Article:
Liu, T., Chen, S., Yang, P. et al. (2024) Lifelong learning meets dynamic processes: an emerging streaming process prediction framework with delayed process output measurement. IEEE Transactions on Control Systems Technology, 32 (2). pp. 384-398. ISSN: 1063-6536

https://doi.org/10.1109/TCST.2023.3312850

Lifelong Learning Meets Dynamic Processes: An Emerging Streaming Process Prediction Framework with Delayed Process Output Measurement

Tong Liu, *Member, IEEE*, Sheng Chen, *Fellow, IEEE*, Po Yang, *Senior Member, IEEE*, Yunpeng Zhu, *Member, IEEE*, Mehmet Mercangöz, and Chris J. Harris

***Abstract*—As an emerging machine learning technique, lifelong learning is capable of solving multiple consecutive tasks based upon previously accumulated knowledge. Although this is highly desired for streaming process prediction in industry, lifelong learning methods have so far failed to gain applications to mainstream adaptive predictive modeling of time-varying industrial processes. This is because when faced with a new data batch, existing lifelong learning approaches need both input and output data to construct local predictors before knowledge transfer can succeed. But in many process industries, the process output data is hard to measure online and it often takes time to acquire them from off-site lab analysis. This delayed acquisition of target output data makes it challenging to apply lifelong learning and other existing adaptive mechanisms to dynamic industrial processes with delayed process output measurement. To overcome this difficulty, this paper proposes a novel lifelong learning framework that can rapidly predict new data batches with input data only before the arrival of the process output measurement. Specifically, we propose to incorporate process input information into lifelong learning via coupled dictionary learning, to enable the prediction of new batches without target output data. The input feature is linked with a local predictor through two dictionaries that are coupled by a joint sparse representation. Because of the learned coupling between the two spaces, the local predictor for the new batch can be reconstructed by knowledge transfer given only process inputs. Two industrial case studies are used to evaluate the effectiveness of our proposed framework and reveal the intrinsic learning mechanism of our lifelong process modeling to perform knowledge base adaptation.**

***Index Terms*—Lifelong learning, dynamic industrial processes, delayed output measurement, process drifts, knowledge transfer**

I. Introduction

Machine learning has taken a center stage in the recent years for scientific and engineering developments. Inspired by human intelligence, the recent advances in deep learning bring it to new heights [1], and it has been successfully applied to all walks of life, such as image recognition, natural language processing, competitive games (AlphaGo) [2], protein structure prediction (AlphaFold) [3] and short-term weather forecasting [4]. Following the success of machine learning in these areas, process industry has also begun to harvest the benefits of these breakthroughs [5]. Exploiting the availability of explosive process data, the current industrial revolution, also known as Industry 4.0, is focusing on advanced data modeling and analytics to improve control and high-level decision-making [6], [7]. Hence, adaptive and accurate modeling of industrial plants from massive and long-term process data will aid intelligent and autonomous industrial systems.

The current mainstream paradigm for industrial predictive models, which can predict the evolution of process output given process inputs, is to run machine learning algorithms on a given dataset that was historically collected from industrial plants, and to hope that the trained model will generalize well for the new data unseen in training [5], [8], [9]. In machine learning terminology, this is called isolated learning because it does not retain and accumulate knowledge learned in the past training and use it to facilitate future learning. Without the ability to accumulate knowledge from the past learning, a machine learning model typically needs a large number of training samples to learn effectively. In particular, the collected dataset needs to be informative and sufficiently rich to cover the whole dynamics of the underlying process. However, this is often not the case in practice, because many industrial systems operate in a continuous manner and generate data from their operation in the form of streams, whose state changes over time [10], [11]. This time-varying process characteristics can be caused by many factors, such as changes of raw materials and operating conditions, mechanical abrasions, and catalyst deactivation. Due to these process drifts, predictive models trained over historical dataset become obsolete, as them do not represent the newly emerged process state.

To tackle the above problem, various adaptive mechanisms are employed for industrial predictive models, to realize online adaptation with newly measured labeled samples, i.e., both process input and output samples, so as to maintain satisfactory performance over a long operating period [12]. One notable issue for this online learning is that the model adaptation needs to be performed at each sampling time with both process input and output information. This imposes two challenges. First, the online computation cost by re-estimating model structure and parameters should be sufficiently small to be accommodated within each sampling period that is determined by control systems [13]. More importantly, the process output data should be obtained timely at each time step so that the current modeling residual can be calculated to perform the model

T. Liu and M. Mercangöz are with Department of Chemical Engineering, Imperial College London, London SW7 2AZ, UK (E-mails: tong.liu@imperial.ac.uk, m.mercangoz@imperial.ac.uk).

S. Chen and C.J. Harris are with School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK (E-mails: sqc@ecs.soton.ac.uk, chrisharris57@msn.com). S. Chen is also with Faculty of Information Science and Engineering, Ocean University of China, Qingdao 266100 China

P. Yang is with Department of Computer Science, University of Sheffield, Sheffield S1 4DP, UK (E-mail: po.yang@sheffield.ac.uk).

Y. Zhu is with School of Engineering and Material Science, Queen Mary University of London, London E1 4NS, UK (E-mail: yunpeng.zhu@qmul.ac.uk).

adaptation. The first issue has been addressed to some extent by numerous techniques, and some methods are reviewed in Subsection II-A. However, the need for label or process output data immediately at every sampling time is often infeasible in many process industries, because these output variables or labels are typically hard to measure online and they can be obtained only through off-line lab analysis [5], which may take hours. For example, it is essential to monitor the lignite moisture online for the microwave lignite drying process [14], which serves as the feedback information for real-time control of the microwave power to prevent overheating. However, measurement of lignite moisture takes time and it cannot be acquired timely at every sampling time in a closed control-loop. This causes delayed label data acquisition for process control. As a result, the unavailability of timely process output data or desired labels makes such online learning strategy impractical for streaming process prediction application.

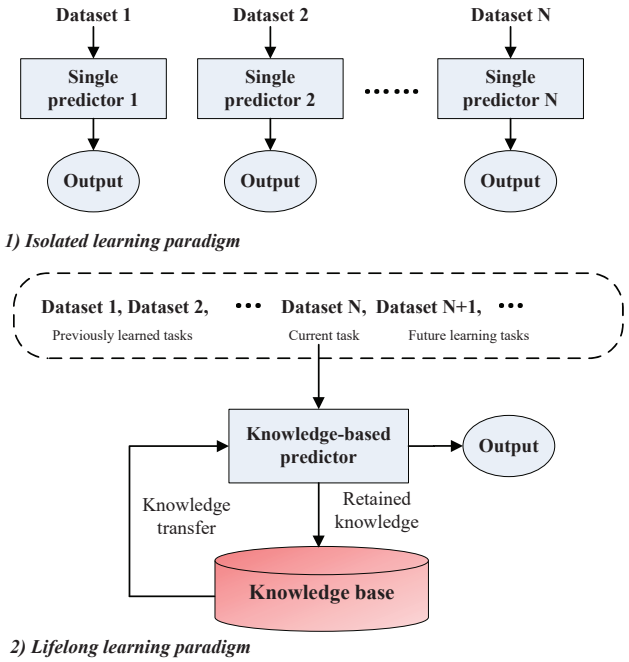


Fig. 1. Comparison between isolated learning and lifelong learning.

Different from the aforementioned isolated learning, lifelong learning was proposed to mimic the human learning ability of accumulating and maintaining knowledge learned from the past and using it seamlessly for future learning [15]. A simple comparison between the two learning paradigms is shown in Fig. 1. Isolated learning learns each dataset independently without knowledge accumulation or transfer. By contrast, lifelong learning learns consecutive new data batches based upon the previously built knowledge base (KB) as well as automatically updates the KB learned from past encountered data batches upon learning of the new batch [16], [17]. Three key characteristics of lifelong learning system, 1) continuous learning ability, 2) knowledge accumulation and maintenance by KB adaptation, and 3) knowledge transfer to facilitate future learning [18]–[21], makes this emerging technique suitable to deal with long-term data with changing dynamic characteristics, which is the case for streaming process prediction. Hence, by online learning of the predictive models on a batch-by-batch base, lifelong learning has some advantage for streaming process prediction application.

In the lifelong learning community, the efficient lifelong learning algorithm (ELLA) is one of the most popular methods [22]. With the assumption that local models of multiple related tasks share a common knowledge library, ELLA learns new tasks by selectively transferring knowledge from the KB and refining the KB over time to incorporate new knowledge learned from current tasks. This framework has been demonstrated to be effective for handling lifelong regression, classification and decision-making problems, such as image recognition [22], [23] and engineering system control [24]–[27]. However, when faced with a new task or batch, ELLA needs both input and targeted output data to construct task model before knowledge transfer can succeed [22]. As mentioned before, for many process industries, true process output data are hard to measure online and it often takes time to obtain them from off-line lab analysis. Because of this need for output data in constructing task models for new batches, ELLA cannot be applied directly to streaming process prediction with delayed process output data acquisition. This motivates us to investigate a new lifelong learning framework, which is capable of adaptively and accurately predicting new data batches with input data only by knowledge transfer before observing the true process output.

Motivated by the above background, this paper proposes a novel lifelong learning framework that makes full use of process input information to enable predicting consecutive batches of process data with delayed output measurement. Specifically, we encode input data into a feature vector that contains essential process operating information, and treat these input features as side information to augment local predictors on each batch data. To enable knowledge transfer between two spaces, we use coupled dictionary learning to connect the input's feature space with the predictor's parameter space, where the two spaces are linked through two dictionaries that are coupled by the same sparse coding. Because of the learned coupling between the two spaces, the lifelong learner is capable of rapidly reconstructing local predictors for the new coming batches given only process inputs. This capacity of 'learning without targeted output' is very important for lifelong process modeling, because the output data is often hard to obtain immediately and the learner requires to quickly make prediction by knowledge transfer before observing true process output. Two industrial case studies, involving a penicillin fermentation process and a wastewater treatment plant, are used to demonstrate the effectiveness of our proposed framework. Most importantly, we reveal the intrinsic learning mechanism of this new lifelong process modeling, and demonstrate how our method deal with process drifts by KB adaptation. In summary, our novel contributions are listed below.

1) We define the lifelong learning or modeling problem for dynamic industrial processes for the first time, and propose a novel lifelong learning framework that fully considers the key characteristics of process drifts and delayed process output measurement.
2) Based on coupled dictionary learning, we incorporate process input information into lifelong learning that uses

a factorized representation of the learned knowledge to facilitate transfer and improve predictive performance.
3) We show that our method is able to accurately predict new data batches using only process inputs through unsupervised knowledge transfer.
4) Two industrial case studies are carried out to demonstrate its effectiveness, and we reveal the intrinsic learning mechanism based on prior process knowledge.

The rest of this paper is organized as follows. Section II summarizes the related works. Section III reviews lifelong learning and presents its challenge in application to industrial processes. Section IV details the proposed lifelong learning based streaming process prediction framework, and Section V evaluates its effectiveness with two industrial case studies. Section VI concludes the paper with remarks for future works.

