Deployment architecture for the local delivery of ML-Models to the industrial shop floor

Andreas Backhaus, Andreas Herzog, Simon Adler and Daniel Jachmann

Fraunhofer Institute for Factory Operation and Automation, IFF
Magdeburg, Germany
`andreas.backhaus@iff.fraunhofer.de`

Abstract. Information processing systems with some form of machine-learned component are making their way into the industrial application and offer high potentials for increasing productivity and machine utilization. However, the systematic engineering approach to integrate and manage these machine-learned components is still not standardized and no reference architecture exist. In this paper we will present the building block of such an architecture which is developed with the ML4P project by Fraunhofer IFF.

1 Introduction

The current industrial revolution challenges companies to optimize production to face scarcity resources and climate change. Initiatives like “Industrie 4.0” should accelerate the required technological innovations and already deepened the understanding about the value of data. Schnieder et al. [1] have written in 1999 that data is a required production resource. Nowadays, data becomes also part of the production result that can be used during operation.

Using data is currently a challenge during production. Even if the companies have identified the potentials of data [2], it is challenging to gain the added values. One reason is that IT-knowledge is in most cases not a key competence of companies in industrial production. To achieve benefits from data-driven machine learning (ML) it is required to analyse the data to gain information e.g. about optimizable parameters or aspects for predictive maintenance. There are multiple algorithms for ML that differ in their requirements, weaknesses and strength and experts were required to utilize ML. Currently there are increasingly software libraries, tools, and frameworks available which hide complex numerical processing, required in ML. Even though most systems try to provide easy access to ML, the outnumbering set requires some overview about the systems and expert knowledge, too. Additionally, there is currently a disagreement regarding data security. Some Frameworks allow the use of cloud-based services but this requires a continuous data-stream to the cloud. Some companies prefer to keep their data local to avoid knowledge transfer and a theoretical possible exposition of the production system.

2 Aim of the presented work

We present a deployment architecture for the local delivery of ML to the industrial shop floor. The architecture is already used during operation in the industry but is still work in progress.

The architecture allows to perform ML locally and without the requirement to deliver data via public domain, even if it also could be used as cloud-service or together with cloud-services. The architecture consists of a server locally to the production unit. It manages the connectivity to the machines programmable logic controller (PLC), collects signals and bypasses signals to the ML-Module. Input signals are verified, and the ML Model is monitored; thus, it can be guaranteed that invalid results are detected. Additional server connectors provide access to available knowledge systems (e.g. ERP, PDM, PLM) and the web-based client. It provides an assistance system for workers, so they can query and collect data during daily routine. ML analytics requires that the digital information model is up to date. The web client gives workers the possibility to keep the information model up to date during their work process.

Using ML during production requires not only the collection of data for training and analysis. Production has additional requirements that are practically motivated or required by law. We therefor present and discuss the requirements raised up during conception and implementation of the presented architecture.

3 Related Work

As the ML paradigm has moved from a scientific exercise to a tool for industrial data processing, the need has arisen to export or describe ML-models independent of their initial creation framework. All large frameworks, some of them not in existents only a couple of years ago, now offer a model description format as well as runtimes for scoring the described model. A model is basically described as graph of computing steps to arrive at an output value from an input value while the training process determined the free parameters of the model in order to minimize a set loss function. As field progresses, the framework provider for training the ML model not necessarily needs to be the provider of the runtime system for scoring the model in production nowadays also referred as ‘model serving’. Examples for model descriptions are xml-based file formats like PMML [3] or binary formats like ONNX [4] or Tensorflow Graphs [5].

In industrial production the company must document their machine configurations, so they can prove the operation conditions at any time. Changes during maintenance are documented with version and revision. If ML is influencing industrial production, they must be documented in a comprehensible representation, too. The advantage of model descriptions is that components of a numerical system can be described in a form that is independent of the model version of a particular learning framework. It furthermore increases the readability and transparency of a numerical model since all computing steps are described (comprehensible).

However, due to fast moving nature of the field of machine learning, not one standardized way of describing models has established yet, even when efforts are being made

[6]. Furthermore, the modules of numerical processing systems are increasingly built by a number of different participants with their favourite frameworks. Due to this fact, we argue for a generalized model serving description that can integrate numerical transformation descriptions from different sources.

4 Architecture

ML is currently discussed as integrated part of controllers from technical systems, to dynamically adapt technical systems to unknown production conditions. ML-models that decide about operation parameters due to analysed data, yield to technical problems and law-relevant questions about responsibilities. These problems can be avoided, if the operator remains responsible for the technical system but is utilizing ML as part of an assistance system. The assistance system has a server and a client part. The server is a communication hub, which connects to different knowledge bases. Each connection translates between an internal protocol and the proprietary protocol of the connected knowledge base.

Figure 1: Simplified Structure. The assistance server is a communication hub, unifying protocols. Signals provided by plc are stored a connected database and can be analysed online. ML-models are provided as part of the machine documentation by a connected PDM system.

