Long Short-Term Memory

Long Short-Term Memory

The LSTM is one of the most frequently used networks of recurrent neural networks with wide application outlook.

From: Applications of Artificial Intelligence in Process Systems Engineering, 2021

Related terms:

Artificial Intelligence, Backpropagation, Learning System, Lithium, Multilayer Neural Networks, Neural Network, Wastewater Treatment, Recurrent Neural Network, Molecular Structure, Deep Neural Network

View all Topics

31st European Symposium on Computer Aided Process Engineering

Akash Das, ... Nitin Dutt Chaturvedi, in Computer Aided Chemical Engineering, 2021

3.3 Long Short-Term Memory Model

The Long-Short Term Memory (LSTM) structure was motivated by an analysis of error flow in existing RNNs, which found that long time lags were inaccessible to existing architectures because the backpropagated errors either blows up or decays exponentially. Unlike the recurrent unit, which computes a weighted sum of the input signal and applies a nonlinear function, each *jth* LSTM unit maintains a memory *cjt* at time *t*. The output *ajt*, or the activation of the LSTM unit is then given by Eq. (4),

(4)

The memory cell *cjt*, is updated by partially forgetting the existing memory and adding a new memory content *cjt* as shown in Eq. (5),

(5)

where the new memory content is given by,

(6)

The extent to which the existing memory is forgotten is modulated by a forget gate fjt And the degree to which the new memory content is added to the memory cell is modulated by an input gate ijt. Thus unlike the traditional RNN, which over-writes its content at each step, an LSTM can regulate whether to keep the existing memory via the introduction of the gates, hence potentially capturing the long-term dependencies (Gravesand Schmidhubera, 2005).

This research considers a model with two bidirectional LSTM layers and one layer of unidirectional LSTM, the output of which is fed to a dense network of five hidden layers.

> Read full chapter

31st European Symposium on Computer Aided Process Engineering

Yeongryeol Choi, ... Junghwan Kim, in Computer Aided Chemical Engineering, 2021

2.4 Long short-term memory (LSTM) Model

The LSTM model, a recurrent neural network (RNN)-based algorithm, was used as the machine-learning algorithm (Hochreiter and Schmidhuber, 1997). The LSTM model can be used to solve the vanishing gradient problem; the weight update of the activation function converges to zero when long-term data are learned in an RNN algorithm, which is useful for data with a long learning period.

The predictive model is composed of an input layer containing selected features, a hidden LSTM layer consisting of four hidden units, and an output layer that outputs the temperature of the 2,3-BDO production stage. Seventy percent of the total data were randomly selected and used for training and validation, and 100 % of the data, including the training data, were used to determine the model performance.

The hyperparameters of the model were set based on a previous study (Kwon *et al.*, 2021). The number of batches, the number of epochs, the learning rate, and the activation function were set to 10, 1000, 0.01, and ELU, respectively. To prevent under-and/or over-fitting of the predictive model, the "patience" option was used as the early stopping option; it is defined as the number of epochs without an improvement after which the training will be stopped. The number of epochs was set at five for this study.

> Read full chapter

> Read full chapter

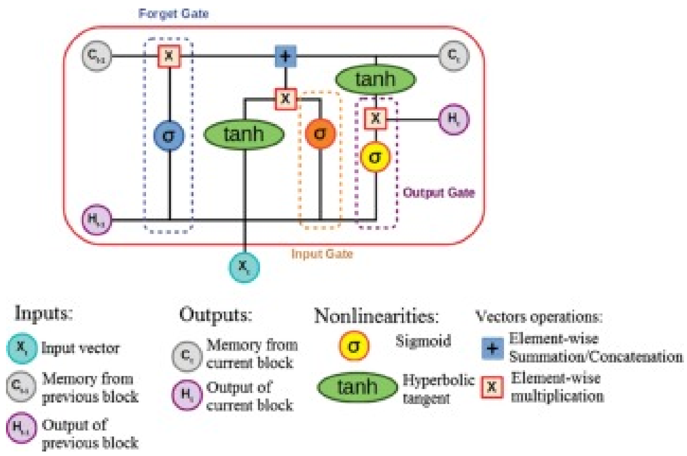
Unsupervised recurrent deep learning scheme for process monitoring

Fouzi Harrou, ... Abdelkader Dairi, in Statistical Process Monitoring Using Advanced Data-Driven and Deep Learning Approaches, 2021

7.2.2 Long short-term memory

Machine learning has been researched extensively over the past three decades. Conventional neural networks are a commonly used machine learning approach. As mentioned above, the main characteristics underlying these networks are the presence of full connections between adjacent layers and the absence of connections between nodes within the same layer. Thus, this type of network is suited to handle sequential data and describe temporal dependencies in the data because it considers only the current measurement without memorizing past measurements. As discussed above, to overcome this problem, the internal memory of an RNN has been used to handle sequential data by considering the actual and previously received measurements. In other words, the hidden units in an RNN receive feedback from the previous state to the current state. Because the depth of the RNN is the time span, information can be lost through time and the error can propagate back. Accordingly, the accuracy of the RNN can be degraded when the time span becomes longer due to the vanishing gradient and exploding gradient problems. To address this issue, long short-term memory (LSTM), which is an extended version of RNN, was first introduced by Hochreiter and Schmidhuber [15] in 1997. LSTM models have been largely utilized in many applications, including handwriting recognition [16], language modeling [17] and translation [18,19], acoustic modeling of speech [20], speech synthesis [21,22], protein structure prediction [23–25], and analysis of audio and video data [26–30]. Moreover, in the modern LSTM architecture, there are peephole connections between internal cells and the gates in the same cell for learning the accurate timing of the outputs [31]. This section first provides an overview of LSTMs and then discusses how they can be used for modeling and process monitoring. We also describe gated recurrent units (GRU), another improved version of RNNs.

The LSTMs are designed as an extension of simple RNNs to solve the vanishing gradient problem by explicitly incorporating a memory unit into the network. They are based on memories and gates making them suitable for learning long-term dependencies. Fig. 7.3 displays a schematic representation of an LSTM.



Forget Gate
Input Gate
Output Gate
tanh
Inputs:
Input vector
Memory from previous block
Output of previous block
Outputs:
Memory from current block
Output of current block
Nonlinearities:
Sigmoid
Hyperbolic tangent
Vectors operations:
Element-wise Summation/Concatenation
Element-wise multiplication

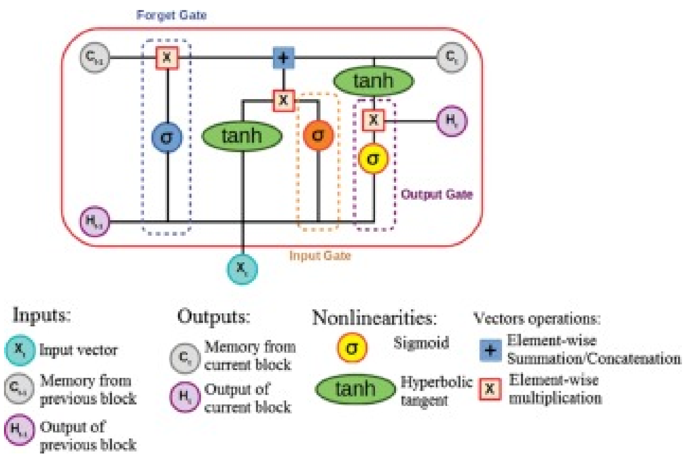


Figure 7.3. A basic illustrative representation of an LSTM unit.

The principal component of LSTM resides in its cell state, which is the horizontal chain shown in the top of the graph (Fig. 7.4). In LSTM, information can be removed or added to the cell by using structures called gates.

