

# Deep Learning on Key Performance Indicators for Predictive Maintenance in SAP HANA

Jaekoo Lee<sup>†</sup>, Byunghan Lee<sup>†</sup>,  
Jongyoon Song<sup>‡</sup>, Sungroh Yoon<sup>\*</sup>  
Electrical and Computer Engineering  
Seoul National University  
Seoul 08826, Korea  
sryoon@snu.ac.kr

Jaesik Yoon<sup>‡</sup>, Yongsik Lee, Donghun Lee  
SAP Labs Korea  
Seoul 06578, Korea  
{jaesik.yoon01,yong.sik.lee,dong.hun.lee}  
@sap.com

## ABSTRACT

With a new era of cloud and big data, Database Management Systems (DBMSs) have become more crucial in numerous enterprise business applications in all the industries. Accordingly, the importance of their proactive and preventive maintenance has also increased. However, detecting problems by predefined rules or stochastic modeling has limitations, particularly when analyzing the data on high-dimensional Key Performance Indicators (KPIs) from a DBMS. In recent years, Deep Learning (DL) has opened new opportunities for this complex analysis. In this paper, we present two complementary DL approaches to detect anomalies in SAP HANA. A temporal learning approach is used to detect abnormal patterns based on unlabeled historical data, whereas a spatial learning approach is used to classify known anomalies based on labeled data. We implement a system in SAP HANA integrated with Google TensorFlow. The experimental results with real-world data confirm the effectiveness of the system and models.

## 1. INTRODUCTION

Over the last four decades, Database Management Systems (DBMSs) have been playing an increasingly critical part in numerous enterprise business applications, and accordingly, their operation has become more crucial to provide reliable services to customers. In addition, the increased complexity of the DBMS architecture with more functionalities that may incorporate multiple engines [65] and the increased requirements of the service level of global businesses have made the operation efforts of Database Administrators

(DBAs) more considerable. These efforts may include monitoring the system health and performance, checking system-generated alerts, managing backup process, changing configurations, tuning performance, managing the landscape and lifecycle, setting security measures, and ensuring availability by minimizing the possible downtime. In a cloud service environment, where numerous DBMSs need to be operated simultaneously, these operation tasks cannot rely on humans and the cost for maintenance also increases exponentially. For example, to achieve 99.999% availability, which corresponds to an allowable downtime of only 5 minutes/year as presented in Table 1, an operation relying on human knowledge and experiences is not a feasible approach, and so, the importance of proactive operation and preventive maintenance becomes more crucial.

Since the late 1990s, much of the work has focused on reducing human efforts. Some examples are building self-tuning database systems [9], modeling system behaviors [63], detecting outliers from the time-series of the Key Performance Indicators (KPIs) [26], or predicting future behaviors based on data and learning [12, 55]. However, these approaches are mainly either for specific domain problems or limited to the experimental data, so that it is still difficult to predict complex system behaviors and abnormal situations in real systems using them. Additionally, with the internal behaviors of a DBMS strongly depending on the characteristics of the given workload and particularly, with the recent trend of mixed workloads such as OLTP and OLAP from diverse and complex applications [54, 57], the prediction of the behaviors with stochastic modeling becomes more challenging. Another challenge is the high dimensionality of the KPIs. A DBMS generates dozens of KPIs that might be highly correlated; therefore, the analysis of the data requires significant insights, which are not possible to obtain from humans or simple models in various cases.

To revisit this problem, we also considered time-series KPI data, assuming that they were sufficiently representative to model the system behaviors. Much of the time-series data modeling has focused on time-sequential stochastic methods [32, 68, 31, 5, 41, 53, 44, 63, 69]. However, these methods yielded good results mostly only with simple time-series that have periodicity and trends, and they did not exhibit a better performance than the learning-based methods that predict the patterns of the KPIs using time-series batches when the periodicity and trends are not constant [32, 68, 31]. Since the early 1990s, learning-based methods such as Artificial Neural Networks (ANNs) have also been studied, but

<sup>\*</sup>To whom correspondance should be addressed.

<sup>†</sup>These authors contributed equally to this work (first author).

<sup>‡</sup>These authors contributed equally to this work (second author).

**Table 1: Annual downtime of availability**

| Availability | Annual Downtime    |
|--------------|--------------------|
| 97%          | 11 days            |
| 98%          | 7 days             |
| 99%          | 3 days 15 hours    |
| 99.9%        | 8 hours 48 minutes |
| 99.99%       | 53 minutes         |
| 99.999%      | 5 minutes          |
| 99.9999%     | 32 seconds         |

they were effective at an experimental level [5, 41, 53, 44]. With the era of Deep Learning (DL), deep neural network models have shown remarkable success in various applications with real data beyond the laboratory level and started to surpass human capabilities [25, 64, 37, 10]. Therefore, we have attempted to apply several DL methods to analyze the time-series data and predict the anomalies.

For choosing the appropriate DL approaches, we have considered the following. First, the models need to detect the unforeseen patterns based on the historical data of the past. This is the problem that the typical time-series anomaly detection methods have tried to resolve. Second, if the problems are known already, then the models need to detect better. This is because for some abnormal situations the issues are already analyzed automatically or by database experts and can be clearly labeled as abnormal situations. In this study, we used two novel DL models that can satisfy the above requirements, and developed a system incorporating these models. For the first requirement, we adopted the Recurrent Neural Network (RNN) structure [10, 25]; in particular, we employed the encoder-decoder structure of the Long Short-Term Memory (LSTM) to predict the future values with a representative abstract that sufficiently incorporates the past patterns. For the second requirement, we labeled the half-year KPI data with anomalies and built Convolutional Neural Networks (CNNs) [37], which showed a strong performance for locality detection. Specifically, in our research, a short-cut connection that can lead to learning of more complex patterns by solving gradient problems is used. Subsequently, in this paper, we will validate the feasibility of the two approaches based on our experimental results.

The remainder of the paper is organized as follows. Section 2 outlines the existing research works related to learning-based analysis on time-series data, system load prediction, and critical issue prediction. Section 3 describes the overall system architecture including the HANA Cockpit as the main control panel, HANA Predictive Analysis Library (PAL) to integrate the machine learning platform, and user scenarios for anomaly prediction in SAP HANA. Section 4 presents in detail both the proposed deep neural network models for KPI learning. Section 5 discusses the experimental setup and results demonstrating the feasibility of the proposed models with the measured prediction performance. Section 6 concludes the paper, addressing the main contribution and future works to further improve our approaches.

## 2. RELATED WORK

### 2.1 Learning-based Analysis on Time-series

The system logs of database systems including the KPIs are a type of temporal data. To detect anomalies in such

temporal data, a model based on all the temporal data is trained, and then the scores are calculated for each test sequence with the learned model. According to Gupta *et al.* [26], the definition of anomalies in the temporal data are classified as: anomalies in time-series batches (*i.e.*, coarse-grained) and anomalies within a given time-series (*i.e.*, fine-grained). Hence, the detection methods for these two categories of anomalies have different characteristics.

First, we address the detection methods for anomalies in time-series batches. These can be unsupervised discriminative, unsupervised parametric, and window-based approaches. In unsupervised discriminative approaches, a similarity function that measures the similarity between the temporal sequences is defined. With this function, the temporal data are clustered using clustering algorithms such as *k*-means [50], Expectation-Maximization (EM) [52], or single-class Support Vector Machine (SVM) [14, 16, 46, 67], and then, the abnormality score (*i.e.*, distance from the centroid of the closest cluster) is calculated for a test sequence. Unsupervised parametric approaches construct a summary model using parametric models such as Finite State Automata (FSA) [8, 47, 48, 61], Markov models [72, 66, 15, 42], or Hidden Markov Models (HMMs) [8, 18, 21, 59, 74]. A score for a test sequence, which is the probability of the generation of the sequence from the summary model, is calculated, and the sequence with low score is classified as an anomaly. In window-based approaches, test sequences have multiple overlapping subsequences. The size of the window (*i.e.*, subsequence) length is defined, and a frequency table of the normal subsequences is constructed. For a test sequence, the subsequences of the test sequence are compared, and the subsequences that are not present in the frequency table are classified as anomalies [21, 7, 13, 23, 24].