The machines plc provides the signal data (sensor, actor, states) and is one knowledge base. Like MES, the assistance server monitors the signal data and stores the data into internal databases, but can intercept signal-streams for online analyses. The resultant values are stored as additional soft sensor. While simplest implementations of such an interception are thresholds or equations, ML methods allow evaluating signal values against trained models.

The machines signals will change their behaviour over time e.g. of wear and tear. Therefore, the models must adapt and may only be conditionally valid. It is very likely that during failure, signals arise that never were trained and are not part of the model.

Even if ML-models can extrapolate, they lose accuracy. Especially during critical operational states (unplanned failure) the machines operator must understand the trustworthiness of the models results.

Therefore, the ML-models requires monitoring on different levels. On the first level, the range and statistical parameters of input and output signals must be checked individually. If an input signal is out of range, because of sensor or connection fault, the system must warn and switch to a save mode. The save mode can be an alternative ML-model, which does not considers the input value; a more robust model, which persevere a single sensor failure, or if no more options exist, a controlled machine stop. On output side, the ML-model does not know the limits of the machine actuators (linear axis, motors, oven, etc.). To prevent the actuators from being destroyed, output signals must be verified too.

On second level, we consider that the training of ML-models consists only a part of the possible combinations of input values. All input combinations outside the training vectors may result in undefined model output. A one-class classifier [Pimentel2014] can be used to detect such an input.

The third level controls the stability in the current working point of the ML-model. To estimate the stability a small variation to the current input value (working point) is added and the model output is processed. Depend on the calculation speed, this can be done in time multiplex or on parallel ML-models. If the output variation exceeds a pre-defined range, the model can be classified as unstable.

All monitoring components added to the plain ML-model result in an industrial runtime with parameters described in an overall model description.

5 Data connectivity and collection

The assistance server collects signal data from connected plc. It can provide parts of the functionality of a manufacturing execution system (MES) but allows also analysing the signals data online.

Connection and signal collection may be difficult. PLC have the major task to execute the programmed logic. Sending (push) actively data from the PLC is a low priority task and is not executed if the plc logic requires resources. Alternatively, the assistance server can scan (pull) the PLCs memory for changes. No changes are required in the PLC, but the update frequency must be higher than the Nyquist frequency of the fastest signal monitored. Besides the signals used to operate all machine part, both approaches have the benefit that internal flags and error codes can be monitored additionally.

For larger machines, a bus-logger can mirror signals from the communication bus and sends signal values if they pass the bus. Because signals are not taken from the PLC, one must be aware that only control required signals are detected. Error signals are only send to the connected HMI, but not distributed over the communication bus. For ML analysis, it is important to know which signals are related to malfunctions or regular operation states. Using bus mirrors requires therefore querying data from multiple sources and synchronization strategies.

For small and medium enterprises, pulling signals can be the easiest and cheapest way to collect operational data. The number of monitored signals can be limited, if expert knowledge is considered. Experts can select relevant sensors. The number of signals can be reduced if dependencies between the signals are detected using ML. Reducing the number of signals allows a higher scan rate to pull sensor data. Additionally, it should be questioned which sensor values are relevant to describe the process. Data analysis provides information to influence some kind of reduced if dependencies between the signals are detected using ML. Reducing the number of signals allows a higher scan rate to pull sensor data. Additionally, it should be questioned which sensor values are relevant to describe the process. Data analysis provides information to influence some kind of process. Controlling a robotic system is a high frequency process while a logistic process is even slower and allows a lower update rate.

This is not true in general. Even in slow processes, high frequency events can be an indicator for important effects, but experts are most likely aware of this and can adjust signal capture strategies.

6 ML-Model Serving

For a reliable integration of machine learning models into the industrial application, a model description must be generated from the training framework and contain a description of all necessary numerical processing steps. Therefore, a general model description and a model-serving component is needed. In Figure 2 the classical ML-model description (a) is extended to incorporate the ideas and building blocks described below.

Figure 2: In its current form, a machine learned model is described as a single pipeline of pre-processing, model inference and subsequent post-processing (a); for industrial deployment, this pipeline must be generalized into a graph of transforms which describe the model deployment as a combination of the prediction model combined with monitoring and fallback models as

well as a number of transform for example for pre- and post-processing (b); each transform is stored with a unit test in order to check at runtime whether the runtime implementation is correctly calculating the transformation steps.

We can formulate the following requirements for this description:

a. Constancy: The implementation of a model serving should not change when a new model is deployed. Patching and updating of a machine serving component needs to be avoided. Saving parameters only is not sufficient, because a full description of the transformation function in the model description is required.

b. Flexibility: Along the machine-learning pipeline developed in ML4P, different participant should be given a number of supported modelling frameworks to describe the base models (or base transforms) in the serving structure, since one will not find one description or framework that will provide all functionality. Furthermore, core transformation descriptions should be exchangeable since the market of machine learning frameworks is in constant movement.

c. Data Interpretability: The serving description must incorporate Meta information about the expected input and output data. Each base model needs a Meta description too, since different partners along the ML4P pipeline generate them.

d. Transform Testability: No transformation is allowed into serving without passing a unit test with attached I/O test data, per transform.

