Enhancing Deep Learning with Semantics: an application to manufacturing time series analysis

Xin Huang, Cecilia Zanni-Merk, Bruno Crémilleux

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Enhancing Deep Learning with Semantics: an application to manufacturing time series analysis

Xin Huang[a], Cecilia Zanni-Merk[a,*], Bruno Crémilleux[b]

[a] *Normandie Université, INSA Rouen, LITIS, France*
[b] *Normandie Université, UNICAEN, ENSICAEN, CNRS - UMR GREYC, France*

Abstract

Manufacturing enterprises are engaged in implementing new technologies to enhance their manufacturing lines in a smart way. These new technologies give manufacturing enterprises the knowledge, understanding, insight and foresight to improve products, processes and decisions, thereby creating a competitive advantage. In this paper, we explore the use of semantics to enhance deep learning models. We propose an ontology-based LSTM neural network, in which the deep architecture is designed with an ontology to extract high-level cognitive features and stacked LSTM layers for learning temporal dependencies. Our model is applied to a real manufacturing data set with multivariate time series for classification problems. The experiments show that our model can improve performance compared with conventional methods.

Keywords: Industry 4.0, Semantics, Deep Learning, Time series

1. Introduction

Industry 4.0 is a vision of current industrial development based on the Cyber-Physical Systems (CPSs) and includes a set of other techniques such as the Internet of things, cloud computing, and artificial intelligence. A significant characteristic of Industry 4.0 is the fusion of information technologies and manufacturing, which creates smart factories. CPSs are central to the smart factory and comprise intelligent machines, storage systems and production facilities capable of automatically exchanging information, trigging actions and controlling each other independently [1]. The tools, machines, workstations, and operators are interconnected, facilitating process traceability, adaptive and flexible control of the production equipment and

* Corresponding author. Tel.: +33-232956563
E-mail address: cecilia.zanni-merk@insa-rouen.fr

real-time response to uncertain situations. Manufacturing enterprises are engaged in implementing these new technologies to enhance their manufacturing lines in a smart way. Briefly, these manufacturing enterprises need to analyze the flow of data measured by the sensors in the CPSs and to understand their meaning, locally and collectively, for intelligent decision making.

Deep learning based methods often ignore the meaning and interrelationships among data and the internal activities of the manufacturing process cannot be well explained. The data collected during a manufacturing process, past successful experiences and industry standards need to be combined and capitalized as knowledge. This knowledge needs to be formally represented for exploitation, i.e. reusing and sharing knowledge of the manufacturing process. Semantic technologies [2] give the necessary tools to do so.

In this paper, we make the hypothesis that the joint use of semantics and deep learning will improve the quality of the interpretation of huge amounts of data and allow us to gain new insights. This hypothesis includes two aspects:

- **Deep learning for semantics**: Ontologies are widely used for knowledge representation and reasoning about semantic content in a structured way. However, manual ontology development is a hard and expensive task that usually requires knowledge of domain experts and skills of ontology engineers. Handcrafted ontologies are often inflexible and complex, which limits their usefulness and makes cross-domain alignment difficult. Recent advances in deep learning [3] have the potential to help mitigate these issues with automatic ontology development and alignment. Hohenecker and Lukasiewicz [4], for example, present a novel deep learning based reasoning approach rather than logic-based formal reasoning to ontology reasoning.
- **Semantics for deep learning**: Conversely, there are quite a few works on semantics for deep learning. Research in semantic data mining [5] has shown that the integration of semantic and data often presents better results on the tasks of data mining and deep learning. However, in most cases of application, formal knowledge representation is usually not well explored. Knowledge is generally encoded in a highly formal and abstract way in description logic. In most machine learning and deep learning algorithms, the input is always in the form of a numerical vector. It is often impractical to apply semantics directly to raw data. Previous works have shown that semantics is mostly applied to data pre-processing by enriching input vectors with ontological concepts and properties [6, 7], and post-processing by using distance measures with ontology semantics [8] rather than in the key stages of machine learning approaches which include model design and training. However, knowledge representation approaches such as ontologies have a similar "geometrical" structure to the ones of deep neural networks. Exploiting this analysis, we expect that a deep architecture designed with the assistance of an ontology can provide high-level cognitive features. Exploring semantics both in the learning and the interpretation process may enhance effectiveness and flexibility of the machine learning approach. However, few advances have been made from this perspective [9, 10].