Next, we discuss the detection methods for anomalies within a given single time-series. There are two possible approaches: considering points as anomalies and taking subsequences as anomalies. In the points as anomalies approach, a summary prediction model is constructed and the deviation of a point from the predicted value is calculated using the summary prediction model. The summary prediction model can be learned from clustering [4, 29], single-layer linear network, multi-layer perceptron [29], support vector regression [45], and mixture transition distribution [38]. In the subsequence as anomalies approach, the subsequence that has the largest distance from its nearest non-overlapping match is defined as an anomaly. To calculate the distance, top-*K* pruning including heuristic reordering of candidate subsequences [35], locality sensitive hashing [70], Haar wavelet and augmented tries [6, 19], and Symbolic Aggregate Approximation (SAX) with augmented trie [43] can be used.

Our proposed approach considers both coarse- and fine-grained anomaly prediction problems. To this end, we propose two models on SAP HANA to complementarily predict the critical issues for achieving predictive maintenance in SAP HANA. One model is for system load KPI prediction by a temporal learning approach, and the other is for critical issue prediction based on a spatial learning approach. The details of each approach are described in the following sections.

### 2.2 System Load KPI Prediction

There have been numerous studies on system load KPI prediction in various areas, including energy management

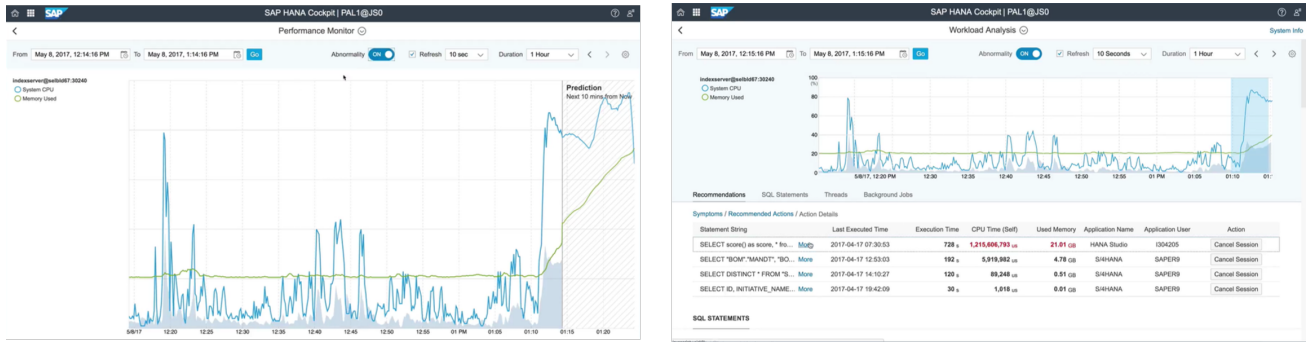


Figure 1: Examples of graphical user interface in HANA Cockpit

systems, cooling systems as well as DBMSs [32, 68, 31, 5, 41, 53, 44, 63, 69, 73]. Basically, this topic has been classified as a regression problem or stochastic time-sequential modeling problem. Generally, short-term prediction by using time-series batches is considered as a regression problem [5, 41, 53, 44], whereas long-term prediction with aggregated time-series data (*i.e.*, hourly, daily, weekly) is considered as a time-sequential modeling problem because there can be a periodic pattern that is yearly or quarterly [32, 68, 31].

The researches considering this topic as a regression problem have been modeling the patterns with ANNs because the model capabilities of these ANNs can be bigger than ridge regression or general linear modeling [5, 41, 53, 44]. However, there are some problems with ANNs [51]. One of them is the back-propagation, which is the most representative method that ANNs learn, requiring significant computation power, and it is also not easy to satisfy it in the system. In addition, a vanishing gradient problem emerges when deep ANNs are implemented to study a complex pattern [51]. The vanishing gradient problem is a phenomenon in which modeling cannot be performed appropriately owing to the poor learning on the lower layer when deep ANNs are implemented. However, the recent General Purpose GPU (GPGPU)-based learning processes and enormous computation power in cloud services have solved some of the problems requiring very large computation usage [37]. The vanishing gradient problem also requires tremendous effort for resolving it. Recently, several studies have made some progress on the vanishing gradient problem [28, 22, 33, 10, 11]. The LSTM encoder-decoder used in this work also has a much more relaxed vanishing gradient problem than the naïve RNN structure. In addition, the encoder sends the compressed valuable information to the decoder, and thus, the decoder estimates a more accurate future system load value than simple naïve RNNs.

The studies considering this topic as a stochastic time-sequential modeling problem, generally model the system load patterns with extraction of the trend or period from the time-series data [32, 68, 31, 63, 69]. Thus, this stochastic time-sequential modeling strongly works for periodic and trend data. By aggregating the data on a daily or weekly basis and observing it, a certain period or trend can be easily observed. However, short intervals of data usually do not have a trend or periodicity. For example, when a KPI is recorded every few seconds, the time for the application to run is not constant, and the trend for the frequency is also difficult to extract with several samples of the data. Consequently, this modeling is used for aggregated data and

long-term prediction.

## 2.3 Critical Issue Prediction

Numerous works have also been conducted for critical issue prediction, particularly in the industrial engineering area to mainly avoid the critical hardware issues, by detecting the precursor symptoms in the performance values or system log of the target system [20, 60, 27]. For predicting a critical issue, in various studies, modeling was conducted with the classification model and labeled data, whose label is whether the issue exists or not.

Similar approaches could be applied to the software area. However, till date, there seem to be only a few outstanding contributions in critical issue prediction against enterprise softwares, particularly for DBMSs. Most approaches are based on time-series modeling such as system load KPI prediction [63, 69]. However, critical issue prediction yields a better performance than system load KPI prediction for a labeled abnormal situation because classification modeling with a label from DBAs can categorize the discriminative results between labeled data as normal and abnormal situations.

## 3. SYSTEM ARCHITECTURE

### 3.1 SAP HANA Cockpit and HANA External Machine Learning Library

The SAP HANA Cockpit [3] is a microservice-based web application that provides a single point of access to a range of tools for administration and allows a detailed monitoring of SAP HANA. DBAs can use this application for administration, such as for database monitoring, user management, and data backup. Users can also start and stop systems or services, monitor the system, configure system settings, manage users, and control authorizations. The HANA Cockpit is equipped for data center management. It can deal with thousands of registered HANA databases and can collect and store KPIs from the registered systems. The performance monitor inside the HANA Cockpit is an application to visually analyze the historical performance data across a range of KPIs related to memory, disk, and CPU usage, as shown in Figure 1. If any suspicious issue is found in the performance monitor, a DBA can continue the analysis in the HANA Cockpit workload analyzer by keeping the context from the performance monitor.