II. Related Works

A. Streaming Modeling of Dynamic Processes

For streaming or online modeling of time-varying dynamic processes, the key is to update the predictive model's structure and parameters to track the changing system dynamics. A popular method widely used in practice is multiple local model learning strategy [28], [29]. By partitioning the overall modeling space into multiple local subspaces, a set of local models can be constructed to capture the overall process characteristics. Based on this principle, the selective ensemble based multiple local model learning enables automatically identifying newly emerged process patterns by growing the local model set online [30]. To reduce computation burden, the growing and pruning selective ensemble regression can not only learn new process patterns but also discard outdated patterns by pruning unwanted local models [31], [32]. However, the local model adaptation for this strategy requires both process input and output data. When acquisition of output data is delayed, the local model adaptation cannot take place, and online prediction has to rely on the existing local model set given input data, i.e., it reduces to a nonadaptive model.

Inspired by gain scheduling [33], [34], another locally linear regression partitions the training input space into multiple subspaces and constructs a local linear model for each region. During inference or online prediction, appropriate local models from the trained model set are selected or combined based on input data using switching or ensemble mechanisms [35], [36]. However, this locally linear regression is a nonadaptive model. During online operation, it cannot adapt the local linear model set even both process input and output are available. The success of this locally linear regression therefore heavily depends on sufficiently rich training data to determine all the number and boundaries of subspaces of the underlying process in the training phase. If the process operates in real-time with time-varying data streams, the online prediction performance of this nonadaptive model may degrade considerably.

Another online modeling strategy is based on global model learning. A typical approach is to adopt radial basis function (RBF) network [37], [38]. During online model adaptation, the output weights of the RBF network are updated by recursive least square (RLS) to track the time-varying characteristics [39]. To further enhance the adaptability to nonstationary data, the fast tunable gradient RBF model adjusts both hidden node structure and output weights to capture newly emerged process patterns [40], [41]. This efficient online tracker is further combined with deep learning technique for high-dimensional nonstationary process modeling in [42]. Although this method is very effective and efficient for online process tracking, it needs both input and output information for model adaptation at each sampling time. This becomes impractical again when the acquisition of process output is seriously delayed. It can be seen that most existing streaming modeling approaches cannot cope with delayed output measurement in process industry, and novel real-time learning framework is urgently needed to tackle this problem.

B. Lifelong Learning

The core idea of lifelong learning is to solve multiple consecutive tasks over long-time scales upon previously accumulated knowledge, and ELLA is one of the most popular approaches [22]. It factorizes the learned task models into a shared latent dictionary as the KB to facilitate knowledge transfer as tasks arrive consecutively. When new task arrives, ELLA transfers knowledge through the shared dictionary to learn new model, and refines the dictionary with the knowledge learned from the current task. By updating the dictionary over time, newly acquired knowledge is incorporated into the KB, thereby improving the performance of previously learned models. Although ELLA-based methods [22]–[27] achieve very good performance in many applications, one important requirement is the need of first gathering sufficient labeled data for the new coming tasks. This need for labeled data imposes a serious challenge for practical problems, because data annotation for every new coming task is time-consuming. Often a learner is expected to learn new task rapidly without the delay to wait for labeling task.

To mitigate this difficulty, the work [43] incorporated high-level task descriptors into lifelong reinforcement learning, and used both task descriptors and training data to model inter-task relationships. This method was extended to regression in [44], where model can be predicted given only input data and task descriptors for new task. However, this method requires domain-specific task descriptors that must accurately characterize the underlying process dynamics. For simple engineering systems, such as a robotic arm, it is possible to define a set task descriptors that accurately reflect the the system's true underlying dynamics. However, practical industrial processes are highly complex, and it is difficult if not impossible to define accurate task descriptors for them. Consequently, similar to other existing lifelong learning methods, the lifelong learning with zero-shot knowledge transfer [43], [44] cannot be applied to practical industrial processes with delayed process output measurement.

Hence, it is necessary and vital to develop new lifelong process modeling framework for industrial process applications, where labeled data for new tasks is difficult to obtain quickly. The novel contribution of this paper is to propose a new lifelong learning framework for streaming industrial

processes with delayed process output measurement. Note that unlike the lifelong learning with zero-shot knowledge transfer [43], [44], our proposed lifelong learning method is capable of performing unsupervised knowledge transfer with input data only and there is no need to first define task descriptors.

III. Lifelong Learning for Industrial Processes

A. Problem Formulation

For a streaming process, let multiple data batches be received consecutively as $\{\mathbb{D}^{(1)}, \mathbb{D}^{(2)}, \cdots, \mathbb{D}^{(T_{\max})}\}$. The predictive model must rapidly predict each new batch by building upon its previously learned knowledge, and a local predictor $f^{(t)}$ can be constructed on each batch data $\mathbb{D}^{(t)} = \{\boldsymbol{x}_i^{(t)}, y_i^{(t)}\}_{i=1}^{n_t}$, where n_t is the number of samples (batch size), $\boldsymbol{x}_i^{(t)} \in \mathbb{R}^d$ and $y_i^{(t)} \in \mathbb{R}$ are the i-th input sample and associated output or label sample, respectively, for batch t.

The generic steaming process prediction is formulated as the following framework. Let $T-1$ be the number of batches with complete process input and output data that the process has generated so far and $\{f^{(1)}, \cdots, f^{(T-1)}\}$ be the previously built local predictors. Because of delayed process output measurement, when new batch T first arrives, it contains only the process inputs $\{\boldsymbol{x}_i^{(T)}\}_{i=1}^{n_T}$. The predictor must be able to accurately predict the true process output $y_i^{(T)}$ based the process input $\boldsymbol{x}_i^{(T)}$ and the knowledge learned from the previous batches, $\{f^{(1)}, \cdots, f^{(T-1)}\}$. Only later when the true process outputs $\{y_i^{(T)}\}_{i=1}^{n_T}$ for a new batch are acquired, the predictor $f^{(T)}$ may then be constructed on the completed data batch $\mathbb{D}^{(T)} = \{\boldsymbol{x}_i^{(T)}, y_i^{(T)}\}_{i=1}^{n_T}$. Ideally, the knowledge learned from the previous batches should accelerate this model construction and the constructed $f^{(T)}$ should contribute its learned new knowledge to the learned knowledge library.

B. Revisit of ELLA

ELLA [22] learns and maintains a KB $\boldsymbol{L} \in \mathbb{R}^{d\times k}$, which forms a shared basis for all predictors and facilitates knowledge transfer between them. For each batch t, ELLA constructs a local predictor $f^{(t)}(\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta}^{(t)})$ that is parametrized by a d-dimensional batch-specific parameter vector $\boldsymbol{\theta}^{(t)}$. This model parameter is a linear combination of the columns of $\boldsymbol{L}$ using the sparse coefficients $\boldsymbol{s}^{(t)} \in \mathbb{R}^k$ as $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$. The dictionary $\boldsymbol{L}$ stores chunks of knowledge that are shared for all the batches, and the sparse code $\boldsymbol{s}^{(t)}$ extracts the relevant pieces of knowledge for a particular batch t. Hence, this model parameter factorization enables effective knowledge transfer among batches. Typically, a local linear model $f^{(t)}(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\theta}^{(t)}$ is adopted, and therefore we must have $k \leq d$. This is because in this case the columns of $\boldsymbol{L}$, i.e., the dimension of the KB, cannot exceed the dimension of the input space d. Typically, $k < d$ as some elements of $\boldsymbol{x}$ may be collinear.

Given the process inputs and outputs $\{\boldsymbol{x}_i^{(t)}, y_i^{(t)}\}_{i=1}^{n_t}$ for each batch t, ELLA optimizes the following cost function

$$\min_{\boldsymbol{L},\boldsymbol{S}} \frac{1}{T}\sum_{t=1}^{T}\left(\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) + \mu\left\|\boldsymbol{s}^{(t)}\right\|_1\right) + \lambda\left\|\boldsymbol{L}\right\|_{\mathrm{F}}^2, \tag{1}$$

where $\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) = \frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{J}\left(y_i^{(t)} - \widehat{y}_i^{(t)}\right)$ with $\mathcal{J}(\bullet)$ being the squared-loss function and $\widehat{y}_i^{(t)} = f\left(\boldsymbol{x}_i^{(t)}; \boldsymbol{\theta}^{(t)}\right)$, $\boldsymbol{S} = \left[\boldsymbol{s}^{(1)}\ \boldsymbol{s}^{(2)}\cdots\boldsymbol{s}^{(T)}\right]$ is the matrix consisting of all the sparse coefficient vectors, and the L_1 norm is used to control the sparsity of $\boldsymbol{s}^{(t)}$ with the regularization parameter μ, while $\|\bullet\|_{\mathrm{F}}$ is the Frobenius norm, which regularizes the complexity of dictionary $\boldsymbol{L}$ with the regularization parameter λ.

To solve this optimization, ELLA takes a second-order Taylor expansion to approximate the objective around an estimate $\widehat{\boldsymbol{\theta}}^{(t)}$ of the local predictor parameters for each batch $\mathbb{D}^{(t)}$, and only updates the coefficients $\boldsymbol{s}^{(t)}$ for the current batch at each time step. This process reduces the optimization (1) to the problem of sparsely coding the local predictors in the shared dictionary $\boldsymbol{L}$, and it enables solving $\boldsymbol{L}$ and $\boldsymbol{S}$ efficiently by the following recursive updating rules [22]:

$$\boldsymbol{s}^{(t)} = \arg\min_{\boldsymbol{s}} \left\|\widehat{\boldsymbol{\theta}}^{(t)} - \boldsymbol{L}\boldsymbol{s}\right\|_{\boldsymbol{\Gamma}^{(t)}}^2 + \mu\left\|\boldsymbol{s}\right\|_1, \tag{2}$$

$$\boldsymbol{A} = \boldsymbol{A} + \left(\boldsymbol{s}^{(t)}\left(\boldsymbol{s}^{(t)}\right)^{\mathrm{T}}\right)\otimes\boldsymbol{\Gamma}^{(t)}, \tag{3}$$

$$\boldsymbol{b} = \boldsymbol{b} + \mathrm{vec}\left[\boldsymbol{s}^{(t)}\otimes\left(\left(\widehat{\boldsymbol{\theta}}^{(t)}\right)^{\mathrm{T}}\boldsymbol{\Gamma}^{(t)}\right)\right], \tag{4}$$

$$\boldsymbol{L} = \boldsymbol{L} + \mathrm{mat}\left[\left(\frac{1}{T}\boldsymbol{A} + \lambda\boldsymbol{I}_{(kd)}\right)^{-1}\frac{1}{T}\boldsymbol{b}\right]_{d\times k}, \tag{5}$$

where $\|\boldsymbol{v}\|_{\boldsymbol{A}}^2 = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}$, the elements of $\boldsymbol{L}$ are initialized by taking values randomly from $(0,\ 1)$, and $\boldsymbol{\Gamma}^{(t)} = \boldsymbol{\Gamma}\left(\widehat{\boldsymbol{\theta}}^{(t)}\right)$ is the Hessian matrix of the loss $\mathcal{J}\left(\widehat{\boldsymbol{\theta}}^{(t)}\right)$, while $\otimes$ denotes the Kronecker product, $\boldsymbol{A} \in \mathbb{R}^{(kd)\times(kd)}$ is initialized to the all zero-elements matrix, and $\boldsymbol{b} \in \mathbb{R}^{kd}$ is initialized to the all zero-elements vector. Furthermore, the vector stacking operator vec$[\bullet]$ stacks the columns of matrix one by one to form a vector, $\boldsymbol{I}_{(kd)}$ is the $(kd)\times(kd)$ identity matrix, and the matrix forming operator mat$[\bullet]_{d\times k}$ converts a (dk)-dimensional vector into a $(d\times k)$-dimensional matrix.