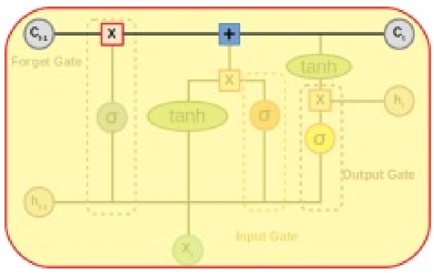


Figure 7.4. Cell state in an LSTM model.

We designed a combined LSTM model as a concatenation of several cell units instead of conventional neural network layers. We now briefly investigate the properties of the LSTM unit. As shown in Fig. 7.3, an LSTM cell comprises three inputs: is the input observation at the current time point, represents the output generated from the preceding LSTM cell, and denotes the memory of the preceding cell. Also, each LSTM cell contains two outputs, which are the output of the actual network and the memory of the current unit, respectively. It is composed of three kinds of gates, namely the input gate, the forget gate, and the output gate (Fig. 7.3). Each gate comprises a sigmoid neural net layer and a pointwise multiplication operation. LSTMs can be seen when unfolded as a chain-like structure. In other words, it is a loop repeating module with a different structure (Fig. 7.5). Depending on the input and the internal feedback (memory state), the output will be generated in a special manner based on gates and feed embedded layers structure. The layers here have activation units based on sigmoid and tanh.

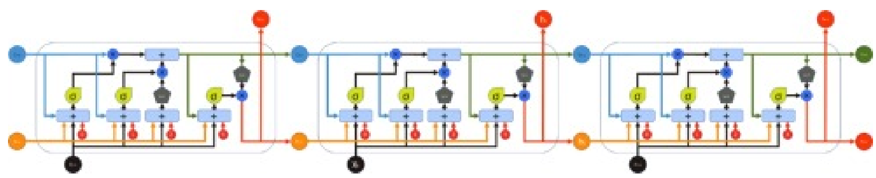


Figure 7.5. Schematic representation of an LSTM when unrolled in time.

- In this step, the old cell state, , is actualized with the new cell state, , and the past cell state, , is actualized with the new cell state, (Fig. 7.8). To do so, the old cell state is multiplied by to forget the irrelevant information, and the candidate, , is added. This represents the new candidate values scaled by how much we choose to update every state value:(7.6)Figure 7.8. Step 3 in LSTM modeling.
- Finally, the output is computed in two steps (Fig. 7.9). First, a sigmoid layer is utilized to select the relevant portions of the cell state to be transmitted to the output (Eq. (7.7)):(7.7) Then, the cell state (Eq. (7.7)) is passed via tanh (for normalizing values within the range −1 and 1) and multiplied by the output of the sigmoid gate; thus we keep only the portions we selected to output (Eq. (7.8)):(7.8)Figure 7.9. Step 4 in LSTM modeling.

At the end of the cycle, the hidden layer, which represents the output of the cycle and the memory state are ready for the next cycle. In summary, in the LSTM model, the three gates are trained for learning what information can be maintained in the memory, how long can be stored, and when it can be read out. Combining several memory cells into blocks permits them to share the same gates, and hence the number of adaptive parameters is reduced.

> Read full chapter

Deep learning in QSPR modeling for the prediction of critical properties

Yang Su, Weifeng Shen, in Applications of Artificial Intelligence in Process Systems Engineering, 2021

2.4 Deep neural network

A DNN combining Tree-LSTM and BPNN is developed in this work. The Tree-LSTM neural network is employed for depicting molecular tree data-structures with the canonical molecular signature while the BPNN is used to correlate properties.

The Child-sum Tree-LSTM can be used to the dependency tree while the N-ary Tree-LSTM is applied to the constituency tree [39], and the mathematical models of these two Tree-LSTM models are listed in Table 2. The gating vectors and memory cell updates of the Tree-LSTM are dependent on the states of child units, which is different from the standard LSTM. Additionally, instead of a single forget gate, the Tree-LSTM unit contains one forget gate *fjk* for each child *k*. This allows the Tree-LSTM to incorporate information selectively from each child. Since the components of the Child-Sum Tree-LSTM are calculated from the sum of child hidden states h_k, the Child-Sum Tree-LSTM is well suited for trees with high branching factor or whose children are unordered. The vector is the sum of the hidden states of all sub nodes under the current node j in the Child-sum Tree-LSTM model. The N-ary Tree-LSTM model can be utilized in the tree structure where the branching factor is at most N and where children are ordered from 1 to N. For any node j, the hidden state and memory cell of its k_{th} child are written as h_{jk} and c_{jk}, respectively. The introduction of separate parameter matrices for each child k allows the N-ary Tree-LSTM model to learn more fine-grained conditioning on the states of a unit's children than those of Child-Sum Tree-LSTM.

can vary the computing graph automatically. The BPNN accepts the output vectors from the Tree-LSTM network and correlates them with the property values. In this way, a DNN is built based on the Tree-LSTM network and BPNN.

can vary the computing graph automatically. The BPNN accepts the output vectors from the Tree-LSTM network and correlates them with the property values. In this way, a DNN is built based on the Tree-LSTM network and BPNN.

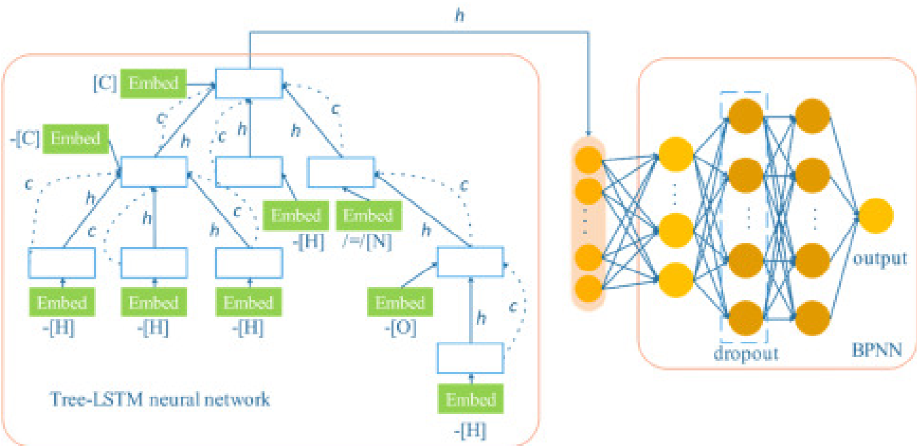


Fig. 5. The computing graph of the neural network describing the molecule acetaldoxime and predicting properties.

Moreover, in this work, the aim of the DNN is to predict a numeric value instead of classification. Hereby, there is no need to employ the activation function “softmax” [43]. The regularization technique “dropout” [44] is introduced to the BPNN for reducing overfitting. Huber loss [45] is adopted as the loss function in the training process, which is different from the classification scheme of Tree-LSTM network. The information about the DNN is provided in Tables 3 and 4.

Table 3. The structural parameters of the DNN.

Names of the DNN structural parameters	Values
Shape of embedding vectors	(50,1)
Shape of parameters of Tree-LSTM	(128,128)
Shape of output vectors of Tree-LSTM	(128,1)
Layer number of the BPNN	3

Table 4. The hyper parameters of training the DNN.

Names of the hyper parameters	Values
Learning rate	0.02 (the first 200 epochs); 0.0001 (others)
L2 weight decay	0.00001
Batch size of training set	200
Batch size of testing set	200

The regularization technique “dropout” is used to reduce overfitting in the proposed DNN. The “dropout” is easily implemented by randomly selecting nodes of a neural network to be dropped-out with a given probability (e.g., 20%) in each weight update cycle. With the cross validation, the expected probability is located between 5% and 25%.