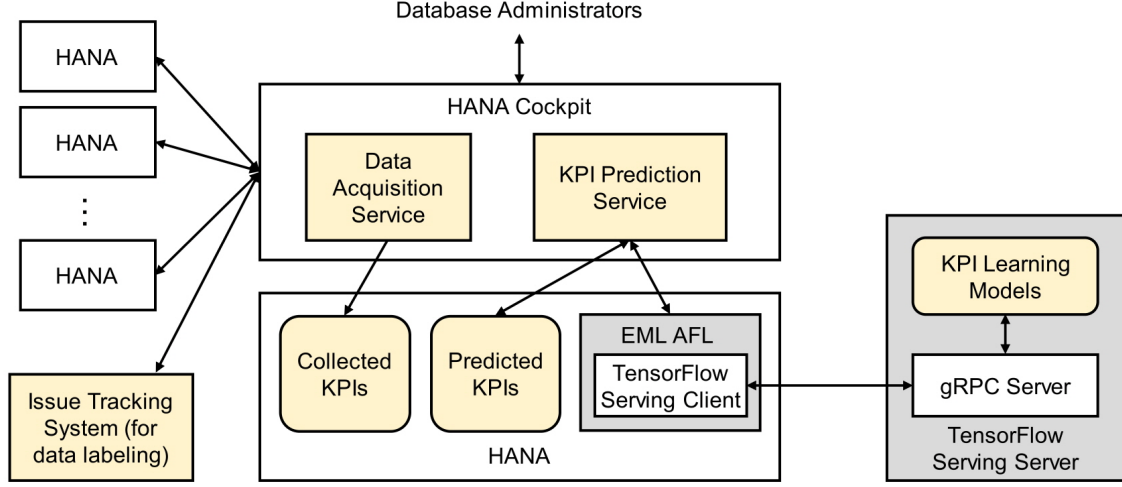


Figure 2: Overall system architecture

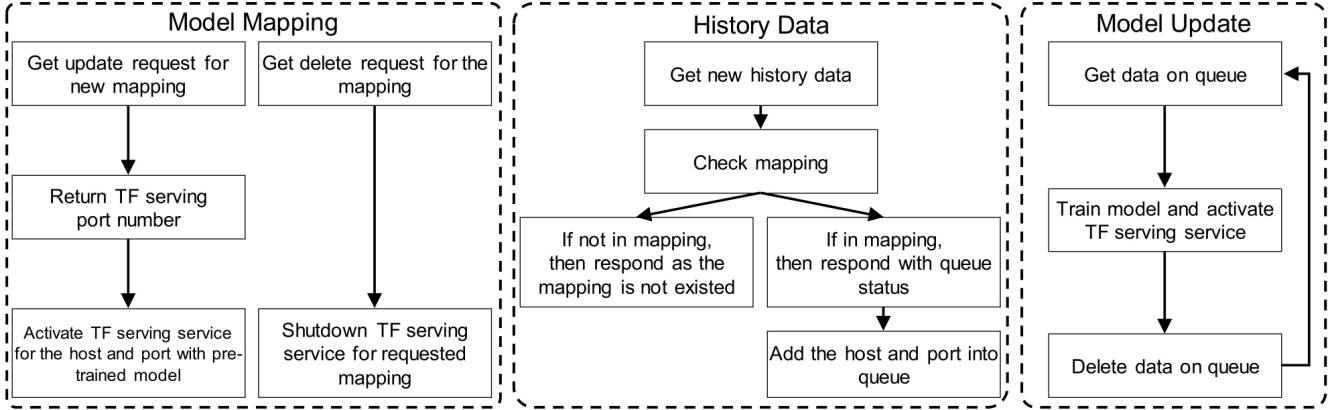


Figure 3: Model management process

SAP HANA has a Predictive Analysis Library (PAL) [2] consisting of numerous machine learning algorithms for applications to leverage the built-in data analysis such as clustering, classification, and regression. It also includes the SAP HANA External Machine Learning Library (EML) [1] to integrate SAP HANA with Google TensorFlow. The integration is achieved by the interface of the SAP HANA Application Function Library (AFL), which communicates with TensorFlow Serving via the gRPC remote procedure call of Google. With this integration, the applications can invoke TensorFlow with the data located in the HANA database.

### 3.2 Overall Architecture

Figure 2 shows the overall system architecture. One common scenario is predicting the system load of each SAP HANA database instance in the near future. This forecasting demand is one of the important scenarios, particularly for realizing maintenance planning or capacity planning. We assumed that enterprise applications have the patterns in their workload and the behavior can be modeled. The predicted values are also used to calculate the normalized difference between the real and estimated values in the past.

If this normalized difference (i.e., the abnormality score) is larger than a preset threshold, then the users are informed to perform a further examination. This reduces false alerts in monitoring and administration and enhances the efficiency of the monitoring efforts. Critical issue prediction is another scenario that can assist DBAs. In contrast with abnormality score that provides warnings based on a general trend, given labeled data such as system unavailability with time stamp information, users can obtain more accurate alerts when a similar situation occurs or is expected to take place. The HANA Cockpit collects the KPI data from the registered systems and stores them into a table in the underlying HANA system. This table is highly compressed by the column store layout of HANA, and the data can be visualized by a system load chart in the HANA Cockpit. In parallel with this Data Acquisition Service, the Prediction Service retrieves the collected data and transfers the data to the GPU-enabled TensorFlow via EML AFL interface. The TensorFlow Serving server receives the data and generates the predicted results from the KPI Learning Models and sends the results back to the Prediction Service via the same EML AFL interface. The returned predicted KPI data together

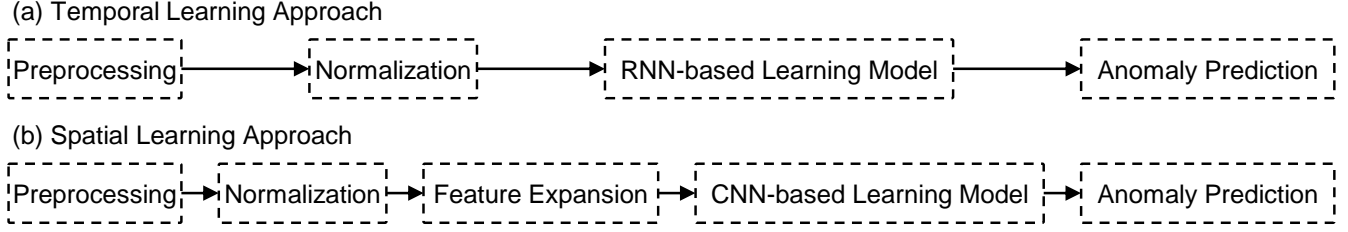


Figure 4: Training and inferring process description on KPI learning models

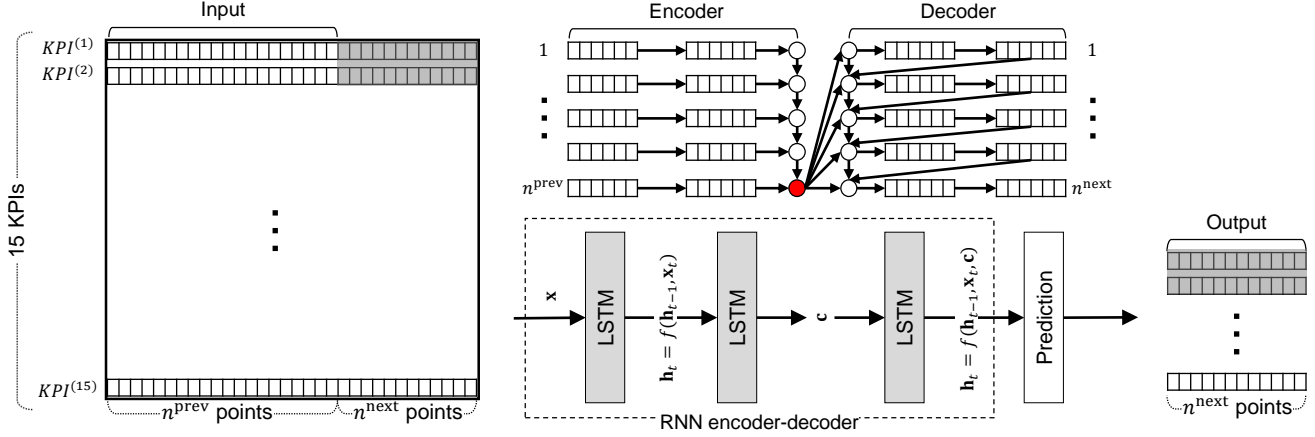


Figure 5: Details of the temporal learning approach

with the anomalies, if existing, are stored in other tables in the underlying HANA and visualized via the HANA Cockpit.

