Quaglini, S., Cavallini, A., Leonardi, G.: Semantic trace comparison at multiple levels of abstraction. In: Aha, D.W., Lieber, J. (eds.) ICCBR 2017. LNCS (LNAI), vol. 10339, pp. 212–226. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61030-6_15
26. Palmer, M., Wu, Z.: Verb semantics for english-Chinese translation. Mach. Transl. **10**, 59–92 (1995)
27. Pascanu, R., Gülçehre, Ç., Cho, K., Bengio, Y.: How to construct deep recurrent neural networks. In: Bengio, Y., LeCun, Y., (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014, Conference Track Proceedings (2014)
28. Ringelstein, E.B., et al.: European stroke organisation recommendations to establish a stroke unit and stroke center. Stroke **44**(3), 828–840 (2013)
29. Sani, S., Wiratunga, N., Massie, S., Cooper, K.: kNN sampling for personalised human activity recognition. In: Aha, D.W., Lieber, J. (eds.) ICCBR 2017. LNCS (LNAI), vol. 10339, pp. 330–344. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61030-6_23
30. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
31. Szegedy, C., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June 2015, pp. 1–9. IEEE Computer Society (2015)

32. Tax, N., Teinemaa, I., van Zelst, S.J.: An interdisciplinary comparison of sequence modeling methods for next-element prediction. *CoRR*, abs/1811.00062 (2018)
33. Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1091–1095 (2007)