Remark 1: The theoretical justification of using L_1 and L_2 norms to regularize the model complexity is well understood in machine learning. In particular, imposing an L_1 norm on the model parameters has the desired property of enforcing the sparsity of the model, i.e., making many parameters to zero. Analysis of ELLA is also widely available in the literature. For example, the convergence analysis of ELLA, (2) to (5), to the solution of the optimization (1) can be found in [22].

Remark 2: The dictionary size or the sparsity level k is an important hyper parameter of ELLA, which is obviously problem dependent. Ideally, it would be highly desirable to be able to determine the value of k from data. For the single task learning, this is indeed achievable as the L_1 norm of the encoding vector in the optimization naturally enforces sparsity and automatically makes k smaller than d according to the underlying data structure. However, ELLA considers the multiple tasks, and it is unknown to us how to automatically determine an appropriate value for k from data. Therefore, the dictionary size k is typically chosen empirically as in [22].

C. Challenge for Applying ELLA to Streaming Processes

As can be seen from the updating rules (2) to (5), each time when a new batch t arrives, ELLA requires first to estimate

an initial local predictor $\widehat{\boldsymbol{\theta}}^{(t)}$ before it can update $\boldsymbol{s}^{(t)}$ and $\boldsymbol{L}$. The updated sparse coding vector $\boldsymbol{s}^{(t)}$ and dictionary $\boldsymbol{L}$ can then be utilized to construct the new predictor to predict the true process outputs for new batch t. In other words, the new local predictor can only be constructed if at least some new batch data contains both the process inputs and corresponding process outputs. However, for many streaming processes, it often takes a long time to acquire process output data. When encountering a new batch at first glance, typically only the process input data are available for making predictions. This imposes a serious challenge for applying existing lifelong models to streaming processes with delayed process output measurement, since they need sufficient labeled data for new tasks to start building new predictors for prediction.

In order to eliminate this need for labeled data for predicting new batches and hence make the lifelong learning better suited for industrial process prediction, we propose a novel lifelong learning framework that makes full use of process inputs to enable unsupervised knowledge transfer on predicting new batches. Specifically, upon learning a few batches with complete process input and output data, future local predictors for new batches can be constructed given only input data. It is worth emphasizing again that our scheme is completely novel and it is very different from the scheme of [43], [44], which needs input data as well as task descriptors to construct local predictors for new batches. For a complex industrial process, it is impossible to craft its task descriptors.

IV. Proposed Streaming Process Prediction Framework

Our proposed industrial lifelong learning system is depicted in Fig. 2. The industrial system operates in real-time to consecutively generate multiple data batches in the form of streams. For each batch of data, we define the process input matrix $\boldsymbol{X}^{(t)} = \left[\boldsymbol{x}_1^{(t)}\ \boldsymbol{x}_2^{(t)} \cdots \boldsymbol{x}_{n_t}^{(t)}\right] \in \mathbb{R}^{d\times n_t}$ and the corresponding desired output vector $\boldsymbol{y}^{(t)} = \left[y_1^{(t)}\ y_2^{(t)} \cdots y_{n_t}^{(t)}\right]^{\mathrm{T}} \in \mathbb{R}^{n_t}$. As a new batch arrives, knowledge accumulated from the previous batches is selectively transferred to predict the new batch, and newly acquired information from the current batch is stored in the KB for future use. In order to achieve knowledge transfer on new batch prediction without output data, we propose to incorporate input features into lifelong modeling framework via coupled dictionary learning, enabling input features and the local predictor to augment each other. For historical batches with complete process input and output data, the local predictor is constructed, while the input features are encoded only by process inputs. To link two feature spaces, we employ two dictionaries that act as knowledge repositories for the two spaces, and they are coupled by a joint sparse representation. Because of the learned coupling, the local predictor for a new coming batch can be reconstructed given only the process inputs. This capacity of learning new predictors without targeted output is particularly suitable for streaming industrial process prediction, where acquisition of true process output data may encounter long delay. We now detail each part of our proposed framework.

A. Learning from Historical Batches

1) Local predictor: For historical batch with complete process input and output data $\left(\boldsymbol{X}^{(t)}, \boldsymbol{y}^{(t)}\right)$, a local model $f(\boldsymbol{X};\boldsymbol{\theta}) = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{\theta}$ is constructed. Specifically, the parameter vector of the local model is computed using the regularized least square (LS) estimator as

$$\widehat{\boldsymbol{\theta}}^{(t)} = \left(\boldsymbol{X}^{(t)}\left(\boldsymbol{X}^{(t)}\right)^{\mathrm{T}} + \beta \boldsymbol{I}_d\right)^{-1}\boldsymbol{X}^{(t)}\boldsymbol{y}^{(t)}, \tag{6}$$

where β is a small positive regularization parameters, e.g., $\beta = 10^{-6}$. The Hessian $\boldsymbol{\Gamma}^{(t)}$ of the squared-loss function $\mathcal{J}(\boldsymbol{\theta}^{(t)})$ around the single task solution $\widehat{\boldsymbol{\theta}}^{(t)}$ is given by

$$\boldsymbol{\Gamma}^{(t)} = \frac{1}{2n_t}\left(\boldsymbol{X}^{(t)}\left(\boldsymbol{X}^{(t)}\right)^{\mathrm{T}} + \beta \boldsymbol{I}_d\right). \tag{7}$$

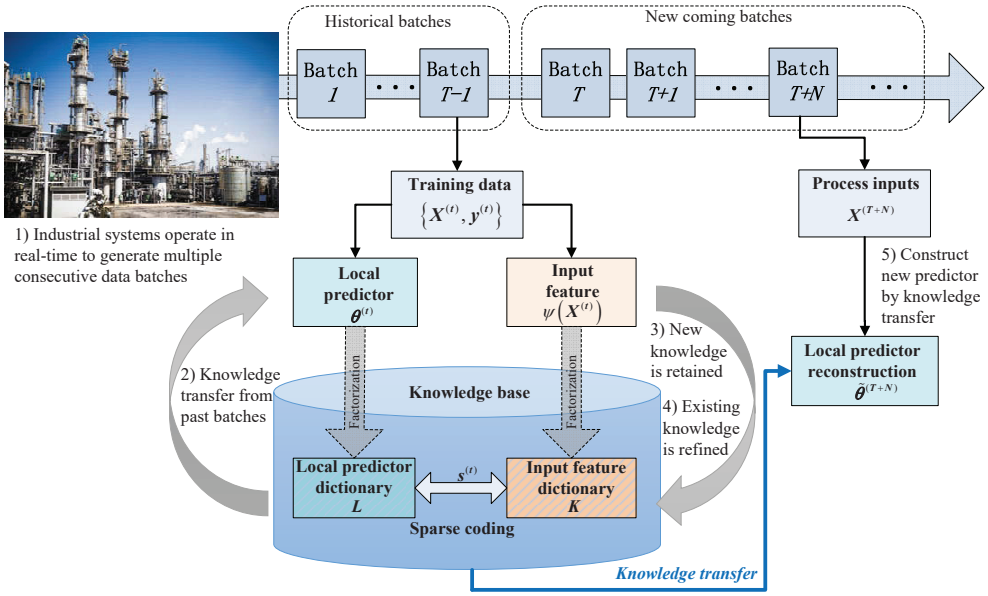


Fig. 2. Illustration of industrial lifelong learning system. For historical batches, the process inputs are encoded as feature vector while both input and output data are used to construct a local predictor. The input features and local predictor are factorized into two dictionaries that are coupled by a joint sparse coding. When new batch arrives, the learner reconstructs local predictor for new coming batch using solely process inputs by knowledge transfer.

Hence for historical data batches, we first compute the predictors parameter vectors $\widehat{\boldsymbol{\theta}}^{(t)}$ and Hessian matrices $\boldsymbol{\Gamma}^{(t)}$ before performing knowledge transfer in the learning process.

2) Input feature: In the process industry, the process inputs also known as operational data $\boldsymbol{X}^{(t)}$ are often easy and quick to acquire online from sensors directly, while the process output data $\boldsymbol{y}^{(t)}$ are difficult to obtain by online measurement and they typically take long time to acquire from off-line lab analysis. In such cases, process output data experience a delayed acquisition, which makes supervised modeling impractical. Although input data itself cannot be used to construct a predictor, it also contains essential process operational information [45]. It is highly beneficial to use the input data for unsupervised learning to supplement the supervised modeling, thereby using it as a 'backup' to predict new batch when desired process output data is unavailable for constructing the predictor timely.

To incorporate process inputs alone into the learning procedure, the input data matrix $\boldsymbol{X}^{(t)} \in \mathbb{R}^{d\times n_t}$ is transformed into a d-dimensional feature vector that can link with the predictor's parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$. To link these two spaces, we can transform $\boldsymbol{X}^{(t)}$ into the d-dimensional feature vector $\psi\big(\boldsymbol{X}^{(t)}\big)$, where $\psi(\bullet)$ is an operator that encodes a matrix into a vector. The simplest encoding is the direction of the mean value in each row of $\boldsymbol{X}^{(t)}$. Expressing the i-th column of $\boldsymbol{X}^{(t)}$ as $\boldsymbol{x}_i^{(t)} = \big[x_{1,i}^{(t)}\ x_{2,i}^{(t)} \cdots x_{d,i}^{(t)}\big]^{\mathrm{T}}$, we have

$$\psi\big(\boldsymbol{X}^{(t)}\big) = \frac{\big[\bar{x}_1^{(t)}\ \bar{x}_2^{(t)} \cdots \bar{x}_d^{(t)}\big]^{\mathrm{T}}}{\Big\|\big[\bar{x}_1^{(t)}\ \bar{x}_2^{(t)} \cdots \bar{x}_d^{(t)}\big]\Big\|_2} = \bar{\boldsymbol{x}}^{(t)} \in \mathbb{R}^d, \tag{8}$$

where $\bar{x}_j^{(t)} = \frac{1}{n_t}\sum_{i=1}^{n_t} x_{j,i}$, $1 \le j \le d$. Hence, $\bar{\boldsymbol{x}}^{(t)}$ are the input features for batch t. We next show how to link input features with local predictor via coupled dictionary learning.

3) Coupled Dictionary Learning: The idea of coupled dictionary learning was used in the scheme [43], [44] to link the high-level task descriptions with the learned model to achieve knowledge transfer for new tasks [43], [44]. Similarly, we employ the coupled dictionaries to link the local predictor's space with the input features' space, so as to fully exploit process input information and achieve predicting a new batch without the need for process output data.

Recall that the lifelong learning approach factorizes the predictor parameters $\boldsymbol{\theta}^{(t)}$ for each task as a sparse linear combination of a shared dictionary by $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$, where each column of $\boldsymbol{L}$ represents a cohesive chunk of knowledge [22]. The sparse coefficient vectors $\boldsymbol{S}$ encode the local predictors in the shared dictionary, providing an embedding of the batches based on how their predictors share knowledge.