In this paper, we explore a novel approach to enhance deep learning models by introducing semantics into a deep learning process. Specifically, we propose an ontology-based model *OntoLSTL*, in which a core manufacturing process ontology is used to design the deep neural networks that will be used for learning. Based on the instantiation of this core ontology (that includes generic concepts such as lines, machines, sensors ...) with information about a specific production line, our method infers the structure of the deep architecture that will be used to learn the features we are interested in. Moreover, by the use of this generic ontology, our method is comprehensive and can be applied to any manufacturing process.

The rest of this paper is organized as follows: Section 2 presents the concepts and technologies used in this paper. Section 3 and 4 deal with the design of deep architectures with semantics. Section 5 presents the experiments, evaluation metrics and results. Section 6 concludes the paper and discusses future work.

2. Related works and background

2.1. Semantic technologies for manufacturing process modeling

A manufacturing process is often very complex with multiple relations within its internal activities. Therefore, we should choose a formal framework such as ontologies for the representation of the manufacturing process. The term 'ontology' originated in a branch of philosophy which was subsequently included in the field of artificial intelligence for knowledge-based systems and adapted to the information retrieval problems. An ontology offers the concepts and their relations in a domain with a commonly agreed and formal expression such that it is possible to exchange semantic information among computers, and it has the reasoning capability for getting implicit information. An ontology of the manufacturing process refers to a universal model of the structure of manufacturing as an ordered wholeness. Several ontologies, such as PRONTO [11] or MASON[12], have been developed for general knowledge representation of whole product's life-cycle in industry. Some other ontologies, such as the work of Oltramari [13], have been developed for a specific manufacturing process that can be used in the production of sheet metal parts. Akmal and Batres [14] present a methodology for developing manufacturing process ontologies, incorporating a group of standards for categorizing classes of processes with formal analysis of the concept.

2.2. Deep Learning

The concept of deep learning stems from the study of artificial neural networks. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to many fields including computer vision or natural language processing. Deep learning combines low-level features to form more abstract high-level representation.

2.2.1. Recurrent Neural Networks (RNNs)

RNNs are applied to make use of sequential information. In traditional neural networks we assume that all inputs (and outputs) are independent of each other. But for many tasks, this assumption is not true. For example, if the idea is to predict the next word in a sentence, it is useful to know which words come before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output depending on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. Theoretically, RNNs can make use of information in arbitrarily long sequences, but in practice, they are limited to looking back only for a few steps. In addition, RNNs suffer of the so called *vanishing gradient problem* [15], meaning that new approaches are needed to learn long-term dependencies.

2.2.2. Long short-term memory networks (LSTMs)

Long Short-Term Memory networks (LSTMs) are a special kind of RNNs, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber [16], and were optimized and popularized by many works [17, 18]. LSTMs can prevent the vanishing and exploding gradient problems experienced in RNNs by employing multiple gates (input (I_G), output (O_G), and forget (F_G) gates) that enforce constant error flow through the internal states of special units called 'memory cells'.

2.3. Deep learning for time series analysis

Traditional work often depends on hand-crafted feature engineering which is expensive to create and requires specific domain knowledge. The process of feature learning is costly and useless, because it generates many features without significance. Feature selection should be applied to determine the most relevant variables in traditional machine learning methods. However, feature selection is usually computationally very expensive that might also cause loss of valuable training information.