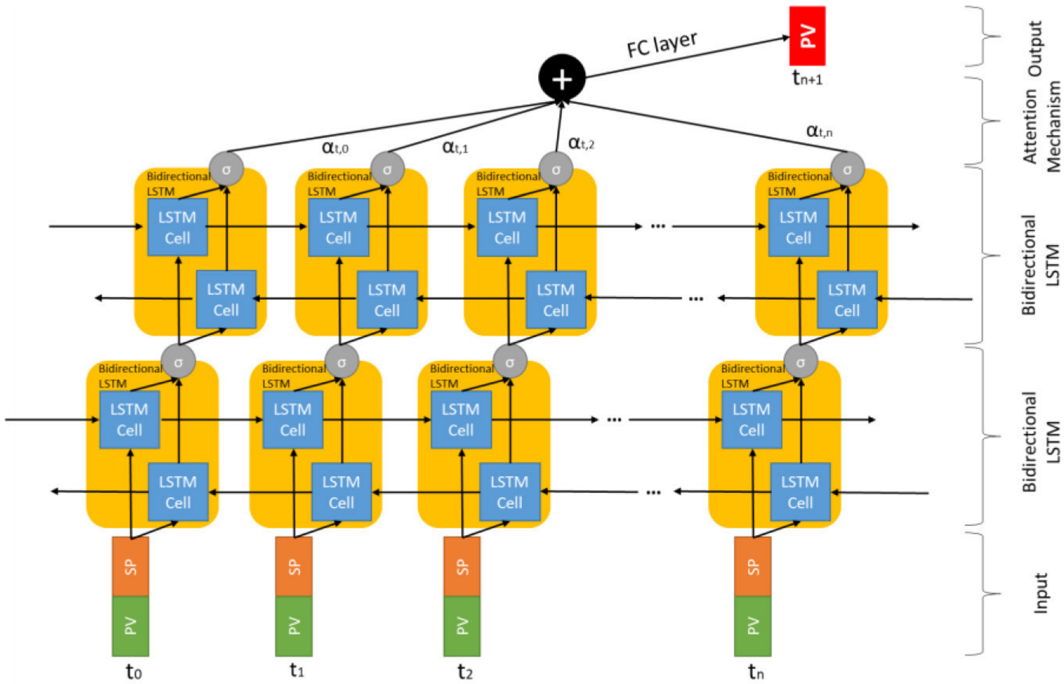
> Read full chapter

30th European Symposium on Computer Aided Process Engineering

Wenbo Zhu, ... Jose Romagnoli, in Computer Aided Chemical Engineering, 2020

3.2 Surrogate Model

In this work, a LSTM-based regression model is opted to learn the time sequential dynamics in the CRR process. The basic idea here is that the PVs at next time step are predicted using the SP and PV in the previous time steps. Besides the basic LSTM layout, recent developed techniques namely, bidirectional RNN structure (Schuster and Paliwal, 1997) and attention mechanism (Rocktaschel et al., 2015) are also incorporated in the regression model. After incorporating the different elements mentioned above, a schematic representation of the surrogate model is depicted by Figure 2.


FC layer
PV
tn+1
αt,0
αt,1
αt,2
αt,n
Bidirectional LSTM
LSTM Cell
SP
PV
t0
t1
t2
tn
Output
Attention Mechanism
Bidirectional LSTM
Bidirectional LSTM
Input

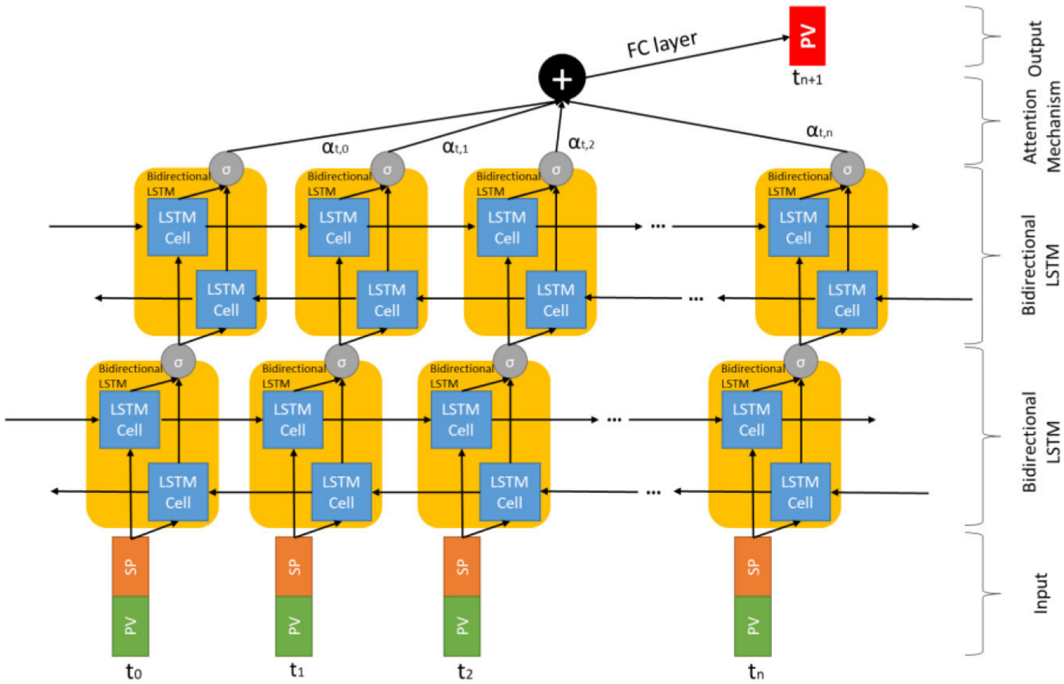


Figure 2. The LSTM-based surrogate model

Once the dynamic surrogate model is accurate enough to mimic the process dynamics, then a feed-forward neural network is trained separately to predict profit using current market prices of natural gas and NGLs, as well as the SP and PV values that represent the current status of the system.

> Read full chapter

31st European Symposium on Computer Aided Process Engineering

Yongbeom Shin, Dongil Shin, in Computer Aided Chemical Engineering, 2021

4.1 Model Performance Comparison

For the performance evaluation of the optimal FNN, LSTM and, AutoML, the test MSEs were compared. In the case of FNN, the average MSE was 0.000400, for LSTM, 0.000207, for AutoML, 0.000253. Figure 2 shows the model performances of FNN, LSTM and, AutoML. Figure 3 shows the prediction performance according to the future time step: LSTM performs the best with the lowest error in all predictions, more suitable for predicting process time-series data. The performance of AutoML was superior compared to the FNN model, and especially, the total development time was reduced to 1/25 compared to the LSTM model with a complex structure that required more than a week of work. Development of predictive models using AutoML only required domain knowledge that can be designed by field engineers and showed similar performance to models designed by experts, thus the model development using AutoML was concluded as applicable.

