ProGAN: Toward a Framework for Process Monitoring and Flexibility by Change via Generative Adversarial Networks*

Maximilian Hoffmann**[1,2], Lukas Malburg**[1,2], and Ralph Bergmann[1,2]

[1] Artificial Intelligence and Intelligent Information Systems, University of Trier, 54296 Trier, Germany
{hoffmannm,malburgl,bergmann}@uni-trier.de
http://www.wi2.uni-trier.de

[2] German Research Center for Artificial Intelligence (DFKI) Branch University of Trier, Behringstraße 21, 54296 Trier, Germany
{maximilian.hoffmann,lukas.malburg,ralph.bergmann}@dfki.de

Abstract Monitoring the state of currently running processes and reacting to deviations during runtime is a key challenge in *Business Process Management (BPM)*. The MAPE-K control loop describes four phases for approaching this challenge: Monitor, Analyze, Plan, Execute. In this paper, we present the ProGAN framework, an idea of an approach for implementing the monitor, analyze, and plan phases of MAPE-K. For this purpose, we leverage a deep learning architecture that builds upon *Generative Adversarial Networks (GANs)*: The discriminator is used for monitoring the process in its environment by using sensor data and for detecting deviations w. r. t. the desired process state (monitor phase). The generator is used afterwards for analyzing the detected deviation and its symptoms as well as for adapting the current process to resolve the deviation and to restore the desired state. Both components are trained together by utilizing each other's feedback in a self-supervised way. We demonstrate the application of our approach for an exemplary scenario in the manufacturing domain.

Keywords: Business Process Prediction, Generative Adversarial Networks, Flexibility by Change, Process Adaptation

1 Introduction

Business Process Management (BPM) [3] is a research field of high interest with different areas of usage, e. g., in companies, in public administration, or in smart environments [23] such as smart homes or smart factories (e. g., [11, 13, 14, 21]). Especially environments that are characterized by high context sensitivity and

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-94343-1_4

** These authors contributed equally to the work.

huge amounts of real-time data can benefit from the advantages of BPM solutions [7]. One such key advantage is the ability to react to changes in the environment flexibly [11] to satisfy process performance indicators or to improve the processes. In order to check whether the actual state of an executed process in the environment is accordant with its desired state w. r. t. its process model, MAPE-K (*Monitor*, *Analyze*, *Plan*, and *Execute* phases as well as the knowledge *K* [6]) control loops can be implemented [22, 23]. Several approaches that can be applied for the monitoring and analyze phases of the MAPE-K control loop utilize deep learning techniques and, primarily, monitor processes based on predictions of the next event or the outcome of a process (e. g., [10, 18–20, 24, 25]). Other approaches based on deep learning (e. g., [15]) or based on other AI techniques (e. g., [14,23,28]) can be used for the plan phase by adapting the processes to the current environmental context by inserting, deleting, or modifying the activities of a process. In this paper, we propose to link the monitor phase with the analyze and plan phases to resolve deviations between the actual and desired state during process execution in order to reduce the manual effort for the user. For this purpose, we present a new framework, called ProGAN, for applying *Generative Adversarial Networks (GANs)* [5] to monitor the current environment by sensors and to detect deviations during process execution as well as to further analyze and plan process adaptations to enable (momentary) flexibility by change [26] for resolving at runtime. GANs, or deep learning approaches in general, are a suitable approach to tackle the described problem due to their learning capabilities that reduce the effort of knowledge acquisition compared to other AI approaches. We also want to leverage the learning capabilities by closely linking the monitor, analyze, and plan phases with the components of the GAN architecture. The aim of this joint integration is to improve the performance of discriminator and generator with the reciprocal feedback of each other. The paper is structured as follows: Section 2 shows foundations of our semantic graph representation, GANs, and MAPE-K control loops and, additionally, presents related work. Further, Sect. 3 presents our concept for using GANs for process prediction (monitor phase) and flexibility by change (analyze and plan phases). Section 4 presents a specific explanation of our proposed framework in the context of a real-world exemplary scenario from the manufacturing domain. Finally, Sect. 5 concludes the paper and shows areas of future work.