In recent years, deep learning based methods have been demonstrated to be particularly useful for learning time series. Purely feed-forward networks ignore the temporal structure of time series. Recurrent neural

networks (RNNs) have shown their capabilities for automatic temporal dependency learning from raw time series data [19]. For this reason, previous research attempted to use deeper architectures and more data in order to get better results compared with shallow neural networks. Pankaj et al. [20] have shown that stacking LSTMs as hidden layers and a sigmoid activation unit enables the learning of higher level temporal features and better results for anomaly detection. Niek et al. [21] use the same idea of stacking LSTMs and have found that one model with multitask learning yield higher accuracy than predicting them by separate models. Deeper neural networks with stacking recurrent layers can get good results for univariate time series analysis such as manufacturing events analysis. However, for multivariate time series, empirical experiments [22] showed that deeper neural networks have found similar or even worse results compared with shallow networks with only one or two hidden layers.

Additionally, the deep networks which are too big and have too many layers makes training difficult and often inefficient due to a lot of meaningless features. For example, a manufacturing time series may contain thousands of redundant features with only a small part of essential features. These redundant or uncorrelated features will degrade the performance of neural networks. In order to solve this problem, some unsupervised learning technologies such as Autoencoder and CNN have been introduced for automating features extraction from raw sensor inputs. Autoencoder [23] is used to learn the encoding of features with lower dimension from high dimensional multivariate time series, and the learned features representation will be used directly as the input of LSTMs. Nijat et al. [24] present a multi-stage deep learning method by stacking LSTMs Autoencoder and Deep Belief Networks to address the problem of multivariate time series classification for monitoring the product quality in the process industry. CNNs [25] is often used for image processing tasks to learn high-quality features for the classification task. Zheng et al. [26] have used CNNs to learn the features of multivariate time series automatically, and combine these learnt features at a final classification layer. Francisco et al. [27] present a deep learning framework DeepConvLSTM composed of convolutional layers and LSTM recurrent layers that are capable of automatically learning feature representations and temporal dependencies respectively.

Measured values coming from sensors in a manufacturing process usually have relationships among each other. We have already known that there are some relations among the sensor variables obtained with a manufacturing process. But the learning process of Autoencoder or other convolutional based methods is random and uncontrollable. Therefore, we believe that an approach integrated with process information, taking into account the relationships among the measured values by the sensors, will make the learning process more efficient and accurate.

3. Process Ontology modeling

In this section, we present a methodology for building a process ontology for manufacturing, such that its structure can assist in the design of deep neural networks. It is evident that the data from sensors are always collected with timestamps, constituting then, time series. It it the case of the sensors in the CPSs in an industrial context, that will continuously collect production-related data. In general, time series are a series of data points indexed in a time order [26, 28].

Definition 3.1. *A time series* $X = \{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$ *is a time ordered sequence of arrays, where each array* $x^{(t)} \in R^m$ *contains* m *values such as* $x_1^t, x_2^t, ..., x_m^t$ *and is associated with a timestamp. This array of* m *values correspond to the input variables measured at a certain time.*

Definition 3.1 was adopted by most works on time series analysis. But this definition can not represent the real manufacturing process. This definition only considers the order of the final products and their properties, but the relations between the manufacturing lines and the machines cannot be reflected.. The values in the same time point may come from different production lines with dependencies. These relations can be formulated with an ontology as a directed acyclic graph. So we can have a new definition for manufacturing time series as follow:

Definition 3.2. *A manufacturing time series* $X = \{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$ *is a time ordered sequence, where each point* $x^{(t)} \in G = (V, E)$ *is a directed hypergraph which contains* v *vertices to represent the sensor values*