In the same way, the input feature vector $\bar{\boldsymbol{x}}^{(t)}$ can also be linearly factorized using a shared dictionary $\boldsymbol{K} \in \mathbb{R}^{d\times k}$ over the process input's space. $\boldsymbol{K}$ has a similar function with $\boldsymbol{L}$, which captures the relationships among the input features for different batches. In order to link the two spaces, we enforce the two dictionaries, $\boldsymbol{L}$ and $\boldsymbol{K}$, to share the same sparse coefficient vectors $\boldsymbol{S}$ so as to reconstruct both the local predictors and the input features. Hence, for batch t,

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)},\ \bar{\boldsymbol{x}}^{(t)} = \boldsymbol{K}\boldsymbol{s}^{(t)}. \tag{9}$$

Because the two dictionaries are enforced to have the same sparse code $\boldsymbol{s}^{(t)}$, the relevant pieces of information for the local predictor becomes coupled with its associated input features. To optimize the coupled dictionaries $\boldsymbol{L}$ and $\boldsymbol{K}$, the objective (1) is reformulated for the coupled dictionaries as

$$\begin{aligned}\min_{\boldsymbol{L},\boldsymbol{K},\boldsymbol{S}} \frac{1}{T}\sum_{t=1}^{T}\Big(\mathcal{J}\big(\boldsymbol{\theta}^{(t)}\big) + \rho\big\|\bar{\boldsymbol{x}}^{(t)} - \boldsymbol{K}\boldsymbol{s}^{(t)}\big\|_2^2 + \mu\big\|\boldsymbol{s}^{(t)}\big\|_1\Big) \\ + \lambda\left(\|\boldsymbol{L}\|_{\mathrm{F}}^2 + \|\boldsymbol{K}\|_{\mathrm{F}}^2\right),\end{aligned} \tag{10}$$

where the parameter ρ balances the local predictor's fit to the input feature's fit.

To solve the optimization (10) in a lifelong learning setting, $\mathcal{J}\big(\boldsymbol{\theta}^{(t)}\big)$ is approximated by a second-order Taylor expansion around the regularized LS estimate $\widehat{\boldsymbol{\theta}}^{(t)}$ given in (6). That is, we expand $\mathcal{J}\big(\boldsymbol{\theta}^{(t)}\big)$ around $\widehat{\boldsymbol{\theta}}^{(t)}$ for each batch as

$$\begin{aligned}\mathcal{J}\big(\boldsymbol{\theta}^{(t)}\big) \approx &\mathcal{J}\big(\widehat{\boldsymbol{\theta}}^{(t)}\big) + \triangledown\mathcal{J}\big(\widehat{\boldsymbol{\theta}}^{(t)}\big)\big(\boldsymbol{\theta}^{(t)} - \widehat{\boldsymbol{\theta}}^{(t)}\big) \\ &+ \frac{1}{2}\Big\|\boldsymbol{\theta}^{(t)} - \widehat{\boldsymbol{\theta}}^{(t)}\Big\|_{\boldsymbol{\Gamma}^{(t)}}^2,\end{aligned} \tag{11}$$

where $\triangledown$ denotes the gradient operator. The first term $\mathcal{J}\big(\widehat{\boldsymbol{\theta}}^{(t)}\big)$ is a constant and can be omitted. Since $\boldsymbol{\theta}^{(t)}$ is the minimizer of the objective $\mathcal{J}\big(\boldsymbol{\theta}^{(t)}\big)$, $\triangledown\mathcal{J}\big(\widehat{\boldsymbol{\theta}}^{(t)}\big)$ is zero, and the second term can also be removed. Thus, the loss function $\mathcal{J}\big(\boldsymbol{\theta}^{(t)}\big)$ is approximated by the last term of (11), which can be rewritten as $\Big\|\widehat{\boldsymbol{\theta}}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\Big\|_{\boldsymbol{\Gamma}^{(t)}}^2$, given $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$. With the approximation (11), the optimization (10) is simplified as

$$\begin{aligned}\min_{\boldsymbol{L},\boldsymbol{K},\boldsymbol{S}} \frac{1}{T}\sum_{t=1}^{T}\Bigg(\Big\|\widehat{\boldsymbol{\theta}}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\Big\|_{\boldsymbol{\Gamma}^{(t)}}^2 + \rho\Big\|\bar{\boldsymbol{x}}^{(t)} - \boldsymbol{K}\boldsymbol{s}^{(t)}\Big\|_2^2 \\ + \mu\Big\|\boldsymbol{s}^{(t)}\Big\|_1\Bigg) + \lambda\big(\|\boldsymbol{L}\|_{\mathrm{F}}^2 + \|\boldsymbol{K}\|_{\mathrm{F}}^2\big).\end{aligned} \tag{12}$$

Further defining:

$$\boldsymbol{\Theta}^{(t)} = \begin{bmatrix}\widehat{\boldsymbol{\theta}}^{(t)} \\ \bar{\boldsymbol{x}}^{(t)}\end{bmatrix},\ \boldsymbol{H} = \begin{bmatrix}\boldsymbol{L} \\ \boldsymbol{K}\end{bmatrix},\ \boldsymbol{\Psi}^{(t)} = \begin{bmatrix}\boldsymbol{\Gamma}^{(t)} & \boldsymbol{0}_{d\times d} \\ \boldsymbol{0}_{d\times d} & \rho\boldsymbol{I}_d\end{bmatrix}, \tag{13}$$

where $\boldsymbol{0}_{d\times d}$ is the $d \times d$ zero matrix, the optimization (12) can be rewritten in a concise form as

$$\min_{\boldsymbol{H},\boldsymbol{S}} \frac{1}{T}\sum_{t=1}^{T}\left(\Big\|\boldsymbol{\Theta}^{(t)} - \boldsymbol{H}\boldsymbol{s}^{(t)}\Big\|_{\boldsymbol{\Psi}^{(t)}}^2 + \mu\Big\|\boldsymbol{s}^{(t)}\Big\|_1\right) + \lambda\|\boldsymbol{H}\|_{\mathrm{F}}^2. \tag{14}$$

Clearly the optimization (14) has the identical form to (1), and it can be solved in the same way. Note that (14) can be decoupled into the two optimization problems of similar form on $\boldsymbol{L}$ and $\boldsymbol{K}$, respectively. Hence the two dictionaries can be updated independently.

When batch t arrives, three steps are performed to update the lifelong learning model, namely, compute $\boldsymbol{s}^{(t)}$, update $\boldsymbol{L}$ and update $\boldsymbol{K}$. The sparse vector $\boldsymbol{s}^{(t)}$ is first computed using the current basis $\boldsymbol{H}$ by solving the following L_1-regularized

regression problem, which is a Lasso:

$$\boldsymbol{s}^{(t)} = \arg\min_{\boldsymbol{s}} \left\| \boldsymbol{\Theta}^{(t)} - \boldsymbol{H}\boldsymbol{s}^{(t)} \right\|_{\boldsymbol{\Psi}^{(t)}}^{2} + \mu \left\| \boldsymbol{s}^{(t)} \right\|_{1}. \quad (15)$$

After $\boldsymbol{s}^{(t)}$ is obtained, the two dictionaries, $\boldsymbol{L}$ and $\boldsymbol{K}$, are calculated independently by the recursive updating equations (3) to (5). In particular, to update the dictionary $\boldsymbol{K}$, we simply replace $\boldsymbol{\Gamma}^{(t)}$ by $\rho\boldsymbol{I}_d$, $\widehat{\boldsymbol{\theta}}^{(t)}$ by $\bar{\boldsymbol{x}}^{(t)}$ and $\boldsymbol{L}$ by $\boldsymbol{K}$ in (3) to (5).

B. Predicting a New Batch by Knowledge Transfer

The main advantage of linking the local predictor and the input features is that we can construct the local predictor for new batches using only process input data, which is valuable for streaming processes. This capability of unsupervised knowledge transfer is enabled by the coupled dictionary learning, which allows us to use input features to recover the local predictor through coupled dictionaries and sparse coding.

Given the process input data $\boldsymbol{X}^{(T)}$ for new batch T, we first encode $\boldsymbol{X}^{(T)}$ as the feature vector $\bar{\boldsymbol{x}}^{(T)} = \psi\big(\boldsymbol{X}^{(T)}\big)$, and then estimate the sparse coding in the input feature space via Lasso based on the previously learned dictionary $\boldsymbol{K}$

$$\widehat{\boldsymbol{s}}^{(T)} = \arg\min_{\boldsymbol{s}} \left\| \bar{\boldsymbol{x}}^{(T)} - \boldsymbol{K}\boldsymbol{s} \right\|_{2}^{2} + \mu \left\| \boldsymbol{s} \right\|_{1}. \quad (16)$$

Since this estimated $\widehat{\boldsymbol{s}}^{(T)}$ also serves as the sparse coding for the latent dictionary $\boldsymbol{L}$, it can be used to recover the local predictor for new batch T as

$$\widetilde{\boldsymbol{\theta}}^{(T)} = \boldsymbol{L}\widehat{\boldsymbol{s}}^{(T)}. \quad (17)$$

This new local predictor $\widetilde{\boldsymbol{\theta}}^{(T)}$ is obtained with the input data $\boldsymbol{X}^{(T)}$ only, and it can then be utilized to predict the true process outputs for new batch T as $\widehat{\boldsymbol{y}}^{(T)} = \big(\boldsymbol{X}^{(T)}\big)^{\mathrm{T}}\widetilde{\boldsymbol{\theta}}^{(T)}$. This completely eliminates the need to wait for output data to construct a model.

It can be seen from (16) and (17) that the construction of a new local predictor depends on the previously built dictionary $\boldsymbol{L}$ and the current input features $\bar{\boldsymbol{x}}^{(T)}$. $\boldsymbol{L}$ contains the knowledge learned from all the past batches, while the input features contain the latest process operation information. Combining both $\boldsymbol{L}$ and $\bar{\boldsymbol{x}}^{(T)}$ can enhance the predictive performance of the new model. This is another advantage of the proposed unsupervised knowledge transfer.

Remark 3: Based on the coupled dictionary learning, the premise of using the learned dictionary $\boldsymbol{K}$ and input features to reconstruct the new predictor is that $\boldsymbol{L}$ and $\boldsymbol{K}$ are closely related. Recall that we factorize the local predictor's parameters and input features as $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$ and $\bar{\boldsymbol{x}}^{(t)} = \boldsymbol{K}\boldsymbol{s}^{(t)}$, respectively. The dictionary $\boldsymbol{L}$ is basically extracted from the previously learned local predictors and hence it captures the inner characteristics of the underlying system or the relationship between process input and output. If we assume that $\boldsymbol{K}$ and $\boldsymbol{L}$ contain similar knowledge of the process, the input features given in Subsection IV-A2 must also characterize the underlying system dynamics as well. In other words, the predictor's parameter is an implicit function of the process input for individual batches. If the process characteristics were completely independent of the process input, the use of input features and dictionary $\boldsymbol{K}$ would not be able to reconstruct an accurate predictor. However, this cannot be the case in practice. This is because the output is always related to the input and hence, the process input is closely related to the system characteristics. Since input features contain essential process operation knowledge, the integration of the latest operating information into the previously built KB can enhance the accuracy of new predictor construction.

C. Algorithm Summary

The proposed lifelong learning framework for industrial process prediction is summarized in Algorithm 1, which operates naturally in three stages, namely, **initial training**, **online prediction**, and **knowledge base adaptation**.