2 Foundations and Related Work

The foundations comprise an introduction to our process representation format, i. e., semantically annotated graphs (see Sect. 2.1), Generative Adversarial Networks (GANs; see Sect. 2.2), and MAPE-K control loops (see Sect. 2.3). We also cover related work from the fields of (predictive) process monitoring and adaptation, papers with and without a deep learning background (see Sect. 2.4).

2.1 Semantic Graph Process Representation

In our work, we use semantically annotated graphs named *NEST* graphs introduced by Bergmann and Gil [2] to represent processes. In contrast to other process representations such as Petri nets or BPMN 2.0, semantic graphs allow to model complex semantic information for its nodes and edges as well as to represent state changes during the production by using explicitly modeled data nodes, e. g., for intermediate products. Nevertheless, the presented concept is independent of the underlying process representation and, in fact, we use the de facto standard workflow modeling language BPMN to model the processes and afterwards convert them into NEST graphs.

NEST graphs consists of four elements: *N*odes, *E*dges, *S*emantic descriptions, and *T*ypes. Figure 1 illustrates a data-driven sheet metal manufacturing process represented as a *NEST* graph with semantic annotations. One type of nodes are task nodes that represent manufacturing operations in the process (e. g., `Burn`) and another type are data nodes that specify the material that is processed or produced during the execution of a task (e. g., `Sheet Metal`). The set of edge types consists of control-flow edges that define the execution order in a process (e. g., the task `Unload from Warehouse` is executed before the `Pick Up from Warehouse and Transport to Oven` task). Another edge type are data-flow edges that specify the flow of the material in the manufacturing process, for example, the `Burn` task processes the unprocessed steel slab and produces a rolled piece of sheet metal of a certain size and thickness. Semantic descriptions can be used to attach specific information to nodes and edges. They are based on a semantic metadata language (e. g., ontologies [9]) that expresses domain-independent and domain-dependent knowledge. In the *NEST* graph illustrated in Fig. 1, semantic descriptions are used to express properties of the tasks (e. g., the status of the resource that executes the manufacturing operation or the concrete parameter settings) or of the data nodes (e. g., the position of the material or the concrete properties such as the size). By using *NEST* graphs, it is possible to express processes as semantically annotated graphs.

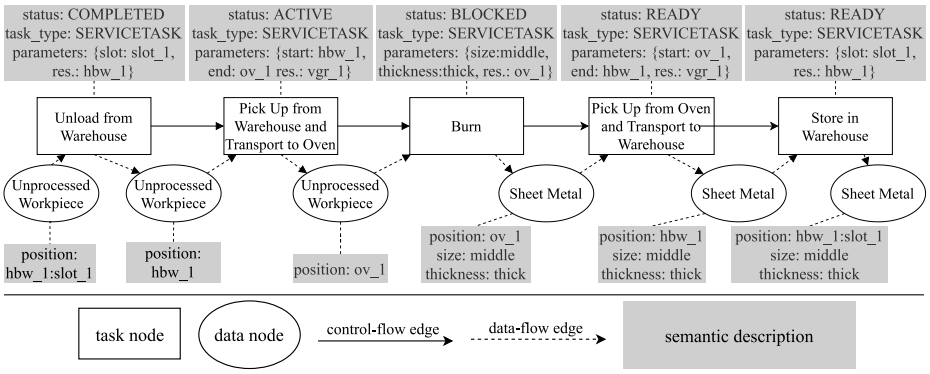

Fig. 1. Exemplary Semantic Graph representing a Sheet Metal Manufacturing Process.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) introduced by Goodfellow et al. [5] leverage an adversarial training procedure of two deep learning models, i. e., a *Generator* G and a *Discriminator* D. Training both models recreates a two-player game situation where D tries to correctly distinguish generated samples of G from original training examples. At the same time, G tries to generate samples from random input noise that are classified as training examples by D. The classification output of D is usually a binary value (sample comes from the training examples or not). This training strategy has the advantage that both parts are trained in a self-supervised way, leading to a generator G that generates samples matching the training data distribution and a discriminator D that learns to detect samples from the training data distribution.