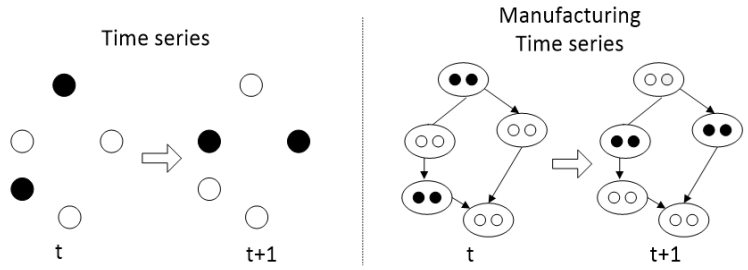

Fig. 1. Time series vs Manufacturing time series

$V = \{x_1^t, x_2^t, ..., x_v^t\}$ from m machines in $S = \{s_1, s_2, ..., s_m\}$ at timestamp t. The set of directed hyperedges above V is a set of ordered pairs (h_i, h_j) of disjoint subsets of V, where h_k contains the values measured by the sensors of machine s_k in S.

where s_i precedes s_j in a production line. We can have a formal definition of hyperedges $E = \{(s_i, s_j) | (s_i \text{ precedes } s_j),\ s_i, s_j \subseteq V,\ s_i \bigcap s_j = \emptyset)\}$.

In the example in Figure 1, we can clearly see the difference between traditional time series and manufacturing time series. Both examples in Figure 1 show the evolution of time series in time t and $t + 1$, the properties are independent in traditional time series, but each property in a manufacturing time series is dependent on other properties.

3.1. Abstract ontology for manufacturing time series

In general, a manufacturing process aims to achieve given requirements by transforming raw materials into physical objects with specific functions, shapes, structures, and other properties. The works in the previous subsection can provide ontologies for describing the manufacturing activities and their relationships which can be used in many software applications for monitoring a manufacturing process. These ontologies can only provide representations of manufacturing domain knowledge from a general point of view. The challenge here is how to represent the manufacturing domain knowledge properly to facilitate the integration with machine learning applications. For using a deep leaning approach to learn a model of the line from the time series coming from the sensors, we only need to know the hierarchical relationships among the different manufacturing activities and do not need to know the specific meaning of each activity. Therefore, we propose a novel methodology for developing the abstract ontology of the manufacturing process by using only a small number of classes and properties.

- We have production lines (represented by concept $\mathcal{L}$). A standard manufacturing process may include several production lines (meaning that each part is sequentially assembled into the final product through several lines), each line has its own geographic information and depends on the result of a previous line.
- Each production line can consist of multiple machines (represented by concept $\mathcal{M}$).
- The machines are equipped with sensors (represented by concept $\mathcal{S}$) to monitor their operation.
- The data collected from these sensors represented by datatype properties (property *measures*).
- The dependency relation about lines or machines can be expressed by an object property *precedes*.
- The inclusion relation between lines and machines can be expressed by an object property $hasMachine$.
- Similarly, the inclusion relation between machines and sensors can be expressed by an object property $hasSensor$.

Therefore, even a complicated production process can be simplified as an abstract ontology (Figure 2). The constructed abstract ontology can be aligned with any existing manufacturing ontology like PRONTO [11] or MASON[12] to get semantic information about the manufacturing process. In this paper, we do not consider the order among the machines for simplifying the description.