When the new KPI data and labeled data arrive, to manage the models, they are required to be updated to accommodate the new data. Before applying continuous learning in a future work, we developed separate services to update the models regularly via the REST APIs, as displayed in Figure 3. As shown, first, we manage a mapping table to specify the right learning models for each KPI set. Once the right models are specified, the accumulated KPI data are added into a queue in the TensorFlow Serving server. Then the regular learning service updates the models and activates in TensorFlow Serving again based on the configurable period. The learning models are maintained in the changing environments by following this procedure.

## 4. LEARNING MODELS FOR KPIS

In this section, we describe the details of our two approaches. The first is a temporal learning approach that detects some abnormal condition by extracting data-driven features from a given sequential KPIS. The other is a spatial learning approach that predicts known abnormal situations by extracting discriminative features from the changes in the interrelation between the KPIS with label information from the database experts. The learning processes of these two learning models are shown in Figure 4, and each step will be explained in detail in the following sections. The details of the 15 KPIS used in this study are listed in Table 2.

Although both the system load KPI prediction and critical issue prediction are a type of anomaly detection method and receive a previous sequence of KPIS as an input, they have complementary aspects. System load KPI prediction provides an indirect but a more integrated information of the anomaly by comparing the real and predicted KPI values. In contrast, critical issue prediction directly provides the anomaly information, but the model is learned using anomaly labels defined by experts, which can omit implicit and complex evidences for detection. In view of these facts, we conducted experiments with these two approaches independently.

### 4.1 Temporal Learning Approach

Our proposed approach for temporal learning is based on deep RNNs, which can perform various sequence modeling tasks [25]. To be used in the HANA Cockpit, our approach models repetitive patterns in the normal KPIS and predicts the expected subsequent KPIS motivated by a neural machine translation [10]. The tasks include the mapping of an input sequence to an output sequence that is not necessarily of the same length. We assume that the anomaly related system logs have different patterns in the KPIS compared with normal ones.

Figure 4(a) shows the overview of our temporal learning approach. Our *RNN-based Learning Model* has an RNN encoder-decoder-based architecture [10] that consists of two RNNs, encoder and decoder, as shown in Figure 5. The encoder encodes a precedent sequence of the KPIS into a fixed-length representation, and the decoder decodes the fixed-

Table 2: Details of the used KPIs

| Name                       | Description                               |
|----------------------------|---|
| CPU                        | CPU used by all processes                 |
| SWAP_IN                    | Bytes read from swap by all processes     |
| SWAP_OUT                   | Bytes written from swap by all processes  |
| INDEXSERVER_CPU            | CPU used by a service                     |
| INDEXSERVER_SYSTEM_CPU     | OS kernel or system CPU used by a service |
| INDEXSERVER_MEMORY_USED    | Memory used by a service                  |
| PING.TIME                  | Duration of a service for ping request    |
| INDEXSERVER_SWAP_IN        | Bytes read from the swap by a service     |
| BLOCKED_TRANSACTION_COUNT  | The number of blocked SQL transactions    |
| MVCC_VERSION_COUNT         | The number of active MVCC versions        |
| PENDING_SESSION_COUNT      | The number of pending requests            |
| RECORD_LOCK_COUNT          | The number of acquired record locks       |
| CS_UNLOAD_COUNT            | The number of column unloads              |
| WAITING_THREAD_COUNT       | The number of waiting threads             |
| WAITING_SQL_EXECUTOR_COUNT | The number of waiting SQL executors       |

length representation into a subsequent sequence of KPIs. Let a subsequence of KPIs be  $\mathbf{x}$ , where  $x$  is a 15-dimensional vector (see Table 2) to be defined as

$$\mathbf{x} = \langle \mathbf{x}^{\text{prev}}, \mathbf{x}^{\text{next}} \rangle \\ = \langle x_1^{\text{prev}}, x_2^{\text{prev}}, \dots, x_{n^{\text{prev}}}^{\text{prev}}, x_1^{\text{next}}, \dots, x_{n^{\text{next}}}^{\text{next}} \rangle. \quad (1)$$

$\mathbf{x}^{\text{prev}}$  denotes a precedent sequence of the KPIs,  $\mathbf{x}^{\text{next}}$  represents a subsequent sequence of the KPIs,  $n^{\text{prev}}$  denotes the length of  $\mathbf{x}^{\text{prev}}$  and  $n^{\text{next}}$  is the length of  $\mathbf{x}^{\text{next}}$ . The encoder sequentially reads an input sequence of the KPIs,  $\mathbf{x}^{\text{prev}}$ . As it reads each KPI vector, the hidden state of the encoder at time  $t$  changes according to

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t^{\text{prev}}) \quad (2)$$

where  $f(\cdot)$  is an activation function. After reading the end of  $\mathbf{x}^{\text{prev}}$ , the hidden state of the encoder is a summary of the entire precedent sequence, denoted by  $\mathbf{c}$ . Given this summary  $\mathbf{c}$ , the hidden state of the decoder at time  $t$  is defined by

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t^{\text{next}}, \mathbf{c}), \quad (3)$$

and the loss function is the Mean Square Error (MSE) in the output:  $\hat{\mathbf{y}} = \hat{\mathbf{x}}^{\text{next}}$  and  $\mathbf{y} = \mathbf{x}^{\text{next}}$ ,

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{x}^{\text{next}} - \hat{\mathbf{x}}^{\text{next}}\|^2. \quad (4)$$

To train the proposed architecture, we perform a  $\log(1+x)$  transformation for each KPI without any ceiling in the *Pre-processing* step and then apply the min-max normalization (i.e.,  $[0, 1]$ ) for each KPI vector in the *Normalization* step as follows:

$$KPI_i^{(n)} = \frac{KPI_i^{(n)} - \min(KPI^{(n)})}{\max(KPI^{(n)}) - \min(KPI^{(n)})} \quad (5)$$

where  $KPI_i^{(n)}$  denotes the  $i^{\text{th}}$  sample of the  $n^{\text{th}}$  KPI. The missing values (i.e.,  $-1$ 's) are excluded from the above pre-processing procedures, i.e., they remain as  $-1$ . Consequently, each KPI has normalized values ranging from  $-1$  to  $1$ .

## 4.2 Spatial Learning Approach

Recently, there have been outstanding studies on CNNs in various fields [37, 39, 62]. CNNs extract the data-driven features from images, videos and audios defined in a low-dimensional grid domain. CNNs solve a given task by learning and extracting the spatial features using fundamental

statistical characteristics of the inputs expressed in the grid domain, such as the locality and stationary attributes [62, 40]. We also used the locality and stationary assumptions of the given KPIs to learn the spatial features for forecasting the irregularity of the HANA systems. In particular, the proposed CNN-based model is based on ResNet [28], which is known to have deeper layers.