2.3 MAPE-K Control Loops

MAPE-K control loops [6] provide a blueprint for self-managing information systems (see Fig. 2). The entity of interest is the managed element such as, in the example, a process (cf. [22, 23]).
The MAPE-K cycle consists of four phases that are designed as a continuous loop:

1. *Monitor*: The environment in which the managed element is executed is continuously monitored by using sensor data. In case deviations are detected in this data (*Symptoms*), the data is further analyzed.
2. *Analyze*: In the analyze phase, the current situation and the symptoms are checked in more detail. Based on this, a *Change Request* is created.
3. *Plan*: Based on the *Change Request*, the plan phase creates a *Change Plan* with corresponding actions. Applying these actions to the managed element leads to the process no longer deviating from the desired state.
4. *Execute*: Eventually, the *Change Plan* is executed in the environment in which the managed element operates.

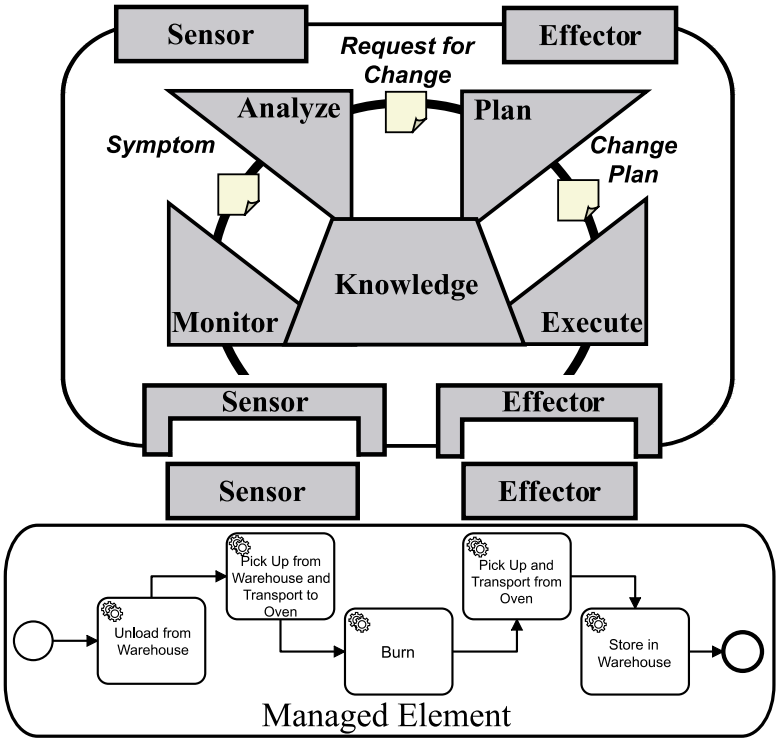

Fig. 2. MAPE-K Control Loop for Processes (Based on: [6] and [23])

Knowledge is shared between the methods of all phases. Relevant knowledge for autonomic systems can be topology information, historical logs, metrics, symptoms, and policies.