1) Initial training: Multiple historical batches with complete process input and output data are collected and used to build the KB for the lifelong learner. After supervised training, the trained KB, $\boldsymbol{L}$ and $\boldsymbol{K}$, are kept.

2) Online prediction: When a new data batch with process input data only arrives, the lifelong learner first constructs a new local predictor based on the trained KB and input data by unsupervised knowledge transfer, and then makes the prediction for the new data patch.

Algorithm 1 Lifelong learning based streaming process prediction

1: **Initial Training**
2: Collect historical data batches $\big\{\mathbb{D}^{(1)}, \cdots, \mathbb{D}^{(T-1)}\big\}$, determine dictionary size k, regularization parameters μ and λ, balance coefficient ρ, and initialize $\boldsymbol{L}$ and $\boldsymbol{K}$.
3: **for** $(t = 0; t \leq T-1; t = t+1)$ **do**
4: Construct local predictor on $\big\{\boldsymbol{X}^{(t)}, \boldsymbol{y}^{(t)}\big\}$ by computing model parameter $\widehat{\boldsymbol{\theta}}^{(t)}$ and Hessian $\boldsymbol{\Gamma}^{(t)}$ using (6) and (7).
5: Encode $\boldsymbol{X}^{(t)}$ into feature vector $\bar{\boldsymbol{x}}^{(t)}$ using (8).
6: Construct matrices $\boldsymbol{\Theta}^{(t)}$, $\boldsymbol{H}$ and $\boldsymbol{\Psi}^{(t)}$ of (13).
7: Solve sparse coding $\boldsymbol{s}^{(t)}$ by Lasso of (15).
8: $\boldsymbol{L} \leftarrow \text{update}\, L\big(\boldsymbol{L}, \boldsymbol{s}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}, \boldsymbol{\Gamma}^{(t)}, \lambda\big)$ by (3)-(5).
9: $\boldsymbol{K} \leftarrow \text{update}\, K\big(\boldsymbol{K}, \boldsymbol{s}^{(t)}, \bar{\boldsymbol{x}}^{(t)}, \rho\boldsymbol{I}_d, \lambda\big)$ by (3)-(5).
10: **end for**
11: **Online Prediction**
12: When new batch with only process input data arrives, set $t = t+1$.
13: Encode $\boldsymbol{X}^{(t)}$ into feature vector $\bar{\boldsymbol{x}}^{(t)}$) using (8).
14: Solve sparse coding $\boldsymbol{s}^{(t)}$ by Lasso of (16).
15: Recover local predictor by computing its parameter vector $\widetilde{\boldsymbol{\theta}}^{(T)}$ using (17).
16: Predict true process output by $\widehat{\boldsymbol{y}}^{(t)} = \big(\boldsymbol{X}^{(t)}\big)^{\mathrm{T}}\widetilde{\boldsymbol{\theta}}^{(t)}$.
17: **Knowledge Base Adaptation**
18: After observing the true process output $\boldsymbol{y}^{(t)}$, construct local predictor, and compute model parameter $\widehat{\boldsymbol{\theta}}^{(t)}$ and Hessian $\boldsymbol{\Gamma}^{(t)}$ using (6) and (7).
19: Construct matrices $\boldsymbol{\Theta}^{(t)}$, $\boldsymbol{H}$, and $\boldsymbol{\Psi}^{(t)}$ of (13).
20: Solve sparse coding $\boldsymbol{s}^{(t)}$ by Lasso of (15).
21: $\boldsymbol{L} \leftarrow \text{update}\, L\big(\boldsymbol{L}, \boldsymbol{s}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}, \boldsymbol{\Gamma}^{(t)}, \lambda\big)$ by (3)-(5).
22: $\boldsymbol{K} \leftarrow \text{update}\, K\big(\boldsymbol{K}, \boldsymbol{s}^{(t)}, \bar{\boldsymbol{x}}^{(t)}, \rho\boldsymbol{I}_d, \lambda\big)$ by (3)-(5).
23: Return to **line 12**

3) KB adaptation: After observing the true output data, the completed current batch data are used to update the KB so as to acquire the most up-to-data knowledge.

It is worth recapping that the classic ELLA [22] cannot be applied to this industrial streaming process prediction application. This is because in order to perform online prediction for new batches, ELLA first requires sufficient labeled data to identify task relationships before bootstrapping a model via knowledge transfer. However, the labeled data are not available for online prediction. Also the scheme of [43], [44] cannot be applied to this industrial streaming process prediction application, since it is generally impossible to craft high-level task descriptors for a practical industrial process. By contrast, our framework constructs predictors for predicting new batches by unsupervised knowledge transfer based on process input information only. Additionally, we employ the online KB adaptation in our framework because the process characteristics can change significantly. It is vita to capture the latest data characteristics in the KB. This is the elegance of lifelong learning. Hence, our proposed lifelong learning framework is very suitable for dynamic process prediction and modeling particularly when the acquisition of process output data is seriously delayed.

We now analyze the online computational complexity of learning new batch by our method. The construction of local predictor by the regularized LS estimator (6) has a complexity on the order of $\mathsf{O}(d^3)$. The adaptation of single dictionary $\boldsymbol{L} \in \mathbb{R}^{d\times k}$ and sparse coding $\boldsymbol{s}^{(t)} \in \mathbb{R}^k$ costs $\mathsf{O}(k^2d^3)$. Since we incorporate input features into lifelong learning framework by augmenting $\boldsymbol{L} \in \mathbb{R}^{d\times k}$ into $\boldsymbol{H} \in \mathbb{R}^{(2d)\times k}$, the coupled dictionary adaptation costs $\mathsf{O}\big(k^2(2d)^3\big)$. Thus, the overall complexity of per-batch adaptation is $\mathsf{O}\big(d^3+k^2(2d)^3\big)$, which is independent of batch number.

Remark 4: The online operations of our proposed lifelong learning framework include online prediction and knowledge base adaptation. Specifically, when a new batch with process input data only arrives, a local predictor is constructed explicitly by unsupervised knowledge transfer based on process input and the constructed model is used to predict the process outputs for the batch. Later, only when the corresponding true process output data are available, the supervised knowledge base adaptation takes place. Since the base model is linear, i.e., d is very small, and furthermore $k < d$, the operations of online prediction and knowledge base adaptation are very fast, and the operation time of these online operations is well within the time duration of a batch, which may contains tens to hundreds of samples. In a nutshell, the online operation time of our lifelong learning framework is not an issue.

V. Experiments

Two industrial applications for a penicillin fermentation process and a wastewater treatment plant (WWTP) are used to verify the effectiveness of our proposed lifelong modeling framework. Three metrics, the mean square error (MSE), mean absolute error (MAE) and the coefficient of determination (R^2), are utilized to evaluate both training and online prediction performance.

We operate the proposed framework in the following three different learning modes. 1) Pro-nonadaptive: The training data is first used to build a KB. During online operation, the trained KB is fixed and the learner makes prediction after receiving each batch data. This is essentially Algorithm 1 minus **Knowledge Base Adaptation** part. 2) Pro-adaptive: This is the completed Algorithm 1. After online prediction and when the true process outputs for the current batch become available, the KB is adapted to capture the system characteristics in the current batch. 3) Pro-idealized: This scheme continuously operates in training mode. After observing the complete process input and output data of a new batch, a fully updated KB is obtained that accumulates all the knowledge from all the batches. This 'full' KB is used to reconstruct all the individual models for the individual batches. The individual models then make the 'prediction' for the individual batches. It can be seen that only Pro-nonadaptive and Pro-adaptive can be applied to the scenario of adaptive online modeling and prediction for streaming processes with delayed process output measurement. Pro-idealized is impractical and is used here to represent the idealized performance limit (training performance) of the lifelong modeling framework.

Since the base model for the proposed framework is linear, for a fair comparison, the proposed framework is compared with state-of-art linear models, including LS, Bayesian augmented Lagrangian (BAL) [46], [47], partial LS (PLS) [6], [45], RLS [39], [48], and clustering-based locally linear regression (CLR) [35], [36]. For the LS, BAL and PLS, the training data are used to construct models, and the trained models are fixed during online operation. For the CLR, the k-means clustering is first used to partition the training input data into multiple data clusters, and local models are built by LS for individual clusters. The number of clusters is determined by grid-search based on the training MSE. The trained local linear model set is fixed in online operation. During online prediction, the Euclidean distances are measured between the training clusters and the new input data, and the local model with the closest distance is selected for prediction. We also modify this CLR with ensemble model learning called 'CLR-ensemble', where the trained local models are combined with weights based on distances for prediction. The standard RLS performs online prediction and adaptation on the sample-by-sample base. Specifically, the model adapted at the previous sample is used to make the prediction on the current input sample. After this sample prediction, the true process output sample is assumed available, and the RLS performs the model adaptation using the process input-output sample pair. We refer to this standard RLS as 'RLS-idealized', which cannot be used in adaptive online modeling and prediction for streaming processes with delayed process output measurement. It is used here to represent the idealized performance limit achievable if there exists no delay in the acquisition of process output. To have a fair comparison with our Pro-adaptive, we modify the classic RLS with batch prediction and adaptation called 'RLS-batch'. Specifically, the RLS model updated from the previous batch is used to predict the new batch. After the true process output data for this new batch have acquired, it updates the model over the batch, and the updated model will be used

for the prediction of the next batch. It is worth recapping that LS, BAL, PLS, CLR, CLR-ensemble and our Pro-nonadaptive are nonadaptive models, i.e., they only make online prediction with input and do not make model adaptation.

A. Penicillin Fermentation Process

The penicillin fermentation process is a biochemical fed-batch process with multi-mode time-varying characteristics. It was widely utilized for performance assessment of adaptive modeling approaches [28], [29]. During the fermentation process, the penicillin and substrate concentrations are two hard-to-measure process outputs, which may take hours to acquire from lab analysis. Our goal is to predict these two process outputs using easy-to-measure process input variables before obtaining their true values from lab measurements. The process inputs and outputs are tabulated in Table I. Based on the knowledge of this process, the process input vector at sample i is defined as $\boldsymbol{x}_i = \boldsymbol{u}_i = \left[u_{1_i}\ u_{2_i} \cdots u_{10_i}\right]^{\mathrm{T}} \in \mathbb{R}^{10}$. By changing different operating conditions, 1600 samples are collected from the PenSim tool [28], and they are divided into the training (800 samples) and online testing (800 samples) sets. During online operation, batches of data are received consecutively. In practice, the batch size is determined by applications, e.g., delayed acquisition time for process output. In general, a large batch size enables constructing a more accurate local model for each batch while a small one can better capture the time-varying local data characteristics of different batches. Also the batch size should not be too large so that the process can be approximated by a local linear model over each batch. The batch size is set to 100 in this experiment.

For the proposed framework, the dictionary size k and the regularization parameters are chosen empirically. Additionally, ρ is an important parameter that balances the model's fit to the input data's fit [44], and we set $\rho = 1$ to achieve its best prediction performance. The forgetting factor of RLS is set to 0.98. The ridge term for LS models is empirically set to 10^{-5}. The number of latent variables for PLS is set to 6 to attain its best performance. The augmented Lagrangian parameter and regularize parameter for BAL are carefully chosen to be 10^{-3} to obtain its best performance. The number of clusters for CLR and CLR-ensemble is set to 8, as the training MSE does not decrease with further increasing of clusters.