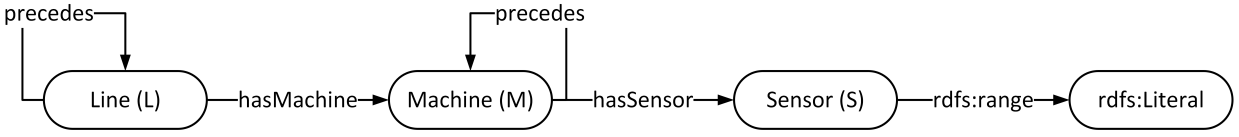

Fig. 2. Abstract ontology for manufacturing time series

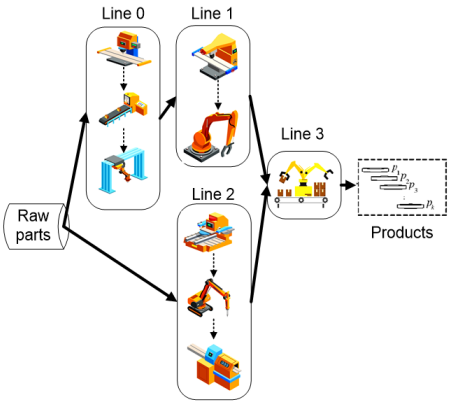

Fig. 3. A sample production line

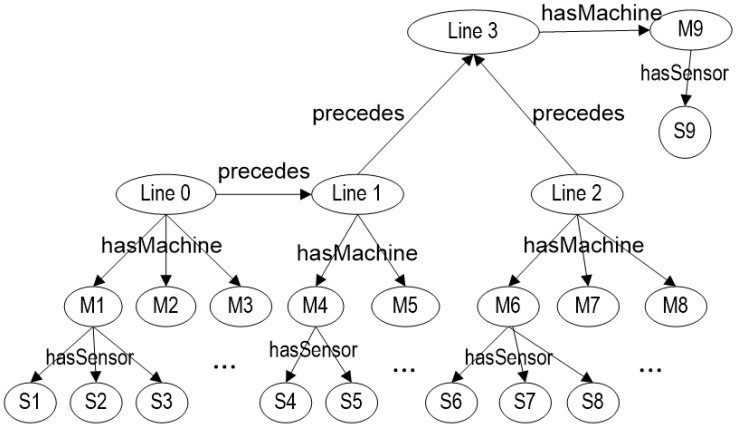

Fig. 4. Instantiation of our abstract ontology

3.2. Instantiation of the abstract ontology

Each factory has a specific production process that can be considered as an instantiation of this abstract ontology. In the example in Figure 3, the part begins at line 0 and then is processed by line 1, and another part begins at line 2. The processing parts from line 1 and line 2 can be assembled into a final product in the assembly line 3. All the information of manufacturing process can be expressed with the assistance of our abstract ontology in Figure 2. We propose a methodology that aims at instantiating the abstract ontology for a specific manufacturing process which is associated with a manufacturing time series.

1. Identify the production lines of a manufacturing process and their hierarchical relationships $precedes(line0, line1)$.
2. Identify the machines of each production line $hasMachine(line0, machine1)$.
3. Identify the sensors of each machine $hasSensor(machine1, s1)$.
4. Identify the values measured by the sensors $measures(s1, val1)$

Applying these steps to the manufacturing facility in Figure 3 yields to the instantiation presented in Figure 4 that will be used as a foundation for the design of the deep architecture.

4. Ontology based deep architecture

In this section, we present our algorithm to design the ontology-based deep architecture for learning the representation of products and its extension *OntoLSTM* for learning the temporal dependencies. An ontology-based deep architecture is a deep neural network model with stacking of several fully connected layers called *dense* layers deriving from an ontology. A flat representation of a product's properties as an input cannot reflect the domain knowledge. The key idea of our algorithm is that the representation of a product will be learnt from its own manufacturing lines, the representation of a manufacturing line can be learnt from its own machines and the representation of a machine will be learnt from its sensors and the collected data. Therefore, a product can be represented by these hierarchical relations.

In this way, the presented ontology-based deep architecture can learn a set of manufacturing component representations from both the data distribution and the manufacturing process ontology. In this architecture, each production line and each machine correspond to a dense layer in the neural network. The model construction (see algorithm 1) starts from adding a series of dense layers for learning the representation of production lines iteratively. Similarly, a series of dense layers can be added also for learning the the representation of machines. Then the object property *precedes* and *hasMachine* can help us to find the topological structure of our production lines. Therefore, we can connect the added dense layers with the topological structure. we can already get a representation of the manufacturing process. However, since industrial production data usually come in the form of time series, we need to stack an LSTM to capture the temporal dependencies of the data. If a production line does not have a precedent one (that is, it is the last assembly line), its associated dense layer is connected to the LSTM layer.