2.4 Related Work

Related work for our paper covers approaches based on deep learning (e. g., [4, 10, 15, 18, 19, 24, 27]) and based on other techniques (e. g., [14, 23, 28]) for business process monitoring and adaptation of processes. Rama-Maneiro et al. [19] provide a survey of different deep learning techniques, benchmarked in predictive process monitoring tasks on multiple process logs. See their work for a systematic overview from which we want to highlight some approaches in the following: Tax et al. [24] and Evermann et al. [4] apply *Long-Short Term Memory Neural Networks (LSTMs)* to perform business process monitoring, in particular next activity prediction based on event logs. In [10], Kratsch et al. compare deep learning and classical machine learning approaches in the context of business process monitoring by predicting process outcomes on publicly available event logs. Poll et al. [18] present their framework for proactive process forecasting, what can be seen as a paradigm of proactively predicting future process behavior rather than reactively responding to happened events. In [15], Metzger et al. approach the task of process monitoring with a novel online reinforcement learning setup where they learn when to trigger alerts based on process performance indicators. Weinzierl et al. [27] present an approach of prescriptive business process monitoring where process predictions of currently executed processes are examined regarding *Key Performance Indicators (KPIs)*. The presented approach utilizes a learning method that can be used for recommending beneficial actions that are optimized regarding certain KPIs. A new type of approach in context of predictive process monitoring is presented by Taymouri et al. [25]: They use *Generative Adversarial Networks (GANs)* and show that the respective adversarial learning paradigm outperforms other approaches based on non-adversarial neural network architectures. In their work, the trained generator can be used for process predictions, e. g., predicting the next label given a process trace.

In [14], Marrella et al. present the *SmartPM* system that detects deviations during process execution. Their system is motivated by knowledge-intensive processes such as in emergency management scenarios with structured processes and ad-hoc exceptions. To resolve deviations during process execution, SmartPM uses automated planning techniques that reduce the gap between the physical reality and the expected reality. Similar to this is the *PROtEUS* system presented by Seiger et al. [22, 23] that enables self-healing processes in smart homes by compensating occurred mismatches in the environment by searching for other resources that can perform the required activity. Similar to our proposed approach, they also apply MAPE-K control loops to cyber-physical processes in order to check the synchronization and consistency of them in smart environments. Wieland et al. [28] present an approach for using situation-aware adaptive workflows in the manufacturing domain. In addition to the standard workflow model, situational workflow fragments are manually constructed that define which actions should be performed in certain real world contexts to adapt the currently running process.

Compared to our self-supervised learning approach, the presented ones require sophisticated domain knowledge such as for automated planning (e. g., [14])

or need manually modeled compensations (e. g., [28]) to resolve deviations in the managed environment. In addition, the single phases of the MAPE-K loop are considered separately, whereas we combine the monitor with the analyze and plan phases in one approach to enable self-supervised learning.

3 Implementing MAPE-K Control Loops with GANs

This section contains the definition of our ProGAN framework that uses GANs in an implementation of the MAPE-K control loop. In this section, we first elaborate on the generic GAN architecture including its training procedure and relevant data. Given this foundation, we explain how this procedure is integrated in ProGAN and what characterizes the data in our GAN setup (see Sect. 3.1). Afterwards, we discuss design and application of the trained generator and discriminator w. r. t. the MAPE-K control loop (see Sect. 3.2).

3.1 Training Procedure and Data

We use a similar training procedure as in the generic setup (see Sect. 2.2) and apply it to data from the context of business processes. Figure 3 visualizes the GAN's training procedure and components with data being depicted as cylinders and data-flow as arrows. We also redefine the purpose of the discriminator that is used to monitor running processes as managed elements in the MAPE-K loop as well as the generator that is used to analyze the observed deviation and to generate adapted processes that are in line with the desired state, i. e., resolve the deviation (see Sect. 3.2). A central aspect in our framework is the concept of deviation, which is important for the training procedure, all involved data, and the relation to MAPE-K control loops. We do not technically define this concept but rather harness the capabilities of GANs to automatically learn to recognize deviations between desired and observed state of the managed element and how to resolve them. This also allows detecting deviations at different granularities which can be handled by using different neural network models for generator and discriminator, coupled with the respective datasets. For instance, a deviation can be on the process level where a certain constellation of tasks indicates an undesired state. Deviations can also occur on the task level when observing undesired resource allocations or misconfigured task parameters.