The performance comparison of various methods for predicting penicillin concentration and substrate concentration are tabulated in Tables II and III, respectively. As expected, the training performance of the LS, BAL, PLS and RLS are the same or similar but our proposed lifelong learning framework attains a slightly better training performance, since it includes input features in knowledge transfer to enrich the modeling accuracy. CLR and CLR-ensemble attain much better training performance than the other methods, because they incorporate

TABLE II
PERFORMANCE COMPARISON OF LS, BAL, PLS, RLS, CLR, AND PROPOSED METHOD FOR PREDICTING PENICILLIN CONCENTRATION IN PENICILLIN FERMENTATION PROCESS.

Methods	Online Adaptation	Initial Training			Online Prediction		
		MSE (dB)	MAE	R^2	MSE (dB)	MAE	R^2
LS	nonadaptive	-32.10	0.0186	0.9946	-25.19	0.0463	0.9744
BAL	nonadaptive	-32.10	0.0186	0.9946	-25.15	0.0466	0.9742
PLS	nonadaptive	-32.02	0.0190	0.9945	-25.29	0.0456	0.9750
CLR	nonadaptive	-48.08	0.0026	0.9999	-19.26	0.0771	0.8999
	nonadaptive (ensemble)	-50.89	0.0016	0.9999	-17.98	0.1134	0.8656
RLS	batch	-32.10	0.0186	0.9946	-4.23	0.4817	-2.1929
	idealized (sample by sample)	-32.10	0.0186	0.9946	-31.82	0.0176	0.9944
Proposed	nonadaptive	-32.54	0.0183	0.9951	-25.48	0.0439	0.9761
	adaptive (batch)	-32.54	0.0183	0.9951	-28.58	0.0308	0.9883
	idealized (all training)	-30.71	0.0228	0.9926	-31.65	0.0204	0.9942

TABLE III
PERFORMANCE COMPARISON OF LS, BAL, PLS, RLS, CLR, AND PROPOSED METHOD FOR PREDICTING SUBSTRATE CONCENTRATION IN PENICILLIN FERMENTATION PROCESS.

Methods	Online Adaptation	Initial Training			Online Prediction		
		MSE (dB)	MAE	R^2	MSE (dB)	MAE	R^2
LS	nonadaptive	-31.87	0.0209	0.9944	-23.55	0.0565	0.9603
BAL	nonadaptive	-28.80	0.0273	0.9886	-25.68	0.0435	0.9757
PLS	nonadaptive	-30.48	0.0220	0.9923	-25.31	0.0443	0.9736
CLR	nonadaptive	-44.29	0.0022	0.9997	-17.98	0.0351	0.8572
	nonadaptive (ensemble)	-45.68	0.0017	0.9998	-9.60	0.3132	0.0153
RLS	batch	-31.87	0.0209	0.9944	-4.39	0.4017	-2.2637
	idealized (sample by sample)	-31.87	0.0209	0.9944	-31.46	0.0202	0.9936
Proposed	nonadaptive	-34.01	0.0134	0.9966	-28.27	0.0290	0.9866
	adaptive (batch)	-34.01	0.0134	0.9966	-30.01	0.0237	0.9910
	idealized (all training)	-32.78	0.0165	0.9954	-34.17	0.0141	0.9966

TABLE I
VARIABLE DESCRIPTION OF PENICILLIN FERMENTATION PROCESS.

Process variables		Description
Inputs	u_1	Aeration rate
	u_2	Agitator power
	u_3	Substrate feed rate
	u_4	Substrate feed temperature
	u_5	Dissolved oxygen concentration
	u_6	Culture volume
	u_7	Carbon dioxide concentration
	u_8	pH
	u_9	Fermentor temperature
	u_{10}	Generated heat
Outputs	y_1	Penicillin concentration
	y_2	Substrate concentration

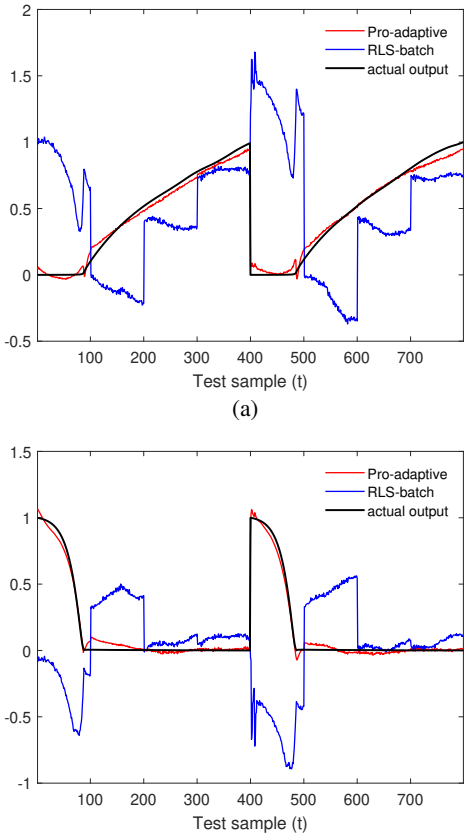


Fig. 3. Prediction performance comparison of RLS-batch and Pro-batch for predicting: (a) penicillin concentration, and (b) substrate concentration in penicillin fermentation process.

local data characteristics of different regions to enhance the overall modeling capacity. What really matters however is the prediction performance. In terms of online prediction accuracy, the nonadaptive LS, BAL and PLS schemes achieve similar performance but again our Pro-nonadaptive achieves a slightly better online prediction performance, again because it includes input features in constructing predictive model. With batch adaptation after predicting new batches, our Pro-adaptive scheme outperforms the Pro-nonadaptive scheme by around 2 dB in the testing MSE. The online prediction accuracy of CLR and CLR-ensemble degrade spectacularly from the training accuracy and become even inferior to nonadaptive single modeling approaches. Its poor prediction performance may be due to the non-smoothness of the boundaries for the subspaces identified in training. Also local models learned in training may not capture newly emerged data states in unseen data. The RLS-batch has very poor online prediction performance. At first glance this may seem strange but it actually makes sense. RLS algorithm has a short memory, allowing it to forget the past data and concentrate on the current data characteristics. Hence after batch adaptation, the model forgets most of the past knowledge and captures the characteristics of the current batch. This model is used to predicting the next batch. For a process with highly time-varying characteristics, the characteristics of the next new batch can be very different from those captured in the model, and this is the root cause of its poor adaptive performance. By contrast, in our Pro-adaptive scheme, the learned KB contains information from all the past batches and, moreover, the process operational information of the current batch is extracted for unsupervised model adaptation to accurately predict the current batch. In Fig. 3, we compare the online predictions of RLS-batch and our Pro-adaptive with the true process outputs for further illustration of the above discussion. With sample-by-sample

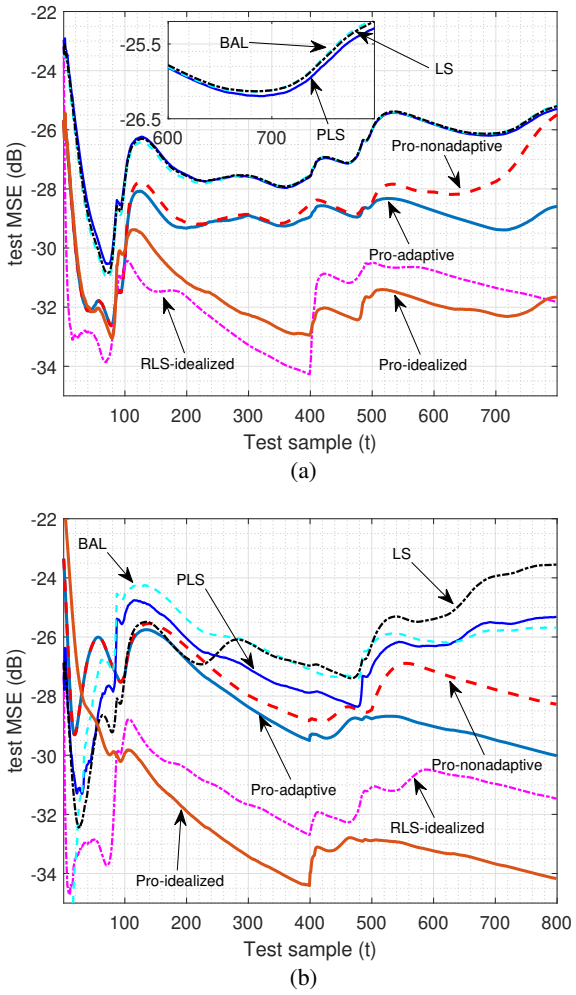


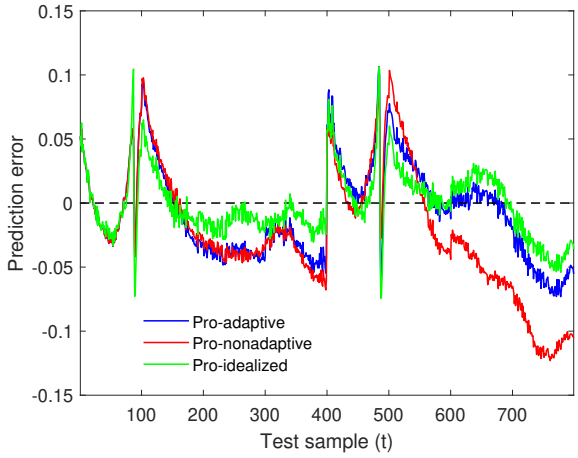
Fig. 4. Comparison of online MSE learning curves of various methods for predicting: (a) penicillin concentration, and (b) substrate concentration in penicillin fermentation process. The curves of RLS-batch, CLR, and CLR-ensemble are outside the plot.

adaptation, RLS-idealized is capable of attaining the excellent online prediction performance that is very close to the training performance. But this adaptive scheme cannot be applied to the penicillin fermentation process with delayed process output measurement. Also as expected, the impractical Pro-idealized provides the ultimate performance limit. The online MSE learning curves of various methods are compared in Fig. 4, where the MSE value at test sample t, MSE(t), is calculated from the first test sample to the t-th test sample as

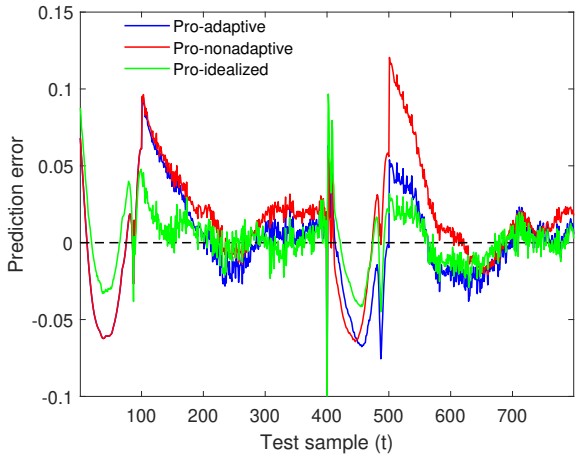
$$\mathrm{MSE(t)} = \frac{1}{\mathrm{t}} \sum_{i=1}^{\mathrm{t}} \left(y_i - \widehat{y}_i\right)^2, \tag{18}$$

in which y_i denotes the i-th process output sample and $\widehat{y}_i$ is its prediction.