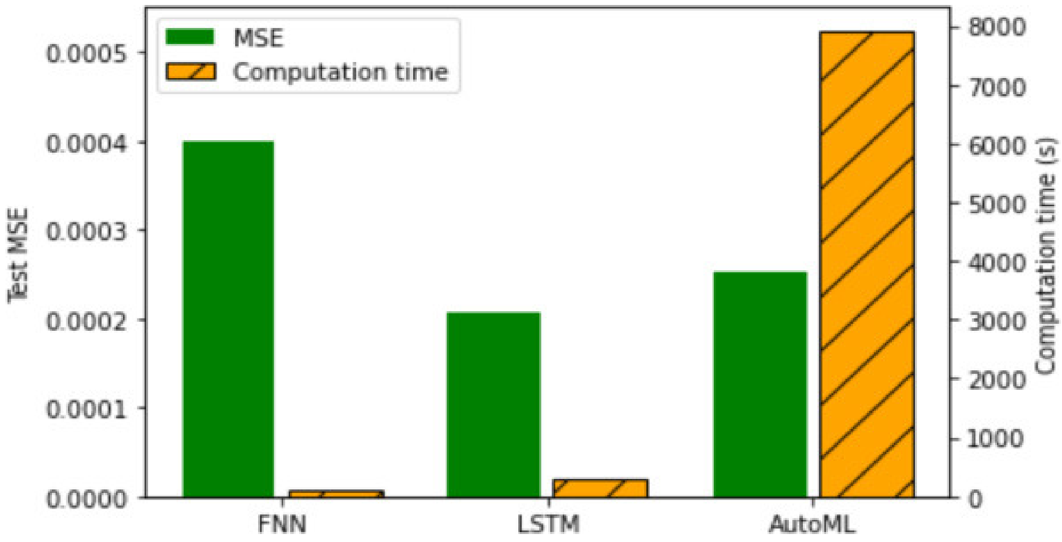
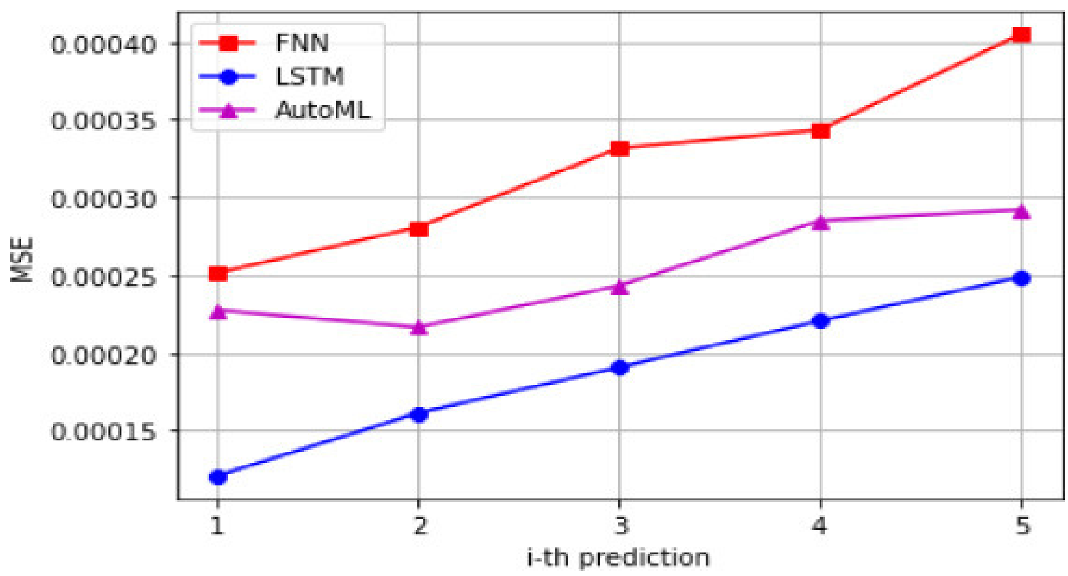


Figure 2. Model performance of FNN, LSTM and AutoML.


FNN
LSTM
AutoML
MSE
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
1
2
3
4
5
i-th prediction

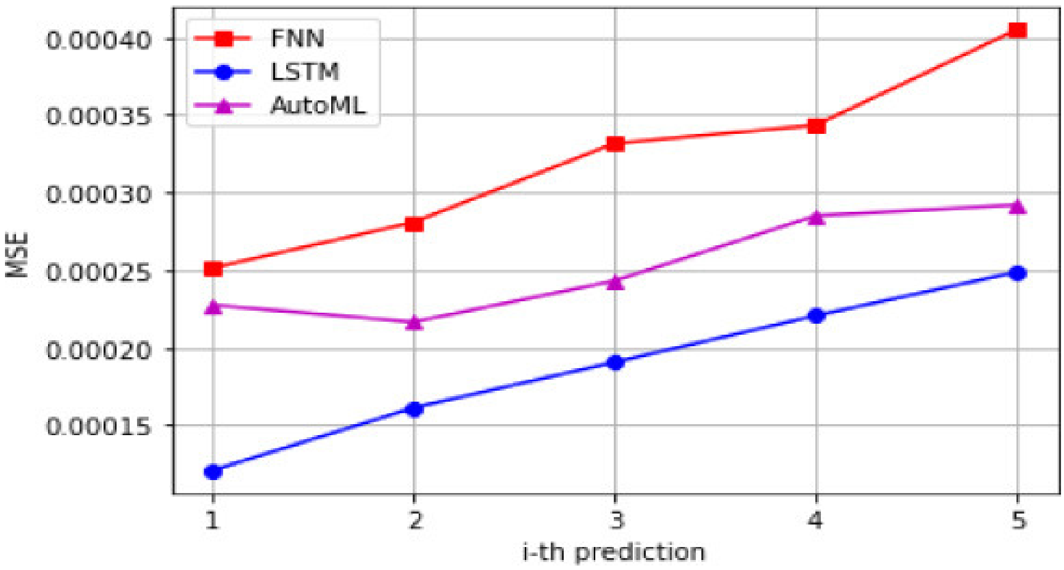


Figure 3. MSE according to the output sequence.

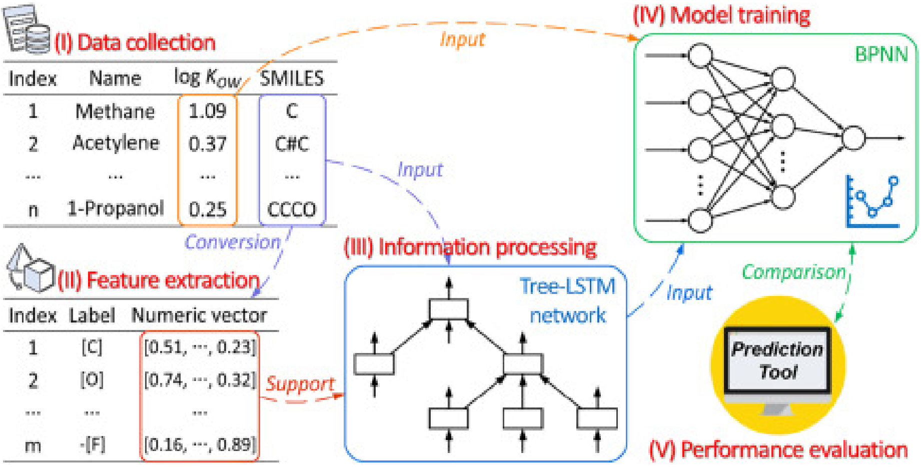
> Read full chapter

Predictive deep learning models for environmental properties

Zihao Wang, Weifeng Shen, in Applications of Artificial Intelligence in Process Systems Engineering, 2021

2 Methodology

A DNN model, which couples the Tree-LSTM network and back-propagation neural network (BPNN), was developed in this study based on the deep learning approach. It was built to specialize in the determination of the correlation between molecular structures and log K_{oc} values of organic compounds. The process of developing a reliable QSPR model with the DNN model is comprised of the following five basic steps, as illustrated in the Fig. 1.