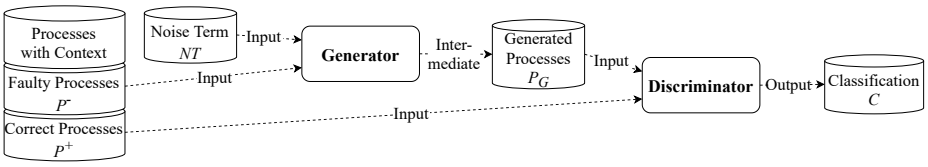

Fig. 3. Training Procedure and Data of the GAN Architecture.

The base for learning in the ProGAN framework is given by the training data $P = P^- \cup P^+$. As illustrated in Fig. 3, the training data serves as input to the discriminator and the generator. Our approach uses traces of executed processes represented as semantic graphs (see Fig. 1) as training data where each process also contains contextual data about its environment. For instance, in smart environments this contextual data could consist of sensor data related to the process execution (e. g., states of machines, light barriers). In addition, an ontology of a smart factory (cf. [9]) could provide further data to be integrated into the learning process. In this regard, the approach benefits from the deep learning infrastructure and its ability of automatically learning patterns. This allows to include a wide range of information with a reduced expert knowledge acquisition effort, which partly limits other approaches (e. g., automated planning techniques [14] or manually modeled compensations [28]). Further, the training data is split-up into two groups of processes, i. e., processes with corresponding contextual data in which these managed elements show deviations from their desired state (P^-) and those that do not (P^+). The groups serve different purposes in the training procedure. Thereby, P^- is used as a generator input to enable the generator to learn how to resolve deviations and P^+ as a discriminator input to enable the discriminator to learn how processes without deviations are characterized.

Closely related to P is P_G. P_G has the same shape as the training data, i. e., executed processes, but has a different semantic. It stems from the generator that has the goal of reducing the deviation between desired state and actual state. Since the generator learns to generate adapted processes without deviations given data from P^-, P_G contains processes that are in line with the desired state. Besides P^-, the generator also uses additional input data NT for generating processes from P_G. NT is a noise term that is the only input of the generator according to the GAN's original definition [5] (see Sect. 2.2). We also integrate this component into our learning procedure in order to allow a certain amount of randomness for the generated data. As demonstrated in literature (e. g., [16]), this combination of contextual data – processes and the environmental state in our case – and random noise serves as a good input for suitable training generalization and enables new, unseen generated samples.

The discriminator takes P^+ and P_G as input and produces the output C. We reused the GAN's original definition [5] of C which is a binary classification of each example, indicating whether this example is from the training dataset or was created by the generator. Implicitly, this enables the discriminator to learn how processes without deviations look like, i. e., P^+, and what distinguishes them from generated processes, i. e., P_G. In a converged learning state, the discriminator should not be able to distinguish both types of processes which indicates a trained generator and a trained discriminator.

3.2 Usage of Generator and Discriminator in MAPE-K Control Loops

After being trained, the generator and discriminator can be used for different tasks regarding the MAPE-K control loop (see Sect. 2.3). The basic setup is as follows: The discriminator is used for monitoring the managed element by using sensor data as input besides the state of the process (monitor phase). Based on detected deviations, the generator is used for an in-depth analysis of the *Symptom* of the deviation and generates an adapted process to reduce the occurred deviation between the observed and the desired process state in the environment (analyze and plan phase). Afterwards, a *Workflow Management System (WfMS)* is used to execute the resulting adapted process. The joint integration of discriminator and generator in a framework to implement MAPE-K control loops distinguishes our approach from previous proposed ones where no direct integration between monitoring, analysis, and the plan phase is examined (see Sect. 2.4). The aim of this joint integration is to improve the performance of discriminator and generator with the reciprocal feedback of each other. We expect better results of this setup, compared to standalone methods for each phase that are not applied or trained in combination.