7 Monitoring Strategies

The basic principal of machine learning is to model a system behaviour by observing the system and fitting a generic mathematical model to the observed data by means of numerical optimization. In comparison to a physical model, a machine learning model is only valid within the boundaries of the dataset it was trained on. A physical model is valid everywhere the laws of physics apply. This arises to the need to monitor the input and output data of an ML-model as well as to observe its behaviour. Figure 3 shows monitoring strategies. Monitoring is performed by one-class models that act as anomaly / novelty detectors [8].

Figure 3: Several monitoring strategies, which can be deployed to check whether a model is still working in a valid data environment.

1. **Monitoring single inputs:** A model input can originate from different sources, which should be modelled for their valid behaviour separately. This strategy is for spotting abnormalities in sensor inputs. A similar control system can monitor the model output to prevent actuator damages.
2. **Monitoring multiple inputs:** A monitoring model is checking the validity of the complete input vector in order to determine whether the input data is still within the boundary of the training data.
3. **Fall-back Model:** In case of an abnormality, a fall-back model is added to the model description in order to continue operations. A fall-back model could be a physical model, or a more transparent less parameter machine learning model.
4. **Parallel Model:** The input is processed with a parallel copy of the prediction model. The input is varied with in magnitude of the signal noise and used to measure the prediction model stability at the model's working point.

8 Lifecycle Management

ML Models are stored related to the structure of the technical system. A technical system has different types of bill of materials (BOM) that depict structure how parts of the technical system are connected in a specific view. In manufacturing the engineering and service BOM are motivated by spatial view, because they have their origin during

engineering. A BOM is a very common type of an ontology how parts of a technical system are related.

ML is used to analyse the behaviour of a technical system, but behaviour dependencies are currently not formalized very often. Therefor additional ontologies like SysML [9] may become a higher importance in the future.

Maintenance is the process to keep a technical system alive and may require replacing parts during the product life cycle. The replacement is most likely not identical to the original part. It maybe new and without wear and tear, a newer version, from a different distributor, or a completely new setup. What does that mean for a trained ML model?

After replacement the operation of a ML model must be considered as on trial. Using the monitoring, the ML model can provide a confidence value how much the operator may trust the values. This also allows reusing older models if models adapt during operation. After replacement with an identical part, the current ML model and an older version, which was used in the beginning of the parts life cycle, can be benchmarked against each other.

For a larger machine, multiple ML models may be used to for different functionalities of the machine. The operator requires some overview where he can trust the data driven decision support using ML. This requires that it is documented which signals of which parts are used as input and which parts, assemblies, or modules are influenced by the ML model result. As an example: Consider that there are cameras that monitor the heating of a material. The cameras maybe part of an assembly but the material is in a heater assigned to a different assembly. The ML model is using the image data to monitor the heating process. If a camera is replaced it may influence the utilized image data. From a technical point of view, the model is using data of the camera's assembly, but the effects will influence the decision making of the material heating.

The ML-model is associated with the parts from which it uses signal values. If a part is changed during maintenance, every model can be identified that may be compromised. Additionally, the association to all influenced parts is required, to inform the operator automatically for which part and function the derived result is temporarily invalid.

9 Discussion

Using ML to optimize production requires interdisciplinary work between ML experts and experts from industry. A sustainable integration of ML models requires the integration of models in the product life cycle and their maintenance. Operational personal must be able to understand not only results of the online analyses but also their dependencies and their trustworthiness.

We established the corner stones of a deployment architecture for data-driven models acquired by machine learning in an industrial machine environment. While products like personal assistants that incorporate some form of machine learned component are now distributed widely, not much standardization on the engineering process of deploying and managing machine learning models has been formulized. Here, the Fraunhofer

project “Machine Learning for Production (ML4P)” will serve as a testing ground for implementing the deployment methodology laid out in this paper.

10 Acknowledgement

This work was supported as a Fraunhofer LIGHTHOUSE PROJECT.

References

- [1] Eckehard Schnieder: Methoden der Automatisierung. Vieweg+Teubner Verlag, 1999.
- [2] M. Eisenträger, S. Adler, M. Kennel and S. Möser, "Changeability in Engineering," 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Stuttgart, 2018, pp. 1-8.
- [3] Specification PMML Format Version 4.3: <http://dmg.org/pmml/v4-3/GeneralStructure.html>
- [4] Specification ONNX Format: <https://github.com/onnx/onnx>
- [6] Martin Abadi et. al TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. White Paper <https://www.tensorflow.org/about/bib>
- [7] Sculley, D & Holt, Gary & Golovin, Daniel & Davydov, Eugene & Phillips, Todd & Ebner, Dietmar & Chaudhary, Vinay & Young, Michael & Dennison, Dan. (2015). Hidden Technical Debt in Machine Learning Systems. NIPS. 2494-2502.
- [8] A review of novelty detection. Marco A.F.Pimentel, David A.Clifton,Lei Clifton,Lionel Tarassenko. Signal Processing, Volume 99, 2014, Pages 215-249
- [9] Matthew Hause, The SysML Modelling Language, Fifteenth European Systems Engineering Conference, 2006

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.