As shown in Figure 4(b), the proposed spatial learning approach is performed sequentially as per the procedure described in the following paragraphs. First, in the *Preprocessing* step, 15 KPIs, which are the main factors for the detection of the anomalies, are selected by removing redundant or constant variables of all the KPIs obtained from the HANA Cockpit similar to the temporal learning approach (refer to Table 2). This step moderates the curse of dimensionality in learning by reducing the dimensions of the KPIs as the input. In addition to choosing the KPIs for the learning efficiency, this step also performs data cleansing, such as handling missing values in the KPIs. Basically the data point is collected every 10 seconds, but if the system is not available, the state is recorded as a distinct value of  $-1$ . Moreover, if the system is overloaded, the data points might be recorded with a time delay. Because this information is also important to classify the abnormal situations, we modeled it by augmenting new dimensions, as depicted in Figure 6. In the next *Normalization* step, the min-max normalization with a ceiling is performed for each KPI to remove not only the differences in the variable ranges among the selected 15 KPIs but also the irrelevant outliers. These outliers of each KPI are ceiled with an average value of the top 1% of the values. The average value ( $\overline{KPI}_{i_t}$ ) to be ceiled is defined as follows:

$$\overline{KPI}_{i_t} = \frac{1}{t} \sum ([KPI_{i_1}, KPI_{i_2}, \dots, KPI_{i_{t-1}}, KPI_{i_t}]) \quad (6)$$

where  $[KPI_{i_1}, KPI_{i_2}, \dots, KPI_{i_{t-1}}, KPI_{i_t}]$  are the first  $t$  partial higher values in ascending order of the given  $KPI_i$ , and  $t$  is empirically set to 0.1% of the length of the samples. Each ceiled KPI performs the min-max normalization using Equation (5).

In the *Feature Expansion* step, new dimensions ( $x^{(n)'} where  $x_i^{(n)'} = x_i^{(n)} - x_{i-1}^{(n)}$ ) having a difference (increase or decrease information) between the samples in each KPI and augmented dimensions from the missing values are added. Consequently, an input having total 32 dimensions consists$



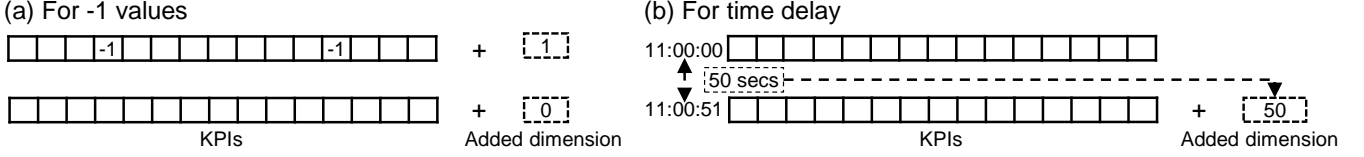


Figure 6: Dealing with the missing values in the KPIs

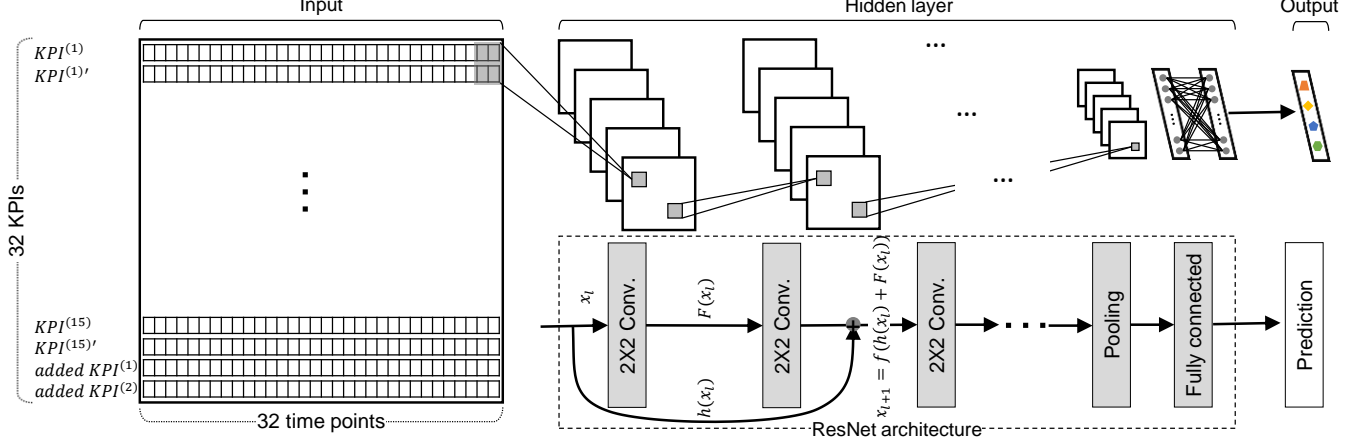


Figure 7: Details of the spatial learning approach

of the original KPIs (15), their variations (15), and augmented variables (2) for the missing values.

The input obtained through the series of the previous steps is depicted in the input in Figure 7. In the *CNN-based Learning Model*, the obtained inputs are actually trained and outputs are inferred by a model of the ResNet architecture [28]. Similar to the cardinal concept of the CNNs, our ResNet also has multiple layers consisting of a convolution layer, pooling layer, and fully connected layer. In the convolutional layer, our ResNet learns the spatial features based on a  $2 \times 2$  matrix filters to integrate the complementary information of original  $x$  and  $x'$  as its variation.

As shown in the ResNet architecture in Figure 7, ResNet proposes a short-cut connection (or a skip connection) that skips over the intermediate layers to prevent the information attenuation caused by the deep layers. When  $\mathbf{X}_i$  is the input and  $\mathbf{X}_{i+1}$  is the output, a residual block having a short-cut connection is defined as follows:

$$\mathbf{X}_{i+1} = f(h(\mathbf{X}_i) + F(\mathbf{X}_i)) \quad (7)$$

where  $F$  is an operation with a weight matrix such as a convolution,  $h(\mathbf{X}_i) = \mathbf{X}_i$  is the identity mapping function, and  $f$  is an activation function.

By stacking these layers, our ResNet also has a deeper neural network architecture that extracts better discriminative features. In our ResNet, the operation of the convolutional layer is similar to the general CNNs. The  $j$ -th output,  $\mathbf{X}_{k+1,j}$ , from the  $k$ -th layer is defined as

$$\mathbf{X}_{k+1,j} = f \left( \sum_{i=1}^{\phi_k-1} F_{k,i,j} * \mathbf{X}_{k,i} \right), \quad j = 1, \dots, \phi_k \quad (8)$$

where  $\phi_k$  is the number of feature maps,  $\mathbf{X}_k$  is the input of  $k$ -th layer,  $h$  is a nonlinear function, and  $F_{k,i,j}$  is the

$2 \times 2$  filter matrix from the  $i$ -th feature map to the  $j$ -th feature map. The striding filters are based on  $2 \times 2$  matrix integrating the original  $x$  and  $x'$  as they pass through the layers.

In the *Anomaly Prediction* step, the model is trained on the given KPI dataset with an extended classification task that predicts four changes in the system states (four-class classification): ‘from normal to normal’, ‘from normal to abnormal’, ‘from abnormal to normal’, and ‘from abnormal to abnormal’ after approximately 5 minutes.

The implementation of the given classification task uses Softmax to make the output of the last layer have relative probabilities. The maximum of these probabilities is selected as the final output of the model in the inference. Our ResNet is efficiently trained using the ReLU activation function [49, 71], momentum optimizer [58], L2 weight decay [30], cross-entropy cost function, and batch normalization [34].