Discriminator: We apply the discriminator in the ProGAN framework for monitoring processes in their environment by using sensor data and to check whether the processes correspond to their desired state (see monitor phase). This refers to the goal of detecting deviations between the desired state and the currently observed state during process execution (similar to [22, 23]). Thereby, the prediction result is a binary indication which corresponds to the *Symptom* that the current process shows a deviation (1) or not (0). Since a binary indication does not provide information on the kind of deviation observed, we only propose to use the fast inference of the discriminator in the monitor phase.

Generator: In the ProGAN framework, we propose to apply the generator in the analyze and in the plan phases. The generator is employed as a single entity to be used in both phases at once since planning an adapted process that resolves the deviation also requires the prior analysis of weaknesses. How the current deviation can be analyzed and how the desired state can be restored by an adapted process, is automatically learned during offline training of the GAN. A limitation of this single entity is that the request for change is not explicitly defined, but rather implicitly reflected in the change plan, i. e., the adapted process, at the end of the plan phase. This hinders explainability of the change plan and the performed process adaptations since no separate cause of the deviation is given. However, these problems are also faced in many other DL-based approaches, and they are already considered in research (e. g., [8]).

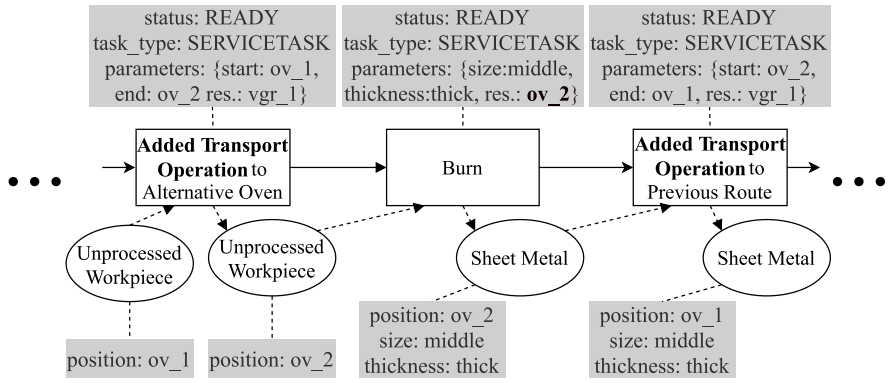

Fig. 4. Exemplary Adapted Process of the Generator with Modified Task Sequence.

4 Application Scenario: Using GANs to Implement MAPE-K Control Loops

This section presents an application scenario of the ProGAN framework in the manufacturing domain. Our use case deals with structural adaptations in which tasks or whole process fragments need to be adjusted for correct execution and with substitutional adaptations in which parts of the semantic description, e. g., parameter settings of an activity, need to be modified (see [1, pp. 225-230] for more details on adaptation types). We refer to the already presented manufacturing process from Fig. 1: It is assumed that the oven that performs the `Burn` activity is not available (e. g., defective; see status "blocked" of the task node). In such a case, there may be a need for more than just a simple modification of the parameter configuration. The reason for this is that a second alternative oven is located at a different position as the planned oven. In addition to changing the parameters of which resource (in this case oven) performs the corresponding activity, other transport routes that are valid for the current situation must also be added accordingly and the semantic correctness considering the subsequent tasks must be ensured, i. e., the positions of the workpieces that are captured by the data nodes must be considered. Figure 4 depicts an excerpt of the adapted process from Fig. 1 with two inserted transportation operations before and after the burn activity to re-establish the semantic correctness (i. e., correct positioning of the workpieces) of the manufacturing process as well as the change of the resource for the `Burn` activity (both in bold font).

The training data P in this use case stems from a repository that stores previously executed processes. These processes contain contextual information on the utilization of the machines and states of the resources in the factory throughout the execution of the process. In addition, a domain ontology of the model factory [9] is provided as input data in order to make knowledge about the factory layout such as transport routes, machines, and sensor positions available. P is split-up into P^+, i. e., successfully executed processes, and P^-, i. e., not

properly executed processes, according to the learning procedure in Sect. 3.1. The generator is fed with processes from P^- and the contextual information in order to generate processes, i. e., P_G, that reflect the desired state. We propose to use a definition where the generator generates an adapted process that corresponds again to the desired state by resolving the deviation. The discriminator has the goal to distinguish between processes without misconfigured tasks from P and processes with (possibly) misconfigured tasks from P_G. C is used originally as a binary indicator for each individual task.