(I) Data collection
Index
Name
log K_{OW}
SMILES
1
Methane
1.09
C
2
Acetylene
0.37
C#C
n
1-Propanol
0.25
CCCO
Conversion
(II) Feature extraction
Index
Label
Numeric vector
1
[C]
[0.51, ···, 0.23]
2
[O]
[0.74, ···, 0.32]
m
-[F]
[0.16, ···, 0.89]
Support
Input
Input
(III) Information processing
Tree-LSTM network
Input
(IV) Model training
BPNN
Comparison
Prediction Tool
(V) Performance evaluation

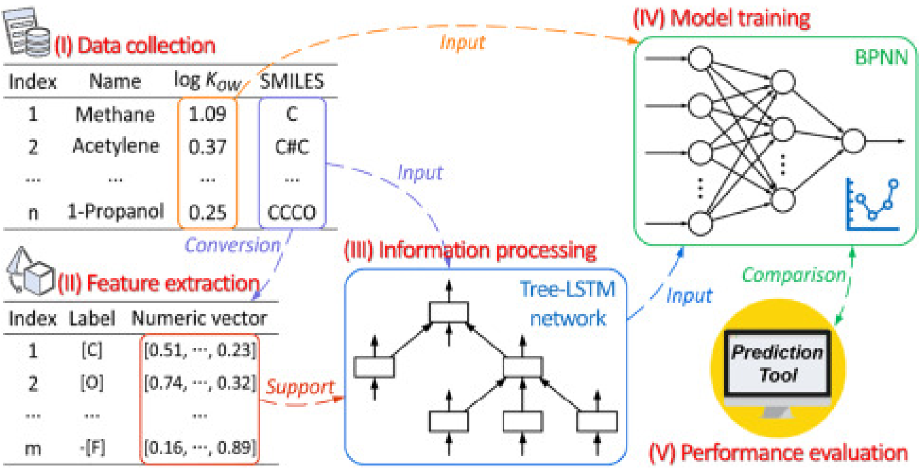


Fig. 1. The schematic diagram of the process for developing a QSPR model with the deep learning approach.

Step 1: Data collection. The experimentally measured log *K*ow values and simplified molecular-input line-entry system (SMILES) strings of compounds were collected since they are necessary for developing a QSPR model. Herein, the SMILES strings sufficed for representing the basic molecular structural information.

Step 2: Feature extraction. The SMILES strings of compounds were utilized to generate a list of numeric vectors based on a proposed atom embedding algorithm which was implemented with the atom signatures. The vectors are able to describe molecular structures and represent their features.

Step 3: Information processing. The SMILES strings were converted to canonical molecular signatures with the theory of the canonizing molecular graph [58]. On this basis, these signatures were mapped onto the Tree-LSTM networks with the aim of creating the vectors as inputs for the BPNN.

Step 4: Model training. After receiving the inputs from the Tree-LSTM networks, the BPNN supported the correlation process, and it was repeatedly run to learn a satisfactory QSPR model. In the training process, parameters were updated to optimize the DNN model and finally the QSPR model with better performance was preserved for log Kow estimation.

Step 5: Performance evaluation. Based on the developed QSPR model, the generalization ability was assessed by the predictive performance of an external dataset. And the external competitiveness of the QSPR model was evaluated by comparing to an authoritative predictive model.

All the above steps for obtaining the QSPR model to predict the *K*ow were achieved with a series of programs which were written in the Python language and successfully tested on Windows platforms.

2.1 Data acquisition and processing

The dimensionless K_{OW} values span over 10 orders of magnitude and therefore the decimal logarithm of K_{OW} (log K_{OW}) was frequently adopted in property estimation. A large number of experimentally measured log K_{OW} values of chemical compounds were collected [59], and all the experimental values were originated from references to guarantee the reasonability of the predictive model. To investigate the QSPR model for organic compounds, a number of irrelevant compounds were eliminated. The excluded irrelevant compounds involve the inorganic compounds (e.g., carbon dioxide, sulfur hexafluoride, and hydrazine), metal-organic compounds (i.e., the organic compounds containing metal atoms such as sodium, chromium or/and stannum) and mixtures consisting two or more compounds. Hence, the remaining 10,754 pure organic compounds were assembled for the model development.

The dimensionless K_{OW} values span over 10 orders of magnitude and therefore the decimal logarithm of K_{OW} (log K_{OW}) was frequently adopted in property estimation. A large number of experimentally measured log K_{OW} values of chemical compounds were collected [59], and all the experimental values were originated from references to guarantee the reasonability of the predictive model. To investigate the QSPR model for organic compounds, a number of irrelevant compounds were eliminated. The excluded irrelevant compounds involve the inorganic compounds (e.g., carbon dioxide, sulfur hexafluoride, and hydrazine), metal-organic compounds (i.e., the organic compounds containing metal atoms such as sodium, chromium or/and stannum) and mixtures consisting two or more compounds. Hence, the remaining 10,754 pure organic compounds were assembled for the model development.

As a large dataset was collected, the data cleaning is essential to be carried out by detecting and removing outliers, which contain gross errors. Accordingly, the Pauta criterion [60], also referred to as the three-sigma rule, was applied for the cleaning process. It describes that 99.73% of all values of a normally distributed parameter fall within three times the standard deviation (σ) of the average (μ). Any error beyond this interval is not a random error but a gross error. Accordingly, data points which include gross errors are regarded as outliers and should be excluded from the sample data. The data cleaning process with Pauta criterion is graphically illustrated in the Fig. 2.

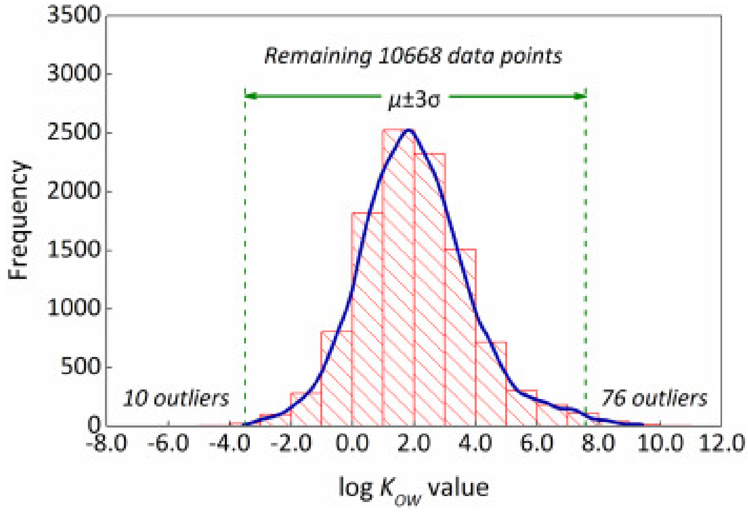


Fig. 2. The distribution of experimental data of 10,754 organic compounds.

As a result, 86 out of 10,754 organic compounds (about 0.8% of the dataset) were detected as outliers based on their experimental values and they were removed from the dataset. The remaining 10,668 organic compounds were preserved as the final dataset for developing a QSPR model to predict log K_{OW}. The dataset of compounds spans a wide class of molecular structures including aliphatic and aromatic hydrocarbons, alcohols and phenols, heterocyclic compounds, amines, acids, ketones, esters, aldehydes, ethers, and so on. In order to demonstrate the chemical diversity of the dataset, the corresponding counts of different types were detailed in Table 1, and their distributions in the training, test, and external sets were also provided. Since the subsets were divided with a random selection routine, proportions of different types of compounds in each subset approximate corresponding proportions for the compounds of subset in the entire dataset.

The signature molecular descriptor was introduced specifically for describing molecular structures, and all the connectivity information for every atom in a molecule was retained. Additionally, it can be theoretically applied to represent any organic compound which means that it is able to cover various molecular structures without limitation.

The signature molecular descriptor was introduced specifically for describing molecular structures, and all the connectivity information for every atom in a molecule was retained. Additionally, it can be theoretically applied to represent any organic compound which means that it is able to cover various molecular structures without limitation.

Herein, taking 1-propanol (CAS RN: 71-23-8; SMILES: CCCO) as an example. When a root atom was specified to the molecule, a tree spanning all atoms and bonds of the molecule was constructed (refer to Fig. 3A), and the signatures were generated relying on the theory of canonizing molecular graph [58].