The proposed model is called *OntoLSTM*. The neural network includes two parts: an ontology-based neural network which is a stack of several dense layers for learning the representation of lines and machines of a manufacturing process, and a LSTM neural network for learning the temporal dependencies. Based on the manufacturing process ontology in Figure 4, we can construct our ontology-based LSTM neural network as in Figure 5 with Algorithm 1. The production assembly process is composed of 4 manufacturing lines with precedence dependencies and 9 machines. Therefore, our *OntoLSTM* model will have 13 dense layers and 1 LSTM layer. Each dense layer $dense(\sharp)(x))$[1] represents a machine or a production line x.

Algorithm 1 Ontology based deep architecture design

OntoLSTM(Instances of Ontology $\mathcal{O}$)
Let $\mathcal{T}$ be an empty model of neural networks
Let L be the set of individuals of concept $Line$ in $\mathcal{O}$
Let M = be the set of individuals of concept $Machine$ in $\mathcal{O}$
for each l in L **do**
  $\mathcal{T}.add(dense(\sharp)(l))$
end for
for each m in M **do**
  $\mathcal{T}.add(dense(\sharp)(m))$
end for
for each x in L **do**
  Let $s_l = \{p \; is \; a \; set \; instantiation \; of \; the \; concept \; Line \; in \; \mathcal{O} \mid precedes(p, x)\}$
  for each y in s_l **do**
    $dense(\sharp)(y) \rightarrow dense(\sharp)(x)$ // $' \rightarrow \; '$ indicate the two layers are connected
  end for
  Let $s_m = \{m \; is \; a \; set \; instantiation \; of \; the \; concept \; Machine \; in \; \mathcal{O} \mid hasMachine(x, m)\}$
  for each z in s_m **do**
    $dense(\sharp)(z) \rightarrow dense(\sharp)(x)$
  end for
  if $(s_r = \{h \; is \; a \; set \; instantiation \; of \; the \; concept \; Line \; in \; \mathcal{O} \mid precedes(x, h)\} = \emptyset)$ **then**
    $dense(\sharp)(x) \rightarrow LSTM$
  end if
end for
return $\mathcal{T}$

[1] In $dense(\sharp)(x)$, $\sharp$ represents the number of output neurons for the concerned layer.

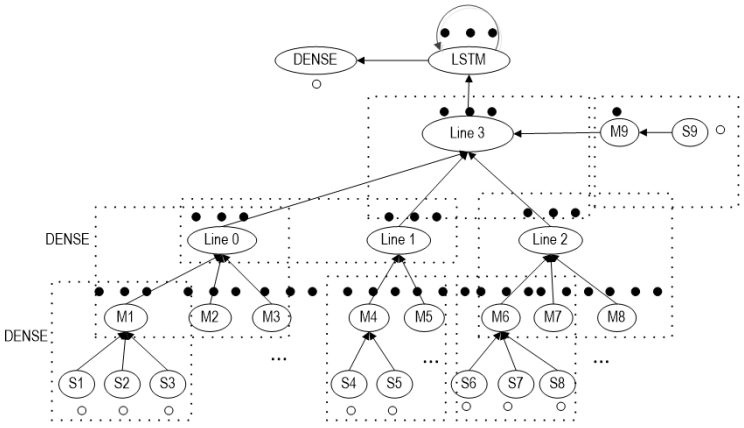

Fig. 5. The structure of *OntoLSTM* for our example of manufacturing process

5. Experiments

To show the feasibility of our approach, we present here a series of experiments by using a dataset proposed by a real manufacturing company. The goal is to predict which products will fail quality control.