## 5. EXPERIMENTAL RESULTS

### 5.1 Experiment Setup

The proposed system and learning models have been tested and validated using the real KPI datasets obtained from several HANA systems. The systems run on six different hosts ( $H_1, \dots, H_6$ ), and the diverse workloads are executed against each HANA instance. During the monitored period, several abnormal situations occurred. Table 3 presents the time period having the labeled abnormal situations analyzed by DBAs. Each dataset is collected monthly from a HANA system having approximately 100K time points per KPI in 12 days. During the data collection period (approximately six months), there were 25 episodes of anomalies lasting from 5 minutes to 3 hours including the restart time after the required actions to resolve the issues.

Table 3: Used KPI datasets of the HANA systems

| HANA system | H <sub>1</sub>  | H <sub>2</sub>    | H <sub>3</sub>    | H <sub>4</sub>                         | H <sub>5</sub>  | H <sub>6</sub>                         |
|-------------|---|-------------------|-------------------|--|---|--|
| Period      | Abnormal time (DD,hh:mm)                                    |                   |                   |  |   |  |
| MAY         | 12,16:00~12,17:30<br>20,08:50~20,10:15                      | 12,16:40~12,17:36 | 12,01:29~12,09:24 | 12,15:34~12,17:40<br>20,09:13~20,09:28 | 12,15:39~12,16:39   | 12,16:10~12,16:40<br>20,06:02~20,09:28 |
| JULY        | 26,14:20~26,14:30<br>27,14:20~27,14:35<br>28,13:20~28,13:35 | -                 | 26,01:08~26,01:13 | 21,18:41~21,19:37                      | -   | -                                      |
| AUGUST      | 01,10:45~01,12:40<br>09,10:45~09,10:51<br>10,01:01~10,02:22 | 09,21:07~         | 09,22:20~10,01:47 | 09,20:07~10,02:24                      | 01,11:51~01,14:15<br>08,13:29~08,13:35<br>09,22:20~10,02:23 | 09,10:43~09,10:54<br>09,22:10~09,23:16 |
| SEPTEMBER   | -   | -                 | -                 | -                                      | -   | -                                      |

Table 4: Experimental results of the temporal learning approach

|     | H <sub>1</sub> |      |        | H <sub>2</sub> |        | H <sub>3</sub> | H <sub>4</sub> |      |        |
|-----|----------------|------|--------|----------------|--------|----------------|----------------|------|--------|
|     | MAY            | JULY | AUGUST | MAY            | AUGUST | AUGUST         | MAY            | JULY | AUGUST |
| AUC | 0.92           | 0.9  | 0.93   | 0.84           | 0.93   | 0.99           | 0.81           | 0.99 | 0.83   |

Table 5: Generalization of the spatial learning approach

| Train \ Test   | H <sub>1</sub> | H <sub>2</sub> | H <sub>3</sub> | H <sub>4</sub> | H <sub>5</sub> | H <sub>6</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| H <sub>1</sub> | <b>0.971</b>   | 0.457          | 0.335          | 0.495          | 0.503          | 0.472          |
| H <sub>2</sub> | 0.328          | <b>0.978</b>   | 0.409          | 0.518          | 0.582          | 0.517          |
| H <sub>3</sub> | 0.370          | 0.392          | <b>0.955</b>   | 0.331          | 0.328          | 0.508          |
| H <sub>4</sub> | 0.493          | 0.412          | 0.434          | <b>0.962</b>   | 0.426          | 0.519          |
| H <sub>5</sub> | 0.490          | 0.395          | 0.423          | 0.329          | <b>0.959</b>   | 0.530          |
| H <sub>6</sub> | 0.463          | 0.459          | 0.394          | 0.387          | 0.453          | <b>0.974</b>   |

## 5.2 Prediction Performance

To evaluate the performance of our temporal learning approach, we trained a model for each host on its normal KPIs of the ‘SEP’ period listed in Table 3 by using 10 epochs and 100 batch size. We used Adam [36] as an optimizer with learning rate 0.001 and decay 0.001. We set the length of the KPIs as  $n^{\text{prev}} = 90$  points (15 minutes) and  $n^{\text{next}} = 6$  points (1 minute). This implies that we predict the future values of the target KPIs for the next 6 points (1 minute), and then compare the real values with the predicted values. If the difference, which is value by which the MSE exceeds the threshold, is regarded as unforeseen data pattern, then an alarm will be raised to the DBA instantly. We used the architecture of (15-5-15-5-15) that showed the best performance during our simulation, where the first (15-5-15) is for the encoder, and the next (15-5-15) is for the decoder. Note that the architecture is denoted by the number of nodes in each layer in the presented models. Table 4 lists the experimental results of our temporal learning approach in terms of the Area Under the receiver operating characteristic Curve (AUC). The AUC is widely used as a metric, which ranges from 0 to 1, to compare classifiers. The value 1 represents a perfect performance. From the experimental results listed in Table 4, we note that the AUC ranges from 0.81 to 0.99, thereby validating that our approach effectively detects the anomalies by modeling the normal KPIs. The performance is not perfect (0.8x) in a few test cases for some labeled situations, which are similar to the scenarios expected from the discussion with database experts. This is in alignment with the reason for introducing the spatial learning approach for

the labeled abnormal situations in a complementary manner.

Similar to the experiment for the temporal learning approach, various experiments were performed with the spatial learning approach to validate the proposed learning model. Each dataset was divided into 85% training data and 15% testing data to explore the training tendency and generalization performance of the model. In the experiment, we used the simplest approaches that combined random under-sampling techniques with bagging or boosting ensembles to solve the class imbalance problem of the anomalies. Data augmentation with bagging reduced the ratio of the imbalance between the classes by 50%. Figure 8 shows the precision and loss of the training results of the model for each dataset. The model is well trained as demonstrated by the convergence of the plots for the precision and loss. This implies that the presented spatial model can extract the relevant data pattern from the training data and predict incidents similar to the labeled abnormal situations on each host. One interesting result obtained from the inter-system validation test is presented in Table 5. The table shows that the prediction performances using the training data and test data on the same host are quite good, but the prediction performances using the test data from the other hosts are not sufficiently high. This implies that the labeled data have their own precursor symptoms on each host where the issue occurred. This is understandable because the different workloads imposed by the different application systems on physically different hardware can cause varied abnormal situations with the dissimilar KPI patterns. To improve the inter-system validation results, we are considering the diverse approaches in the future work, which will be presented in section 6.

Figure 9 shows an example of the analysis of abnormal indications conducted by our proposed approaches by a complementary method. The graph shows the prediction results using the two presented models against the same labeled test data. The upper part shows the results obtained by the spatial learning approach. The spatial model predicts the changed status of the future data patterns as normal to normal (N-N), normal to abnormal (N-A), abnormal to normal (A-N), and abnormal to abnormal (A-A). The lower parts