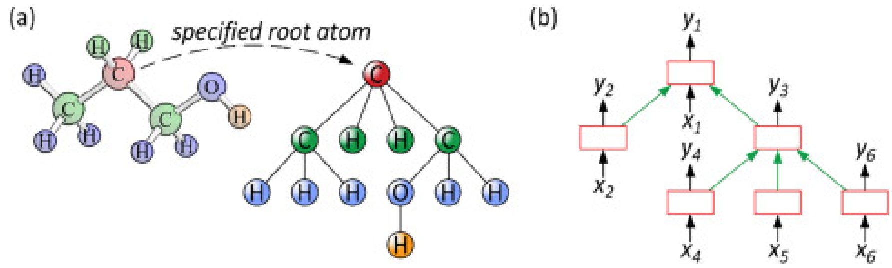


Fig. 3. The tree structures expressing information of (A) the signature tree for the 1-propanol molecule and (B) the Tree-LSTM network.

Up to a point, the syntactic property of natural languages is analogous to the connectivity information for atoms in molecules. The former one is able to be captured by the Tree-LSTM network while the latter one can be expressed with a signature. In addition, the tree structure of Tree-LSTM networks (refer to Fig. 3B) is similar to the signature tree displayed in Fig. 3A. Therefore, it was assumed that the molecular structural information can be processed and transmitted by coupling the signatures and Tree-LSTM network, and this was proven to be practical [47].

2.3 Signature molecular descriptor and encoding rules

The structural information of molecules was extracted from the SMILES strings and expressed by atomic and molecular signatures with text form in this study. The atomic signatures can represent the substructure of the molecule while the molecular signatures describe the whole molecule. To specify atomic features, atoms were converted to strings relying on encoding rules which refer the regulations defined in SMARTS [63] (a straightforward extension of SMILES for describing molecular substructures). In order to be well applied to this task, some new definitions were made as a complement of the encoding rules. Herein, RDKit [64] was adopted as an auxiliary tool for implementing the encoding rules by identifying the element symbols of atoms, the types of chemical bonds, the types of chirality centers and so forth.

Atomic signature of height 1, also called 1-signature, contains only the root atom and its chemical bonds along with bonded atoms (refer to Fig. 4) [47]. The

1-signature of each atom in molecules were generated with encoding rules, and subsequently a series of substrings representing molecular features were extracted with adopting the atom embedding program [57]. During the embedding process, each substring was assigned a numeric vector for distinction and adopted as the label for this vector. In spite of that these vectors were only used to represent molecular features. The structural information of molecules and atom connectivity will be totally preserved with the aid of the combination of signatures and the Tree-LSTM networks. For illustrative purpose, all the symbols involving in the labels of molecular features are listed and explained in Table 2.

1-signature of each atom in molecules were generated with encoding rules, and subsequently a series of substrings representing molecular features were extracted with adopting the atom embedding program [57]. During the embedding process, each substring was assigned a numeric vector for distinction and adopted as the label for this vector. In spite of that these vectors were only used to represent molecular features. The structural information of molecules and atom connectivity will be totally preserved with the aid of the combination of signatures and the Tree-LSTM networks. For illustrative purpose, all the symbols involving in the labels of molecular features are listed and explained in Table 2.

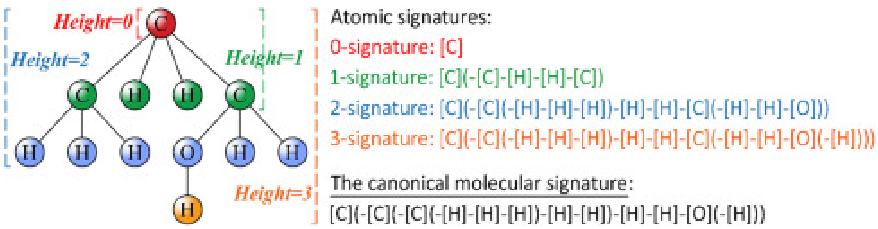


Fig. 4. The signature descriptors generated from the 1-propanol molecule.

Table 2. The explanations and examples for symbols involved in the labels of molecular features.

Symbol	Explanation	Example
[A]	Atom in aliphatic compound	[C]—carbon atom in an aliphatic compound
[a]	Atom in aromatic compound	[c]—carbon atom in an aromatic compound
| r	Atom in a ring	[C | r]—carbon atom in a ring
+ (inside [])	Atom with a positive charge	[N +]—nitrogen atom with a positive charge
– (inside [])	Atom with a negative charge	[N-]—nitrogen atom with a negative charge
– (outside [])	Single bond	-[C]—carbon atom with single bond
=	Double bond	=[C]—carbon atom with double bond
#	Triple bond	#[C]—carbon atom with triple bond
:	Aromatic bond	:[c]—carbon atom with aromatic bond
/=\	Atoms in same side	/=\[C]—carbon atom in same side of connected atom
/=/	Atoms in opposite side	/=/[C]—carbon atom in opposite side of connected atom
*	Atom is a r-chirality center	[C*]—carbon atom is a r-chirality center
**	Atom is a s-chirality center	[C**]—carbon atom is a s-chirality center

The molecular signature was defined as the linear combination of atomic signatures covering all the atoms and bonds [47]. However, the molecular signatures involve redundant and duplicated information. Accordingly, canonical molecular signature, the lexicographically largest atomic signature, which suffices to represent the molecular graph, was introduced to simplify the molecular signature [58]. Herein, to be used in conjunction with the Tree-LSTM network, the canonical molecular signature of each compound was generated in a unique manner for describing the molecular structure. For instance, the canonical molecular signature for 1-propanol (CASRN: 71-23-8; SMILES: CCCO) is represented as [C](-[C](-[C](-[H]-[H]-[H])-[H]-[H])-[H]-[H]-[O](-[H])) relying on the canonizing algorithm [47] and proposed encoding rules.

involve redundant and duplicated information. Accordingly, canonical molecular signature, the lexicographically largest atomic signature, which suffices to represent the molecular graph, was introduced to simplify the molecular signature [58]. Herein, to be used in conjunction with the Tree-LSTM network, the canonical molecular signature of each compound was generated in a unique manner for describing the molecular structure. For instance, the canonical molecular signature for 1-propanol (CASRN: 71-23-8; SMILES: CCCO) is represented as [C](-[C](-[C](-[H]-[H]-[H])-[H]-[H])-[H]-[H]-[O](-[H])) relying on the canonizing algorithm [47] and proposed encoding rules.

The molecular features chosen in the QSPR model rely on the molecular structure of the compound (refer to Fig. 5). First, the canonical molecular signature was generated for a compound, and then the Tree-LSTM network for this compound was built according to the signature tree for mapping the molecular structure. Afterwards, the numeric vectors representing molecular features were fed to the nodes of the Tree-LSTM network. Finally, a vector generated in the Tree-LSTM network was introduced to the BPNN for training the predictive model.

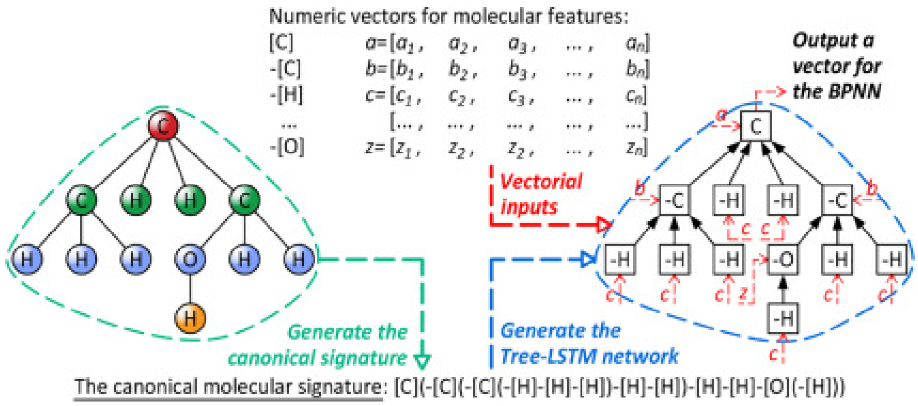


Fig. 5. The way of selecting molecular features for presenting the compound during predictions.