We propose to use both components after training together in a MAPE-K control loop where the discriminator is applied in the monitor phase and the generator is applied in the analysis and plan phases. The goal of the discriminator is to monitor whether the process deviates from its desired state or not. For this purpose, the discriminator could use the provided contextual data such as the current utilization and states of the factory resources. In case a deviation during the process execution is detected, the generator analyzes the current situation and tries to solve the problem in the plan phase by generating an adapted process based on the deviated process. In this context, the generator should create an adapted process where two process fragments are inserted, i. e., the new transport routes, and the resource in the burn activity is modified, i. e., by replacing the current activity with the old parameter configuration with a new one. If the generator is not able to generate an adapted process that leads to a resolution of the deviation, a repository of process fragments resulting from inductive learning methods (see [17] for more details) can also be used by the generator for problem-solving or as additional training data.

5 Conclusion and Future Work

We presented the ProGAN framework as a first step toward an approach for using *Generative Adversarial Networks (GANs)* as an implementation of MAPE-K control loops. We apply the discriminator for business process monitoring of the managed element and its environment and the generator for enabling momentary flexibility by change in the analyze and plan phases of the MAPE-K control loop. Thereby, both components of the GAN, i. e., the generator and the discriminator, are trained together in a self-supervised way by using data in form of executed processes along with contextual data, e. g., state of machines, sensor data. We further described the application of the discriminator in the monitor phase of MAPE-K and the generator in the analyze and plan phases. Consequently, we demonstrated this application for a real-world BPM scenario from the manufacturing domain. Both the generator and the discriminator show potential of being used in similar scenarios.

In future work, we intend to further extend and implement the ProGAN framework as well as provide a technical description of certain implementation scenarios with well-defined data and functions of our GAN components. We also want to extend the discriminator to perform multi-label classifications, providing a more detailed analysis of the detected deviation which is suitable to be used

in combination with the generator in the analyze and plan phases of MAPE-K. This also enables to use discriminator and generator separately after combined training. Furthermore, we also want to show the feasibility of our proposed framework in an experimental evaluation. As already shown in our demonstration, we want to focus on manufacturing processes that are used in the context of our Fischertechnik physical factory simulation model[3] at the University of Trier [9, 11, 13]. This allows us to evaluate against other approaches for process monitoring and our already developed methods for process adaptations [17]. Additionally, we want to compare and combine our ProGAN framework with other approaches to enable flexibility by change such as automated AI planning techniques (e. g., Use Case 2 in [11] or [14]). The combination of the predictions of the discriminator with video-based methods and complex event processing of sensor data to implement a multi-modal process monitoring tool seems to be promising [12].