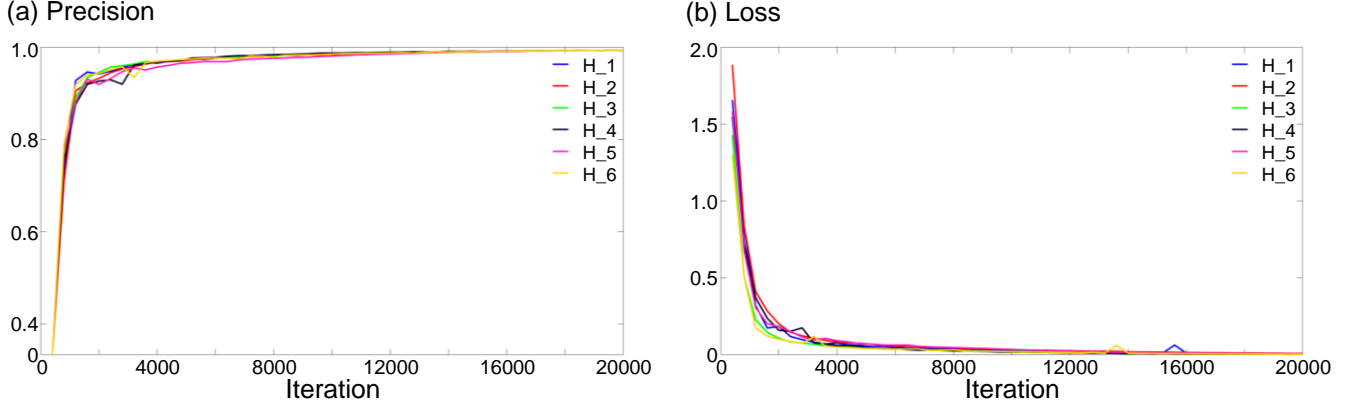


Figure 8: Results of the training on the spatial learning model (best viewed in color)

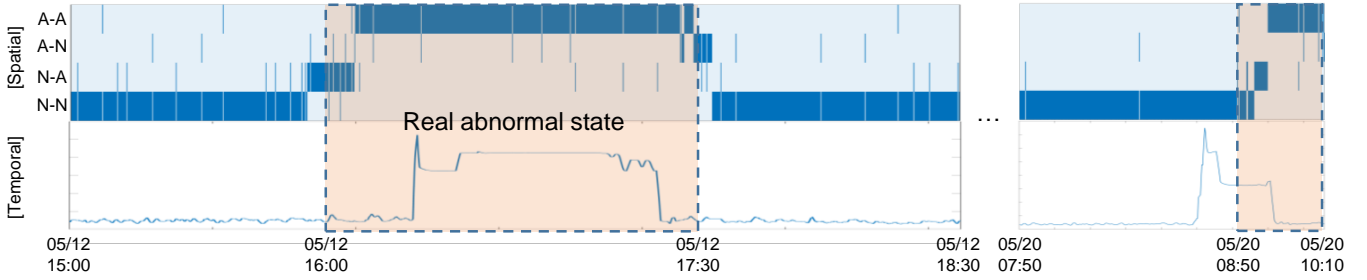


Figure 9: Examples illustrating the complementarity of the two (temporal and spatial) learning approaches, change of state N-N: from normal to normal, N-A: from normal to abnormal, A-N: from abnormal to normal, and A-A: from abnormal to abnormal

shows the results from the temporal learning approaches, which indicate the MSE between the predicted and real values of the target KPIs. The dashed boxes in orange indicate the periods of known critical issues. As shown in this figure, the two approaches are complementary. For instance, on 05/12 at 16:00, the spatial approach detected an issue even before the real issue occurred as an N-A case, whereas the temporal approach detected it after the real issue occurred. In contrast, in case of the issue on 05/20 at 08:50, though the spatial learning approach predicted it after it actually occurred, the issue can be forecasted in advance by the temporal learning approach. This experimental result and the necessary training data for each approach exhibited that both the approaches can contribute to predictive maintenance in a complementary manner.

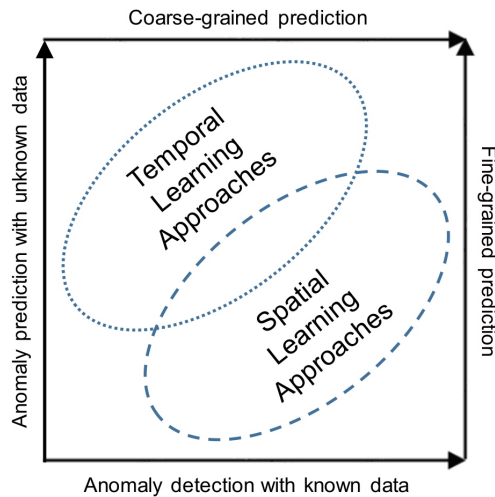
## 6. DISCUSSION

We have introduced two complementary DL approaches to predict the anomalies for a proactive maintenance of a DBMS. A temporal learning approach was presented to detect an abnormal pattern much different from the past KPI data, whereas a spatial learning approach was presented to classify similar patterns with labeled abnormal situations.

We extracted the most relevant KPI dataset of the HANA system during the preprocessing of the learning process and trained the proposed models using the real-world KPI data from the large-scale productive HANA systems. From the experimental results, we confirmed that our learning models could learn and extract the data-driven temporal features

and spatial features of the KPIs. In addition, they could analyze the anomalies successfully against the real-world database systems. We believe that these approaches will improve the overall system availability by detecting proactively in advance an unwanted situation, and that more benefits can be achieved from cloud systems, where thousands of DBMSs are operated simultaneously.

To apply the proposed models to real-world enterprise systems, it is necessary to consider a plan for an integrated system to ensure the best-possible performance. From the results of the previous experiments, it can be seen that the proposed learning models have excellent performance for the in-sample (training) error and out-sample (testing) error for a single HANA system, but the out-sample error of heterogeneous HANA systems is lower than in-sample error. This is because, in general, each database system runs on a physically different hardware and could have different workloads from applications. To supplement this limitation, in the short term, we consider ensemble learning to integrate both the proposed learning models, whose abstract concept is displayed in Figure 10. In the long term, we consider continuous learning, which continually learns new features and accumulates knowledge, to deal with the unseen anomalies continuously. Meta learning can be considered to learn the global patterns from multiple systems; it can then be deployed to real-world enterprise systems to learn more system-dependent data patterns with limited training data within a short time period [56, 17]. We also consider integrating our approaches to the HANA unavailability check sys-



**Figure 10: Abstract concept of the application of the two learning approaches**

tem to learn the newly found abnormal situations from the productive systems without the interpretation and labeling by DB administrators. It can be integrated with continuous learning to learn these new anomalies systematically. Thus, we want to present a concrete blueprint for a new integrated database management system with artificial intelligence to automatically update and analyze.