The prediction errors of our lifelong learning framework in three different learning modes are compared in Fig. 5. It can be seen that the performance of Pro-adaptive is much better than that of Pro-nonadaptive, and it is very close to the idealized training performance offered by Pro-idealized. This again demonstrates the importance of online KB adaptation and knowledge sharing mechanism. To further investigate this online learning mechanism for building KB, we study the impact of the number of training batches on the online prediction

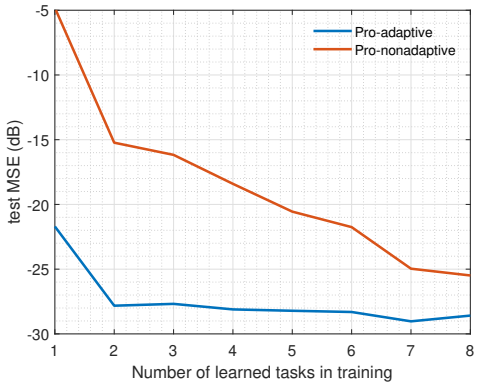


(a)

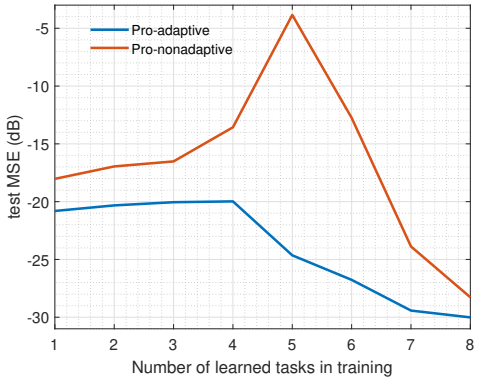


(b)

Fig. 5. Prediction error comparison of the proposed framework in three different learning modes for predicting: (a) penicillin concentration, and (b) substrate concentration in penicillin fermentation process.



(a)



(b)

Fig. 6. Performance of lifelong learning of consecutive training tasks for predicting (a) penicillin concentration, and (b) substrate concentration in penicillin fermentation process.

performance of the two practical lifelong learning schemes, Pro-nonadaptive and Pro-adaptive, over all the online test batches. The experimental results are plotted in Fig. 6, where the effectiveness of the adaptive lifelong model, Pro-adaptive, is self-evident. On predicting the penicillin concentration, for example, two training batches are enough for Pro-adaptive to attain a comparable performance to the case of using all the 8 training batches. This demonstrates that equipped with online KB adaptation, the Pro-adaptive scheme is capable of compensating for lack of training data/tasks by continually acquiring most up-to-data knowledge online.

B. Wastewater Treatment Plant

The WWTP detailed in [49] is a dynamic process subject to large perturbations in influent flow rate and pollutant load, together with uncertainties on the composition of the incoming wastewater. The operation of this WWTP is to remove organic matter and perform nitrification and denitrification. To achieve this aim, it is essential to estimate the flow rate during the plant operation. The process inputs and output of this WWTP are listed in Table IV. The influent data are collected under severe weather condition (combination of dry weather and long rainy period), which makes the underlying system characteristics highly time-varying and imposes a challenge on predictive models [50]. The plant knowledge suggests that the plant input vector at sample i can be chosen as $\boldsymbol{x}_i = \boldsymbol{u}_i = \left[u_{1_i}\ u_{2_i} \cdots u_{5_i}\right] \in \mathbb{R}^5$. We collect 1300 samples

TABLE V
PERFORMANCE COMPARISON OF LS, BAL, PLS, RLS, CLR, AND PROPOSED METHOD FOR PREDICTING FLOW RATE OF WWTP.

Methods	Online Adaptation	Initial Training			Online Prediction		
		MSE (dB)	MAE	R^2	MSE (dB)	MAE	R^2
LS	nonadaptive	-31.98	0.0198	0.9625	-10.98	0.1517	-0.2774
BAL	nonadaptive	-31.98	0.0198	0.9625	-10.95	0.1522	-0.2860
PLS	nonadaptive	-31.88	0.0198	0.9616	-10.94	0.1531	-0.2873
CLR	nonadaptive	-49.01	0.0024	0.9993	-11.92	0.1289	-0.0275
	nonadaptive (ensemble)	-47.95	0.0027	0.9991	-11.64	0.1425	-0.0967
RLS	batch	-31.98	0.0198	0.9625	-11.05	0.1922	-0.2564
	idealized (sample by sample)	-31.98	0.0198	0.9625	-18.61	0.0562	0.7796
Proposed	nonadaptive	-33.06	0.0168	0.9707	-11.61	0.1485	-0.1045
	adaptive (batch)	-33.06	0.0168	0.9707	-14.73	0.1019	0.4622
	idealized (all training)	-21.31	0.0669	0.5616	-19.90	0.0692	0.8364

from the WWTP dataset, among which the first 500 samples are used for training, while the rest are for online prediction. To better capture the local characteristics of this dynamic process, the batch size is set to 50. Hence, there are total of 26 data batches. The first 10 batches are used for training while the rest 16 batches are for online prediction and adaptation. The dictionary size k, regularization parameters and ρ are set to 2, 10^{-8}, and 1, respectively, for the proposed method. The number of latent variables for PLS is 3. The augmented Lagrangian parameter and regularize parameter for BAL are chosen to be 10^{-3}. The cluster number for CLR and CLR-ensemble is set to 25 empirically.

The performance comparison for various models are presented in Table V, and their online MSE learning curves are given in Fig. 7. Again, the same observations regarding the various methods compared can be drawn as for the penicillin fermentation process with two exceptions. First, RLS-batch in this case achieves a slightly better prediction accuracy than the nonadaptive LS, BAL and PLS. This is because the batch size is smaller, and there are sufficient adjacent batches which have similar characteristics, see the true process output sequence plotted in Fig. 8. Second, different from the previous case study, the CLR and CLR-ensemble attain slightly better online prediction accuracy than the LS, BAL, PLS, RLS-batch and Pro-nonadaptive. But they are inferior to the Pro-adaptive, because the trained local models of CLR and CLR-ensemble cannot capture the abrupt change in testing data. This again demonstrates the importance of model adaptation. A notable feature in the online MSE learning curves of Fig. 7 is that around the test sample 300, there is a sharp deterioration in online prediction accuracy for all the methods. This 'abnormal' phenomenon can be explained by the time-varying plant characteristics caused by abrupt weather change, shown in Fig. 8. Like the 500 training samples, the first 300 test samples (samples t = 500 to 800) are taken at a dry weather period but the next 300 test samples (t = 800 to 1100) are taken in a long rainy period. Hence, the underlying plant dynamics experience an abrupt change around t = 800, and the data characteristics become very different from those of the training data. This kind of sudden sharp change is difficult for any adaptive model to cope well with.

The online prediction results of the proposed framework in three learning modes are compared in Fig. 9. It can be seen that Pro-nonadaptive is hardly able to track the abrupt process drift starting at t = 800, while Pro-adaptive does offer some capability to track this abrupt change in the plant dynamics.

TABLE IV
VARIABLE DESCRIPTION OF WWTP.

Input and output variables	Description
u_1	Readily biodegradable substrate
u_2	Particulate inert organic matter
u_3	Slowly biodegradable substrate
u_4	Active heterotrophic biomass
u_5	$NH_4^+ + NH_3$ nitrogen
u	Flow rate

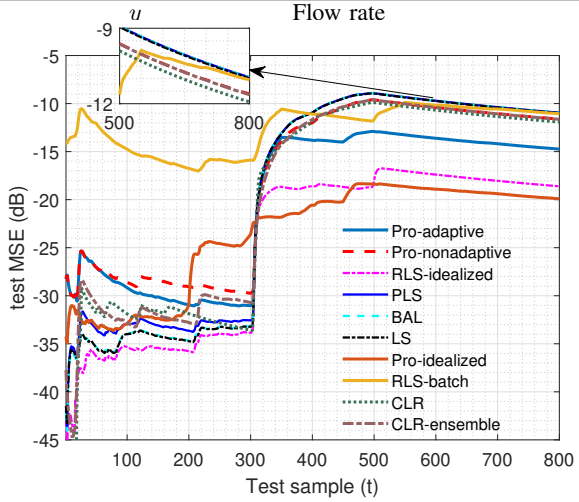


Fig. 7. Comparison of online MSE learning curves of various methods for predicting flow rate of WWTP.

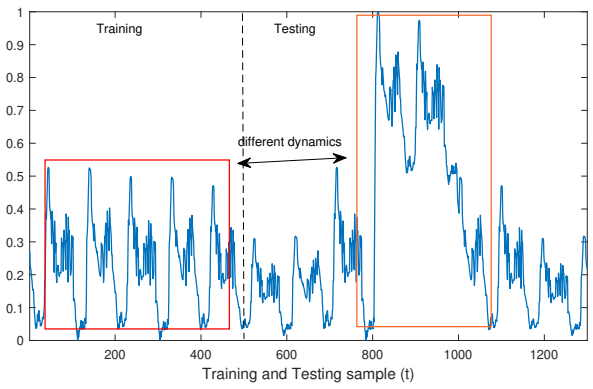


Fig. 8. Process output (flow rate) of WWTP.

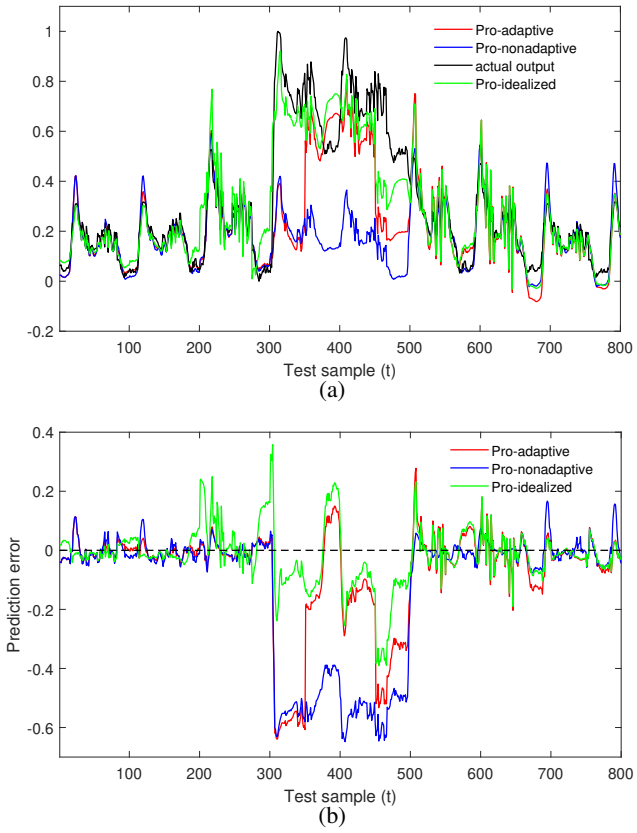


Fig. 9. Comparison of (a) predicted outputs and (b) prediction errors of proposed framework in three different learning modes for WWTP.

It is informative to exam the Pro-adaptive scheme's prediction behaviors over the rainy period in more details. Observe from Fig. 9 (a) that for the first rainy data batch (test samples 300 to 350), the local predictor performs poorly, because the current KB is learned from the previous dry data batches. After online prediction, the batch adaptation enables the KB to encode the new rainy characteristics. Consequently, for the next two batches (test samples 350 to 450), the prediction accuracy improves significantly. The prediction performance is degraded again for the next batch (test samples 450 to 500) as the underlying plant characteristic begins to change to a dry-weather one. Because online adaptation after batch prediction is able to capture this new dynamic, the subsequent batch prediction (test sample 500 onwards) achieves high accurate.