References

1. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications, LNCS, vol. 2432. Springer (2002)
2. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Inf. Syst. **40**, 115–127 (2014)
3. Dumas, M., et al.: Fundamentals of Business Process Management. Springer (2018)
4. Evermann, J., Rehse, J., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**, 129–140 (2017)
5. Goodfellow, I.J., et al.: Generative Adversarial Nets. In: Adv. in Neural Inf. Processing Syst. 27. pp. 2672–2680 (2014)
6. IBM: An architectural blueprint for autonomic computing: Autonomic Computing White Paper (2006)
7. Janiesch, C., et al.: The Internet of Things Meets Business Process Management: A Manifesto. IEEE Syst. Man Cybern. Mag. **6**(4), 34–44 (2020)
8. Keane, M.T., et al.: How Case-Based Reasoning Explains Neural Networks: A Theoretical Analysis of XAI Using Post-Hoc Explanation-by-Example from a Survey of ANN-CBR Twin-Systems. In: Proc. of 27th ICCBR. LNCS, vol. 11680, pp. 155–171. Springer (2019)
9. Klein, P., Malburg, L., Bergmann, R.: FTOnto: A Domain Ontology for a Fischertechnik Simulation Production Factory by Reusing Existing Ontologies. In: Proc. of the Conf. LWDA. vol. 2454, pp. 253–264. CEUR-WS.org (2019)
10. Kratsch, W., et al.: Machine Learning in Business Process Monitoring: A Comparison of Deep Learning and Classical Approaches Used for Outcome Prediction. BISE (2020)
11. Malburg, L., et al.: Using Physical Factory Simulation Models for Business Process Management Research. In: BPM Workshops, LNBIP, vol. 397, pp. 95–107. Springer (2020)
12. Malburg, L., et al.: Object Detection for Smart Factory Processes by Machine Learning. Procedia Comput. Sci. **184**, 581–588 (2021)

[3] https://iot.uni-trier.de

13. Malburg, L., Klein, P., Bergmann, R.: Semantic Web Services for AI-Research with Physical Factory Simulation Models in Industry 4.0. In: Proc. of the Int. Conf. on Innov. Intell. Ind. Prod. and Logist. (IN4PL). pp. 32–43. SCITEPRESS (2020)
14. Marrella, A., Mecella, M., Sardiña, S.: Intelligent Process Adaptation in the SmartPM System. ACM Trans. Intell. Syst. Technol. **8**(2), 25:1–25:43 (2017)
15. Metzger, A., et al.: Triggering Proactive Business Process Adaptations via Online Reinforcement Learning. In: BPM, LNCS, vol. 12168, pp. 273–290. Springer (2020)
16. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. CoRR **abs/1411.1784** (2014)
17. Müller, G.: Workflow Modeling Assistance by Case-based Reasoning. Springer (2018)
18. Poll, R., et al.: Process Forecasting: Towards Proactive Business Process Management. In: BPM. LNCS, vol. 11080, pp. 496–512. Springer (2018)
19. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. CoRR **abs/2009.13251** (2020)
20. Rehse, J.R., Mehdiyev, N., Fettke, P.: Towards Explainable Process Predictions for Industry 4.0 in the DFKI-Smart-Lego-Factory. KI **33**(2), 181–187 (2019)
21. Schönig, S., et al.: IoT meets BPM: a bidirectional communication architecture for IoT-aware process execution. Softw. Syst. Model. **19**(6), 1443–1459 (2020)
22. Seiger, R., Aßmann, U.: Consistency and synchronization for workflows in cyber-physical systems. In: Proc. of the 10th ACM/IEEE Int. Conf. on Cyber-Phys. Syst. pp. 312–313. ACM (2019)
23. Seiger, R., Huber, S., Heisig, P., Aßmann, U.: Toward a framework for self-adaptive workflows in cyber-physical systems. Softw. Syst. Model. **18**(2), 1117–1134 (2019)
24. Tax, N., et al.: Predictive Business Process Monitoring with LSTM Neural Networks. In: Adv. Inf. Syst. Eng. vol. 10253, pp. 477–492. Springer (2017)
25. Taymouri, F., et al.: Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In: BPM. LNCS, vol. 12168, pp. 237–256. Springer (2020)
26. van der Aalst, W.M.P.: Business Process Management: A Comprehensive Survey. ISRN Softw. Eng. **2013**(1), 1–37 (2013)
27. Weinzierl, S., et al.: Prescriptive business process monitoring for recommending next best actions. In: BPM Forum, LNBIP, vol. 392, pp. 193–209. Springer (2020)
28. Wieland, M., et al.: Towards situation-aware adaptive workflows: SitOPT - A general purpose situation-aware workflow management system. In: Int. Conf. on Pervasive Comput. and Commun. Workshops. pp. 32–37. IEEE (2015)