## 7. REFERENCES

- [1] SAP HANA External Machine Learning Library. <https://help.sap.com/viewer/ab6b04eb12d3452aa904d5823416a065/2.0.02/en-US>, 2017.
- [2] SAP HANA Predictive Analysis Library. <https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.02/en-US>, 2017.
- [3] SAP HANA Cockpit. [https://help.sap.com/viewer/p/SAP\\_HANA\\_COCKPIT](https://help.sap.com/viewer/p/SAP_HANA_COCKPIT), 2018.
- [4] S. Basu and M. Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154, 2007.
- [5] A. E. Ben-Nakhi and M. A. Mahmoud. Cooling load prediction for buildings using general regression neural networks. *Energy Conversion and Management*, 45(13-14):2127–2141, 2004.
- [6] Y. Bu, T.-W. Leung, A. W.-C. Fu, E. Keogh, J. Pei, and S. Meshkin. Wat: Finding top-k discords in time series database. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 449–454. SIAM, 2007.
- [7] J. B. Cabrera, L. Lewis, and R. K. Mehra. Detection and classification of intrusions and faults using sequences of system calls. *Acm sigmod record*, 30(4):25–34, 2001.
- [8] V. Chandola, V. Mithal, and V. Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*, pages 743–748. IEEE, 2008.
- [9] S. Chaudhuri and V. Narasayya. Self-tuning database systems: a decade of progress. In *Proceedings of the 33rd international conference on Very large data bases*, pages 3–14. VLDB Endowment, 2007.
- [10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.
- [12] D. J. Dean, H. Nguyen, and X. Gu. Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In *Proceedings of the 9th international conference on Autonomic computing*, pages 191–200. ACM, 2012.
- [13] D. Endler. Intrusion detection. applying machine learning to solaris audit data. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, pages 268–279. IEEE, 1998.
- [14] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.
- [15] E. Eskin, W. Lee, and S. J. Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX’01. Proceedings*, volume 1, pages 165–175. IEEE, 2001.
- [16] P. F. Evangelista, P. Bonnisone, M. J. Embrechts, and B. K. Szymanski. Fuzzy roc curves for the 1 class svm: Application to intrusion detection. In *Proceedings International Joint Conference on Neural Networks, IJCNN*. Citeseer, 2005.
- [17] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.
- [18] G. Florez-Larrahondo, S. M. Bridges, and R. Vaughn. Efficient modeling of discrete events for anomaly detection using hidden markov models. In *International Conference on Information Security*, pages 506–514. Springer, 2005.
- [19] A. W.-C. Fu, O. T.-W. Leung, E. Keogh, and J. Lin. Finding time series discords based on haar transform. In *International Conference on Advanced Data Mining and Applications*, pages 31–41. Springer, 2006.
- [20] S. Fu and C.-Z. Xu. Exploring event correlation for failure prediction in coalitions of clusters. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, page 41. ACM, 2007.
- [21] B. Gao, H.-Y. Ma, and Y.-H. Yang. Hmms (hidden markov models) based on anomaly intrusion detection method. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 1, pages 381–385. IEEE, 2002.
- [22] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. 1999.

- [23] A. K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *Workshop on Intrusion Detection and Network Monitoring*, volume 51462, pages 1–13, 1999.
- [24] A. K. Ghosh, A. Schwartzbard, M. Schatz, et al. Using program behavior profiles for intrusion detection. In *Proceedings of the SANS Intrusion Detection Workshop*. Citeseer, 1999.
- [25] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.
- [26] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.
- [27] G. Hamerly, C. Elkan, et al. Bayesian approaches to failure prediction for disk drives. In *ICML*, volume 1, pages 202–209, 2001.
- [28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [29] D. J. Hill and B. S. Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014–1022, 2010.
- [30] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- [31] T. Hong and S. Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016.
- [32] T. Hong, J. Wilson, and J. Xie. Long term probabilistic load forecasting and normalization with hourly information. *IEEE Transactions on Smart Grid*, 5(1):456–462, 2014.
- [33] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.
- [34] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.
- [35] E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Data mining, fifth IEEE international conference on*, pages 8–pp. Ieee, 2005.
- [36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [37] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [38] N. D. Le, R. D. Martin, and A. E. Raftery. Modeling flat stretches, bursts outliers in time series using mixture transition distribution models. *Journal of the American Statistical Association*, 91(436):1504–1515, 1996.
- [39] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [40] J. Lee, H. Kim, J. Lee, and S. Yoon. Transfer learning for deep learning on graph-structured data. In *AAAI*, pages 2154–2160, 2017.
- [41] K. Lee, Y. Cha, and J. Park. Short-term load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 7(1):124–132, 1992.
- [42] W. Lee, S. J. Stolfo, and P. K. Chan. Learning patterns from unix process execution traces for intrusion detection. In *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56, 1997.
- [43] J. Lin, E. Keogh, A. Fu, and H. Van Herle. Approximations to magic: Finding unusual medical time series. In *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on*, pages 329–334. IEEE, 2005.
- [44] C.-N. Lu, H.-T. Wu, and S. Vemuri. Neural network based short term load forecasting. *IEEE Transactions on Power Systems*, 8(1):336–342, 1993.
- [45] J. Ma and S. Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618. ACM, 2003.
- [46] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1741–1745. IEEE, 2003.
- [47] C. Marceau. Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the 2000 workshop on New security paradigms*, pages 101–110. ACM, 2001.
- [48] C. C. Michael and A. Ghosh. Two state-based approaches to program-based anomaly detection. In *Computer Security Applications, 2000. ACSAC’00. 16th Annual Conference*, pages 21–30. IEEE, 2000.
- [49] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [50] A. Nairac, N. Townsend, R. Carr, S. King, P. Cowley, and L. Tarassenko. A system for the analysis of jet engine vibration data. *Integrated Computer-Aided Engineering*, 6(1):53–66, 1999.
- [51] N. M. Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.
- [52] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan. Ganesha: Blackbox diagnosis of mapreduce systems. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):8–13, 2010.
- [53] D. C. Park, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg. Electric load forecasting using an artificial neural network. *IEEE transactions on Power Systems*, 6(2):442–449, 1991.
- [54] H. Plattner. A common database approach for oltp and olap using an in-memory column database. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1–2. ACM, 2009.
- [55] R. Powers, M. Goldszmidt, and I. Cohen. Short term performance forecasting in enterprise systems. In

- Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 801–807. ACM, 2005.
- [56] A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. *Advances in distributed and parallel knowledge discovery*, 3:81–114, 2000.
- [57] I. Psaroudakis, F. Wolf, N. May, T. Neumann, A. Böhm, A. Ailamaki, and K.-U. Sattler. Scaling up mixed workloads: a battle of data freshness, flexibility, and scheduling. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 97–112. Springer, 2014.
- [58] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.
- [59] Y. Qiao, X. Xin, Y. Bin, and S. Ge. Anomaly intrusion detection method based on hmm. *Electronics letters*, 38(13):663–664, 2002.
- [60] F. Salfner, M. Lenk, and M. Malek. A survey of online failure prediction methods. *ACM Computing Surveys (CSUR)*, 42(3):10, 2010.
- [61] S. Salvador and P. Chan. Learning states and rules for detecting anomalies in time series. *Applied Intelligence*, 23(3):241–255, 2005.
- [62] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
- [63] C. Shallahamer, J. Alkema, T. Gorman, and J. Still. *Forecasting Oracle Performance*. Springer, 2007.
- [64] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.
- [65] M. Stonebraker and U. Cetintemel. ”one size fits all”: an idea whose time has come and gone. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 2–11. IEEE, 2005.
- [66] P. Sun, S. Chawla, and B. Arunasalam. Mining for outliers in sequential databases. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 94–105. SIAM, 2006.
- [67] B. K. Szymanski and Y. Zhang. Recursive data mining for masquerade detection and author identification. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 424–431. IEEE, 2004.
- [68] S. Vemuri, D. Hill, and R. Balasubramanian. Load forecasting using stochastic models. In *Proceeding of the 8th power industrial computing application conference*, volume 1, pages 31–37, 1973.
- [69] D. Vlamis and T. Vlamis. *Data Visualization for Oracle Business Intelligence 11g*. McGraw-Hill Education Group, 2015.
- [70] L. Wei, E. Keogh, and X. Xi. Saxually explicit images: Finding unusual shapes. In *Data Mining, 2006. ICDM’06. Sixth International Conference on*, pages 711–720. IEEE, 2006.
- [71] W.-S. T. WST. Deeply learned face representations are sparse, selective, and robust. *perception*, 31:411–438, 2008.
- [72] J. Yang and W. Wang. Cluseq: Efficient and effective sequence clustering. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 101–112. IEEE, 2003.
- [73] C.-C. M. Yeh, N. Kavantzaz, and E. Keogh. Matrix profile iv: using weakly labeled time series to predict outcomes. *Proceedings of the VLDB Endowment*, 10(12):1802–1812, 2017.
- [74] X. Zhang, P. Fan, and Z. Zhu. A new anomaly detection method based on hierarchical hmm. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT’2003. Proceedings of the Fourth International Conference on*, pages 249–252. IEEE, 2003.