VI. Conclusions and Future Research

This paper has proposed a novel lifelong learning framework for online modeling and prediction of industrial time-varying streaming processes with delayed process output measurement. Our main contribution has been to encode input data into feature vector that contains essential process operating information, and to utilize these input features to augment local predictors on each batch data. We have used coupled dictionary learning to connect the input feature's space with the predictor's parameter space, where the two spaces are linked through two dictionaries that are coupled by the same sparse coding. The proposed learning framework has the capability of unsupervised learning, which enables the learner to rapidly reconstruct local predictors for the new coming batches given only process inputs and provide timely predictions for the true process outputs. Two industrial case studies, involving a penicillin fermentation process and a wastewater treatment plant, have been used to demonstrate the effectiveness as well as the intrinsic learning mechanism of our proposed framework for online prediction and adaptation of industrial processes with hard-to-measure process output variables.

This paper has opened up a new research direction for streaming process data analytics and modeling under the framework of lifelong learning. There are many theoretical and practical issues that deserve further investigation. One important issue is how to define the knowledge base in lifelong learning by incorporating prior process knowledge [7], [45] as well as prior knowledge of the underlying process's critical constraints [51], [52]. This paper has incorporated operation knowledge, namely, process inputs, into lifelong learning to enhance modeling performance and enable online prediction. Other interpretable process physical features can also be integrated. Although the delay time of process output data acquisition is determined by the constraints of industrial applications, there is a scope for investigating how to choose appropriate batch size. The base model for lifelong learning is linear. A long-term research is to study the potential of adopting a nonlinear base model in lifelong learning.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[2] D. Silver, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[3] J. Jumper, *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, Jul. 2021.
[4] S. Ravuri, *et al.*, " Skillful precipitation nowcasting using deep generative models of radar," *Nature*, vol. 597, no. 7878, pp. 672–677, Sep. 2021.
[5] C. Shang and F. You, "Data analytics and machine learning for smart process manufacturing: Recent advances and perspectives in the big data era," *Engineering*, vol. 5, no. 6, pp. 1010–1016, Dec. 2019.
[6] S. J Qin, *et al.*, "Bridging systems theory and data science: A unifying review of dynamic latent variable analytics and process monitoring," *Annual Reviews in Control*, vol. 50, pp. 29–48, Oct. 2020.
[7] M. Mowbray, *et al.*, "Industrial data science – a review of machine learning applications for chemical and process industries," *Reaction Chemistry & Engineering*, vol. 7, pp. 1471–1509, Apr. 2022.
[8] S. Chen, X. X. Wang, and C. J. Harris, "NARX-based nonlinear system identification using orthogonal least squares basis hunting," *IEEE Trans. Control Systems Technology*, vol. 16, no. 1, pp. 78–84, Jan. 2008.
[9] S. A. Billings and S. Chen, "The determination of multivariable nonlinear models for dynamic systems," in *Control and Dynamic Systems*, Volume 7 of *Neural Network Systems Techniques and Applications*, ed. C.L. Leondes, San Diego: Academic Press, 1998, pp. 231–278.
[10] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Mag.*, vol. 10, no. 4, pp. 12–25, Nov. 2015.
[11] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case," *J. Manufacturing Systems*, vol. 54, pp. 138–151, Jan. 2020.
[12] P. Kadlec, R. Grbic, and B. Gabrys, "Review of adaptation mechanisms for data-driven soft sensors," *Computers & Chemical Engineering*, vol. 35, no. 1, pp. 1–24, Jan. 2011.
[13] H. Chen, Y. Gong, X. Hong, and S. Chen, "A fast adaptive tunable RBF network for nonstationary systems," *IEEE Trans. Cybernetics*, vol. 46, no. 12, pp. 2683–2692, Dec. 2016.
[14] Y. Yuan, *et.al.*, "Expert control system-based multimode hybrid switching control strategy for microwave lignite drying," *Drying Technology*, vol. 35, no. 12, pp. 1468–1480, Jun. 2017.

[15] Z. Chen and B. Liu, *Lifelong Machine Learning* (2nd edition), Morgan & Claypool Publishers, 2018.

[16] G. Sun, *et al.*, "Representative task self-selection for flexible clustered lifelong learning," *IEEE Trans. Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1467–1481, Apr. 2022.

[17] G. Sun, *et al.*, "What and how: Generalized lifelong spectral clustering via dual memory," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3895–3908, Jul. 2022.

[18] F. Ye and A. G. Bors, "Lifelong mixture of variational autoencoders," *IEEE Trans. Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 461–474, Jan. 2023.

[19] F. Ye and A. G. Bors, "Lifelong teacher-student network learning," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6280–6296, Oct. 2022.

[20] G. Sun, Y. Cong, and X. Xu, "Active lifelong learning with 'watchdog'," in *Proc. AAAI 2018* (New Orleans, LA, USA), Feb. 2-7, 2018, pp. 4107–4114.

[21] G. Sun, *et al.*, "Lifelong metric learning," *IEEE Trans. Cybernetics*, vol. 49, no. 8, pp. 3168–3179, Aug. 2019.

[22] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Proc. ICML 2013* (Atlanta, GA, USA), Jun. 16-21, 2013, pp. 507–515.

[23] P. Ruvolo and E. Eaton, "Active task selection for lifelong machine learning," in *Proc. AAAI 2013* (Bellevue, Washington, USA), Jul. 14-18, 2013, pp. 862–868.

[24] H. B. Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor, "Online multi-task learning for policy gradient methods," in *Proc. ICML 2014* (Beijing, China), Jun. 21-26, 2014, pp. 1206–1214.

[25] H. B. Ammar, R. Tutunov, and E. Eaton, "Safe policy search for lifelong reinforcement learning with sublinear regret," in *Proc. ICML 2015* (Lille, France), Jul. 6-11, 2015, pp. 2361–2369.

[26] H. B. Ammar, E. Eaton, J. M. Luna, and E. Eaton, "Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning," in *Proc. IJCAI 2015* (Buenos Aires, Argentina), Jul. 25-31, 2015, pp. 3345–3351.

[27] J. Mendez, B. Wang, and E. Eaton, "Lifelong policy gradient learning of factored policies for faster training without forgetting," in *Proc. NIPS 2020* (Vancouver, BC, Canada), Dec. 6-12, 2020, pp. 1–12.

[28] W. Shao, *et al.*, "Online soft sensor design using local partial least squares models with adaptive process state partition," *Chemometrics and Intelligent Laboratory Systems*, vol. 144, pp. 108–121, May 2015.

[29] H. Jin, *et al.*, "Dual learning-based online ensemble regression approach for adaptive soft sensor modeling of nonlinear time-varying processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 228–244, Feb. 2016.

[30] T. Liu, S. Chen, S. Liang, and C.J. Harris, "Selective ensemble of multiple local model learning for nonlinear and nonstationary systems," *Neurocomputing*, vol. 378, pp. 98–111, Feb. 2020.

[31] T. Liu, S. Chen, S. Liang, and C.J. Harris, "Growing and pruning selective ensemble regression for nonlinear and nonstationary systems," *IEEE Access*, vol. 8, no. 1, pp. 73278–73292, Apr. 2020.

[32] T. Liu, *et al.*, "Multi-output selective ensemble identification of nonlinear and nonstationary industrial processes," *IEEE Trans. Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 1867–1880, May 2022.

[33] W. J. Rugh and J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, no. 10, pp. 1401–1425, Oct. 2000.

[34] D. J. Leith and W. E. Leithead, "Survey of gain-scheduling analysis and design," *Int. J. Control* , vol. 73, no. 11, pp. 1001–1025, Nov. 2010.

[35] M. Mojto, K. L'ubusky, M. Fikara, and R. Paulen, "Support vector machine-based design of multi-model inferential sensors," *Computer Aided Chemical Engineering*, vol. 51, pp. 1045–1050, 2022.

[36] B. Ari and H. A. Guvenir "Clustered linear regression," *Knowledge-Based Systems*, vol. 15, no. 3, pp. 169–175, Mar. 2002.

[37] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[38] S. Chen, X. Hong, B. L. Luk, and C. J. Harris, "Construction of tunable radial basis function networks using orthogonal forward selection," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 39, no. 2, pp. 457–466, Apr. 2009.

[39] S. Chen, "Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning," *Electronics Letters*, vol. 31, no. 2, pp. 117–118, Jan. 1995.

[40] T. Liu, *et al.*, "Fast adaptive gradient RBF networks for online learning of nonstationary time series," *IEEE Trans. Signal Processing*, vol. 68, pp. 2015–2030, Mar. 2020.

[41] T. Liu, *et al.*, "Fast tunable gradient RBF networks for online modeling of nonlinear and nonstationary dynamic processes," *J. Process Control*, vol. 93, pp. 53–65, Sep. 2020.

[42] T. Liu, *et al.* , "Deep cascade gradient RBF networks with output-relevant feature extraction and adaptation for nonlinear and nonstationary processes," *IEEE Trans. Cybernetics*, early access, 2022. DOI: 10.1109/TCYB.2022.3152107.

[43] D. Isele, M. Rostami, and E. Eaton, "Using task features for zero-shot knowledge transfer in lifelong learning," in *Proc. IJCAI 2016* (New York, USA), Jul. 9-15, 2016, pp. 1620–1626.

[44] M. Rostami, D. Isele, and E. Eaton, "Using task descriptions in lifelong machine learning for improved performance and zero-shot transfer," *J. Artificial Intelligence Research*, vol. 67, pp. 673–704, Mar. 2020.

[45] S. J Qin, *et al.*, "Integration of process knowledge and statistical learning for the Dow data challenge problem," *Computers & Chemical Engineering*, vol. 153, no. 107451, pp. 1–15, Oct. 2021.

[46] X. Tang, L. Zhang, and X. Li, "Bayesian augmented Lagrangian algorithm for system identification," *Systems & Control Letters*, vol. 120, pp. 9–16, Oct. 2018.

[47] Z. Liu, *et al.*, "Wind turbine blade bearing fault diagnosis under fluctuating speed operations via Bayesian augmented Lagrangian analysis," *IEEE Trans. Industrial Informatics*, vol. 17, no. 7, pp. 4613–4623, Jul. 2021.

[48] C. S. Leung, G. H. Young, J. Sum, and W. K. Kan, "On the regularization of forgetting recursive least square," *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1482–1486, Nov. 1999.

[49] U. Jeppsson, *et al.*, "Towards a benchmark simulation model for plant-wide control strategy performance evaluation of WWTPs," *Water Science & Technology*, vol. 53, no. 1, pp. 287–295, Jan. 2006.

[50] Y. Liu, B. Liu, X. Zhao, and M. Xie, "Development of RVM-based multiple-output soft sensors with serial and parallel stacking strategies," *IEEE Trans. Control Systems Technology*, vol. 27, no. 6, pp. 2727–2734, Nov. 2019.

[51] X. Hong and S. Chen, "A new RBF neural network with boundary value constraints," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 39, no. 1, pp. 298–303, Feb. 2009.

[52] S. Chen, X. Hong, and C. J. Harris, "Grey-box radial basis function modelling," *Neurocomputing*, vol. 74, no. 10, pp. 1564–1571, 2011.