2.4 Structural features and parameters of DNN

In the DNN model, the Tree-LSTM network was utilized in conjunction with the BPNN to develop a QSPR model for predicting log K_{ow}. The Tree-LSTM network was employed to describe molecular structures with canonical molecular signatures while the BPNN was used to correlate structures and properties. Back-propagation (BP) algorithm is a supervised learning procedure in the machine learning process, and it was commonly used to train the DNN [65–67]. In this study, the BPNN was built with three layers including one input layer, one hidden layer, and one output layer. The topological structure of the fully connected three-layer neural network is graphically presented in Fig. 6. The input layer receives the vectors produced by Tree-LSTM network and gives the predicted log K_{ow} values. As single layers of linear neurons, the hidden layer take in a set of weighted inputs from the input layer and produce an output for the output layer.

layers of linear neurons, the hidden layer take in a set of weighted inputs from the input layer and produce an output for the output layer.

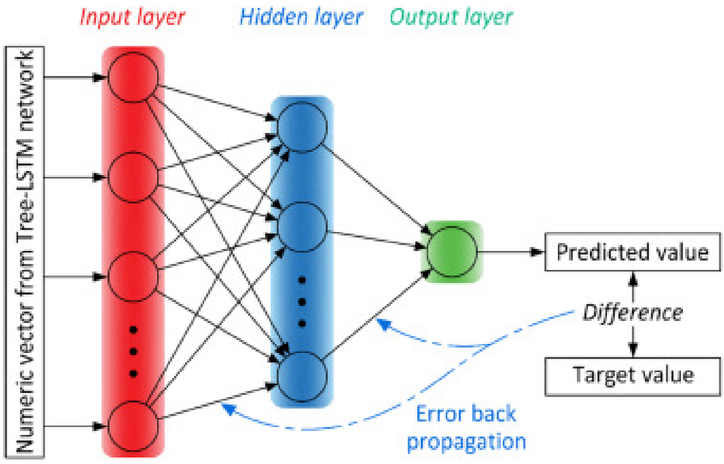


Fig. 6. The structure of the fully connected BPNN model for log *K*ow prediction.

As an open-source deep learning library for Python, PyTorch [68] mainly supported the development of the DNN model in this research. Huber loss is a common loss function which is characterized by rapid convergence and insensitiveness to outliers because it combines the advantages of two basic loss functions, that is, the mean square error and the mean absolute error. Therefore, the Huber loss [69] was adopted as the loss function in this research to evaluate the model performance during the training process. Additionally, Adam Algorithm [70] was employed for optimizing the predictive model by minimizing the loss function due to the attractive benefits that it is computationally efficient and suitable for tasks with a large dataset.

A machine learning model is parameterized, and its refers to numerous variables that can be classified into two types: model parameters and model hyper-parameters. The model parameters, such as weights and biases, are learned from the given dataset and updated with the BP algorithm by calculating the gradient of the loss function during the model training. In contrast, with the purpose of controlling the learning process efficiently, model hyper-parameters were specified before the training activates.

In order to achieve the better performance as well as make the DNN model specialize in the prediction of log Kow, hyper-parameters were specified and detailed as follows:

(i) The hidden layer of the BPNN has 32 neurons.

(ii) The batch size of the set, the number of training examples utilized in one iteration, is set as 250.

(iii) The Learning rate is set as 1.00E-03 to control the rate of convergence.

(iv) The weight decay rate is set as 1.00E-06 to alleviate the problem of the over-fitting.

The algorithm of model development with the Tree-LSTM network and BPNN is illustrated in Fig. 7. For supporting the development of the predictive model, molecular features were firstly extracted from the molecules of the collected dataset. Afterwards, the signature trees of compounds were generated for further mapping to the Tree-LSTM networks. Therefore, the vectors of molecular features can be inputted into the Tree-LSTM networks, and a vector was generated as an input for the BPNN. Within the BPNN, the properties were correlated to the molecular structures, and the QSPR model was obtained after massive training and testing. Afterwards, the QSPR model was evaluated with an external set, discussed on its Applicability Domain and compared with the reported model to investigate its performance. As such, an accurate and reliable QSPR model was generated for predicting the log K_{OW} of organic compounds.

The algorithm of model development with the Tree-LSTM network and BPNN is illustrated in Fig. 7. For supporting the development of the predictive model, molecular features were firstly extracted from the molecules of the collected dataset. Afterwards, the signature trees of compounds were generated for further mapping to the Tree-LSTM networks. Therefore, the vectors of molecular features can be inputted into the Tree-LSTM networks, and a vector was generated as an input for the BPNN. Within the BPNN, the properties were correlated to the molecular structures, and the QSPR model was obtained after massive training and testing. Afterwards, the QSPR model was evaluated with an external set, discussed on its Applicability Domain and compared with the reported model to investigate its performance. As such, an accurate and reliable QSPR model was generated for predicting the log K_{OW} of organic compounds.

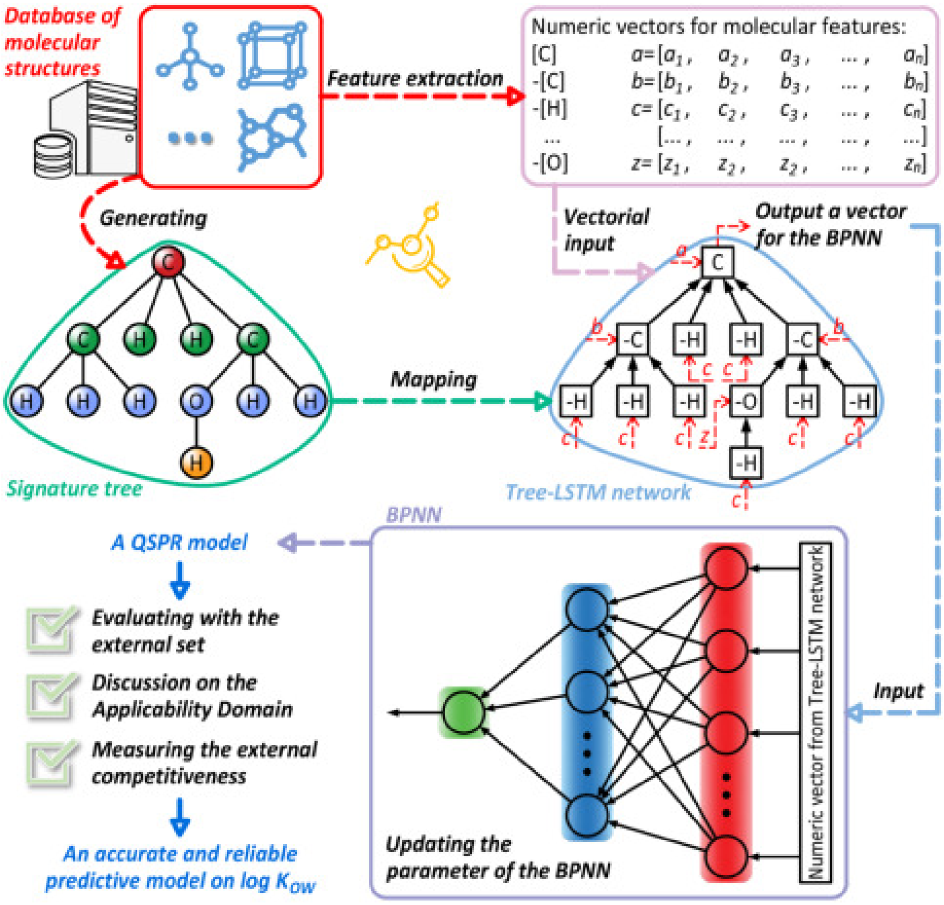


Fig. 7. The algorithm of model development with the Tree-LSTM network and BPNN.