5.1. Data sets

Bosch has supplied an open huge dataset[2] (14.3 Gb) from Kaggle. Each row of this data set represents a product, described by mixed features (numerical, categorical, date stamps), as it progresses through each production line and labels indicating the parts as good or bad. Measurements are done on each part along the assembly line to predict internal failures. Each part has a unique Id. The dataset contains an extremely large number of anonymized features. The training data has 1184687 samples with 968 numerical features, 2140 categorical features and 1156 date features. Features are named according to a convention that tells the production line, the machine on the line, and a feature number. For example, $L3\_S36\_F3939$ is a feature measured on *line*3, *machine*36, and the feature number is 3939. The date features provide a timestamp for each measurement. Each date column ends in a number that corresponds to the previous numerical or categorical feature number e.g. the value of $L0\_S0\_D1$ is the time at which $L0\_S0\_F0$ was taken.

5.2. Preprocessing and Model Construction

The dataset contains 1184687 observations taken from production lines. Each observation is a vector of thousands of sensor measurements plus a label for indicating *pass/fail* quality control. There are only a small amount of cases of failure (encoded as 1), whereas a much bigger amount of cases pass the quality control and are encoded as 0. This is a 1:171 proportion. To overcome the problems related to the learning of imbalanced data, several techniques [29] have been proposed, such as oversampling, undersampling, and cost-sensitive method. In this experiment, we use the techniques of undersampling by discarding data of the majority class. There are a number of methods available to undersample a dataset used in a typical classification problem. For undersampling the dataset, we use the toolkit Imblearn [30] which has already integrated principal oversampling and undersampling methods.

Although the features in this data set are still anonymous, the names of the features and their corresponding timestamps can give us the information about the production lines. We can see which feature comes from which line and which machine by considering its name, and also the order among the lines and

[2] https://www.kaggle.com/c/bosch-production-line-performance

Evaluation Metrics	OntoLSTM	DenseLSTM	AutoencoderLSTM
Accuracy	0.848	0.545	0.802
AUC	0.856	0.586	0.825
F1 score	0.848	0.29	0.828
Precision	0.939	0.98	0.901
Recall	0.774	0.176	0.79

Table 1. Classification performances of the different studied methods

stations with simple data mining on date features. All this information helps us to build an ontology of the manufacturing process from the Bosch dataset.

5.3. Results evaluation

Our task is a typical classification problem, we can use the typical evaluation metrics (Accuracy, AUC, F1...) for classification problems and compare with other methods. Standard cross-validation cannot be used in this experiment, because data have time stamps. Therefore, the first 80% of the data set is used for training, and the remaining 20% is used as a validation set. We made three simulations for each method to avoid sampling bias. Table 1 shows the performances of our model *OntoLSTM* compared to other two methods. DenseLSTM is a deep learning method which stack several dense layers and a LSTM layer. AutoencoderLSTM stacks Autoencoder and LSTM networks encoding the high dimensional input data to the hidden layers by using relevant activation functions and trying to reconstruct the original inputs through the decoder layer to minimize the reconstruction loss. Our model *OntoLSTM* outperforms the other two methods. Increasing the depth of the neural network can improve its performance, however it also increass the difficulty of training and optimization (gradient vanishing/exploding problem). DenseLSTM simply increases the number of layers in the neural network, however it almost failed in our experiments. For a huge data set with a large number of features, optimization becomes difficult. Unsupervised learning technologies like Autoencoder is usually used to solve the issue caused by a lot of features. Indeed, Autoencoder is usually used to get a better feature representation and reduce the dimension of features, but the process of encoding and decoding will make the algorithm inefficient. Our model OntolSTM builds the neural networks by integrating manufacturing process ontology for learning a better representation of line/machine. Therefore, OntolSTM can get better results while balancing the efficiency of the algorithm.

6. Conclusions

In this paper, we present a methodology to design an abstract ontology for representing manufacturing time series with the goal of facilitating the integration of machine learning applications. We introduce a novel ontology-based neural network model *OntoLSTM* for manufacturing time series classification. We have demonstrated with experiments that ontologies can assist the design of a deep architecture that is capable of extracting high-level cognitive features for representing the lines and machines of a manufacturing process. Stacking the ontology-based neural network with an LSTM network for learning the temporal dependencies yields a better performance when compared with conventional methods. In addition, we can easily build models for any new production lines by stacking simple neural network with our algorithm.