The algorithm of the proposed model for predicting with the Tree-LSTM network and BPNN is illustrated in Fig. 8. During predicting with the developed QSPR model, the molecular structure of a new compound is used to generate the signature tree which can be mapped to the Tree-LSTM network. Afterwards, using the vectors which were generated during the model development, the Tree-LSTM network outputs a vector integrated the features of the molecular structure for the BPNN. Relying on the parameters and hyper-parameters of the BPNN determined during model development, the BPNN makes a numeric prediction and outputs a predicted value for the log K_{OW} of the compound.

development, the BPNN makes a numeric prediction and outputs a predicted value for the log Kow of the compound.

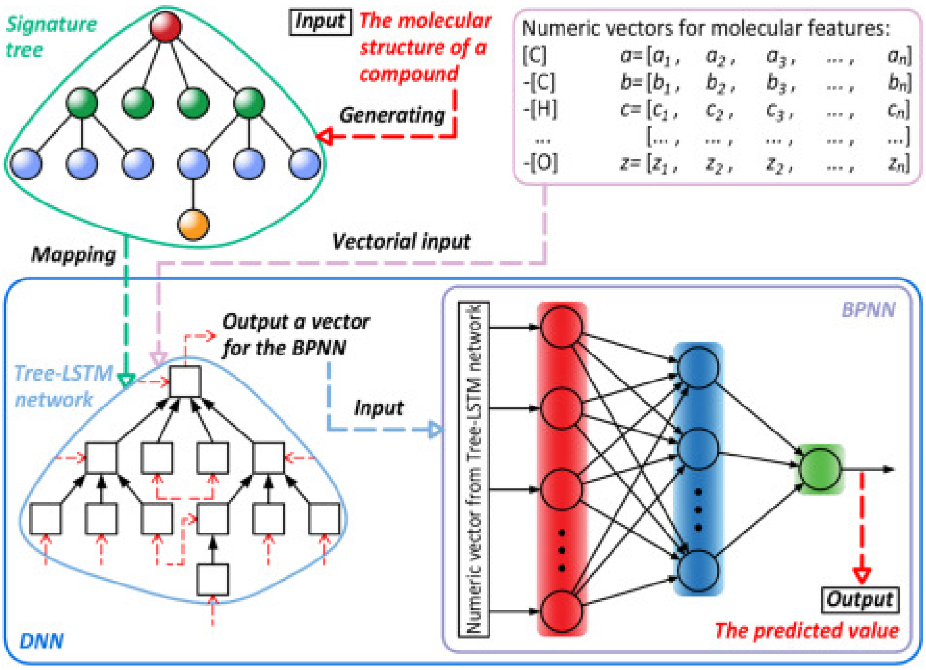


Fig. 8. The algorithm of the model for predicting with the Tree-LSTM network and BPNN.

> Read full chapter

28th European Symposium on Computer Aided Process Engineering

Gyula Dorgo, ... Janos Abonyi, in Computer Aided Chemical Engineering, 2018

Abstract

We introduce a sequence to sequence deep learning algorithm to learn and predict sequences of process alarms and warnings. The proposed recurrent neural network model utilizes an encoder layer of Long Short-Term Memory (LSTM) units to map the input sequence of discrete events into a vector of fixed dimensionality, and a decoder LSTM layer to form a prediction of the sequence of future events. We demonstrate that the information extracted by this model from alarm log databases can be used to suppress alarms with low information content which reduces the operator workload. To generate easily reproducible results and stimulate the development of alarm management algorithms we define an alarm management benchmark problem based on the simulator of a vinyl acetate production technology. The results confirm that sequence to sequence learning is a useful tool in alarm rationalization and,

in more general, for process engineers interested in predicting the occurrence of discrete events.

in more general, for process engineers interested in predicting the occurrence of discrete events.

> Read full chapter

29th European Symposium on Computer Aided Process Engineering

Jie-Jiun Chang, ... Shang-Tai Lin, in Computer Aided Chemical Engineering, 2019

4 Conclusion

The above results serve as preliminary demonstration that molecular classification and prediction of sigma-profile, results of quantum calculations, using text-based molecular descriptors is possible. A recognizer was trained using word-embedment network, and a LSTM transformation network. The network gave no false negative and very few false positive. PCA analysis showed that the transformed space can be used as for molecular feature representation and classification. Use this space as input, we showed that fairly accurate prediction of sigma-profile can be developed. Optimization of network structure have not yet been considered. The promising results suggest that extensions of this approach to a more extensive data base should be a valuable for a priori property prediction and molecular design. Refinement of neural network structure, better sampling of data and the use of other molecular representation should also be investigated.

> Read full chapter

13th International Symposium on Process Systems Engineering (PSE 2018)

Jorge Chebeir, ... Jose Romagnoli, in Computer Aided Chemical Engineering, 2018

4.2 LSTM neural network for natural gas demand prediction

Time series forecasting, at its heart, is the search for an optimal function that maps historical predictors to forecasted data. Since forecasted data is not available a priori, the algorithm is trained by dividing the historical data into past and future data. Neural network models like LSTM have been shown to outperform traditional time series methods on temporal processing tasks. LSTM's replace the neurons in traditional neural networks with cells that have the ability to retain useful information and overwrite extraneous data. In this paper, the model is built by stacking an LSTM, sigmoid and linear layer one over the other to maximally increase the accuracy. The number of nodes/layer, epochs and batch size are design parameters that were optimized using exhaustive enumeration. The input to the model comprises of predictors like the price of natural gas, crude oil price, regional population, regional temperature and past natural gas demand. Results of implementing LSTM can be observed in Fig. 3.

time series methods on temporal processing tasks. LSTM's replace the neurons in traditional neural networks with cells that have the ability to retain useful information and overwrite extraneous data. In this paper, the model is built by stacking an LSTM, sigmoid and linear layer one over the other to maximally increase the accuracy. The number of nodes/layer, epochs and batch size are design parameters that were optimized using exhaustive enumeration. The input to the model comprises of predictors like the price of natural gas, crude oil price, regional population, regional temperature and past natural gas demand. Results of implementing LSTM can be observed in Fig. 3.

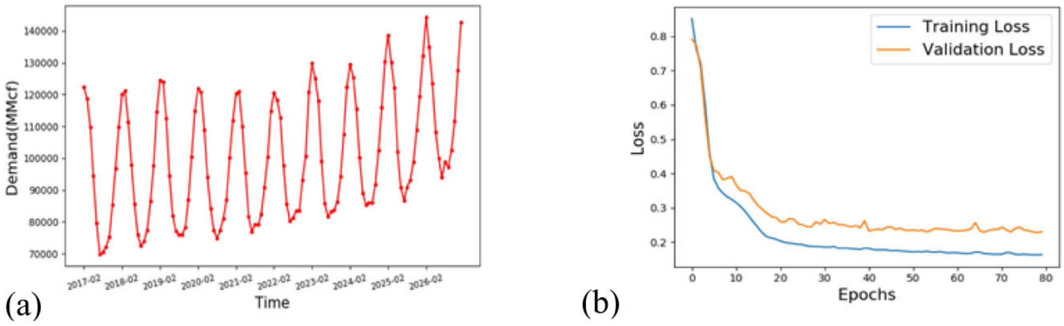


Figure 3. (a) 10-year natural gas demand forecast (b) Training and validation loss.

> Read full chapter