In the future, we are interested in manufacturing process mining with ontology-based neural networks. We expect the new model can predict not only the quality of the final products but also provide smart maintenance plans for the machines during the monitoring of the manufacturing process, to avoid the failures that could occur in the future.

Finally, we acknowledge the Normandy Region (France) for having funded these works in the framework of the NormanDeep project.

References

[1] Carlos Toro, Iñigo Barandiaran, and Jorge Posada. A perspective on knowledge based and intelligent systems implementation in industrie 4.0. *Procedia Computer Science*, 60:362–370, 2015.

[2] K. Efthymiou, K. Sipsas, D. Mourtzis, and G. Chryssolouris. On knowledge reuse for manufacturing systems design and planning: A semantic technology approach. *CIRP Journal of Manufacturing Science and Technology*, 8:1–11, 2015.

[3] Mercedes Argüello Casteleiro, Maria Jesus Fernandez Prieto, George Demetriou, Nava Maroto, Warren J. Read, Diego Maseda-Fernandez, Jose Julio Des Diz, Goran Nenadic, John A. Keane, and Robert Stevens. Ontology learning with deep learning: a case study on patient safety using pubmed. In *SWAT4LS*, 2016.

[4] Patrick Hohenecker and Thomas Lukasiewicz. Deep learning for ontology reasoning. *preprint arXiv:1705.10342*, 2017.

[5] Dejing Dou, Hao Wang, and Haishan Liu. Semantic data mining: A survey of ontology-based approaches. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 244–251. IEEE, 2015.

[6] Andreas Hotho, Alexander Maedche, and Steffen Staab. Ontology-based text document clustering. *KI*, 16(4):48–54, 2002.

[7] Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE, 2003.

[8] Liping Jing, Lixin Zhou, Michael K Ng, and J Zhexue Huang. Ontology-based distance measure for text clustering. In *Proc. of SIAM SDM workshop on text mining, Bethesda, Maryland, USA*, 2006.

[9] Hao Wang, Dejing Dou, and Daniel Lowd. Ontology-based deep restricted boltzmann machine. In *International Conference on Database and Expert Systems Applications*, pages 431–445. Springer, 2016.

[10] Nhathai Phan, Dejing Dou, Hao Wang, David Kil, and Brigitte Piniewski. Ontology-based deep learning for human behavior prediction with explanations in health social networks. *Information sciences*, 384:298–313, 2017.

[11] Marcela Vegetti, Gabriela P Henning, and Horacio P Leone. Product ontology: definition of an ontology for the complex product modelling domain. In *Proceedings of the Mercosur Congress on Process Systems Engineering*, 2005.

[12] Severin Lemaignan, Ali Siadat, J-Y Dantan, and Anatoli Semenenko. Mason: A proposal for an ontology of manufacturing domain. In *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, pages 195–200. IEEE, 2006.

[13] R Ferrario and A Oltramari. A first-order cutting process ontology for sheet metal parts. *Formal Ontologies Meet Industry*, 198:22, 2009.

[14] Suriati Akmal and Rafael Batres. A methodology for developing manufacturing process ontologies. *Journal of Japan Industrial Management Association*, 64(2E):303–316, 2013.

[15] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.

[18] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[19] Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in neural information processing systems*, pages 190–198, 2013.

[20] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[21] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In *International Conference on Advanced Information Systems Engineering*, pages 477–492. Springer, 2017.

[22] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[23] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[24] Nijat Mehdiyev, Johannes Lahann, Andreas Emrich, David Enke, Peter Fettke, and Peter Loos. Time series classification using deep learning for process planning: A case from the process industry. *Procedia Computer Science*, 114:242–249, 2017.

[25] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[26] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *Int. Conf. on Web-Age Information Management*, pages 298–310. Springer, 2014.

[27] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[28] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[29] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

[30] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.