The Road to an Open and Secure Data Exchange Infrastructure for SHM

HORST TRATTNIG and LUKAS BERBUER

Vallen Systeme GmbH, Buergermeister-Seidl-Straße 8, 82515 Wolfratshausen, Germany

ABSTRACT

Today, structural health monitoring systems consist of various hardware and software components that are not always well synchronized. The paper emphasizes the need and challenges to synchronize multi time-domain and multi physical-phenomena data acquisition systems.

On the way from a raw data sensor system to an independent decision-making edge device, there are still some challenges to be solved along the way. Identifying and combining technologies from foreign domains is the focus of the journey. Open and standardized communication protocols like OPC UA or MQTT are already well-established technologies with a high level of security. These communication protocols allow users to discover and receive structured data from compatible sensors and edge devices in the network. Moving the data processing and decision-making further to edge leads to reduced data transfer volume, lower latencies and improved privacy, security, and cost-effectiveness. This requires an extensible platform to embed custom feature extraction algorithms and to deploy trained machine learning models. ONNX, the open standard for machine learning interoperability, is a further puzzle piece on this journey.

The paper shows the status of work, ongoing developments in the open standard community like OpenAE and platform strategies for advanced analytics and AI.

MOVING TOWARDS DISTRIBUTED EDGE DEVICES

Today, structural health monitoring systems are large baskets of hardware and software components, scattered over the structures and are barely synchronized. If it comes to dynamic signals, known as “structure-born sound” or “acoustic emission” (AE), data processing chains are even more complex and demanding. The effort to synchronize multi time-domain and multi physical-phenomena data acquisition systems is a huge challenge. Sensors are on a transition from simple data sources to configurable and scalable edge devices – also defined as smart sensors [1][2]. They combine sensing, data processing, analysis, decision-making and communication capabilities.

By integrating data acquisition and processing into a single device, intelligent sensors eliminate the need for complex and resource-intensive data processing architectures, resulting in more efficient and streamlined data collection processes. The key benefits are:

1. Reduced latency: Edge processing enables real-time data analysis and decision-making at the source. By processing data locally, smart sensors can minimize latency and respond quickly to events or triggers. This is particularly crucial in time-sensitive applications, such as industrial automation, where immediate action is required based on sensor data.
2. Bandwidth optimization: Smart sensors often generate a significant amount of data. Moving computations and intelligence further to the edge reduces the amount of data transfer volume – which, in case of acoustic emission, can be intense (20 MB/s for 10 MHz à 16 bit). By extracting and transferring only relevant signal features, the data rate can be reduced to less than 1 MB/s. This optimization of data transmission minimizes bandwidth requirements, reduces network congestion, and lowers communication costs.

3. Enhanced privacy and security: Edge processing helps address privacy and security concerns related to sensitive data collected by smart sensors. Instead of transmitting raw data to external systems, edge processing allows for local data analysis, ensuring that sensitive information remains within the controlled environment of the edge device. This approach reduces the exposure of sensitive data to potential security breaches, enhances data privacy, and supports compliance with privacy regulations.
4. Offline operation and resilience: Edge processing enables smart sensors to operate autonomously even when connectivity to the central system or cloud is disrupted. By processing data locally, smart sensors can continue to perform critical tasks and make local decisions, ensuring system operation and resilience in environments with intermittent or unreliable network connectivity. This capability is particularly valuable in remote or harsh environments where continuous connectivity may not be guaranteed.
5. Real-time decision-making: Edge processing empowers smart sensors to make immediate decisions based on local data analysis. By embedding processing capabilities within the sensor itself, it can react to sensor readings without relying on external systems for analysis. This enables faster response times and enables smart sensors to trigger actions or alerts promptly, enhancing the overall efficiency and effectiveness of the system.
6. Scalability: Edge processing facilitates the scalability and distributed architecture of smart sensor networks. As the number of smart sensors increases, edge devices can handle processing tasks locally, distributing the computational load across the network. This decentralized architecture supports the scalability and flexibility required for large-scale deployments, allowing for the addition of new sensors without overwhelming the central system or cloud infrastructure.
7. Energy efficiency: Edge processing can contribute to energy efficiency in smart sensor applications. Frequent data transmission and unnecessary communication is avoided, leading to lower power consumption.

Communication protocols

Open and standardized communication protocols like OPC UA (Open Platform Communications Unified Architecture) or MQTT (Message Queuing Telemetry Transport) are already well-established technologies for efficient, reliable and secure data exchange in industrial applications. They can bring several benefits to AE systems, enhancing their capabilities and integration with industrial automation environments.

OPC UA is used in various industries and applications where secure and interoperable communication between devices and systems is required [3]. OPC UA incorporates robust security mechanisms, including encryption and authentication, which are crucial for protecting sensitive AE data. By leveraging OPC UA's security features, AE systems can ensure the confidentiality, integrity, and availability of data during transmission, preventing unauthorized access or tampering. OPC UA allows the creation of a standardized information model that defines the structure, properties, and behavior of AE data. This modeling capability ensures consistent representation of AE data across different systems and facilitates data interpretation and analysis.

MQTT is a simpler, lightweight publish-subscribe messaging protocol that operates on top of different network protocols. It enables efficient and reliable data exchange between IoT devices, sensors, gateways, and cloud platforms.

CASE STUDY: AE EDGE DEVICE

Edge processing is particularly sensible for acoustic emission (AE) devices due to the high data rates associated with AE monitoring. The high sampling rates of e.g. 10 MHz lead to data rates of 20 MB/s per channel. The amount of data can be reduced drastically if the data processing (mainly feature extraction) is moved towards the edge and only relevant signal features are transmitted. As shown in Figure 1, following integration use-cases can be derived from a common acoustic emission data pipeline:

1. **Use case A**: The AE system is used as a data acquisition unit and all data is transferred to a host PC (connected via USB or Ethernet). This is the traditional (centralized) approach.
2. **Use case B**: Signal features are computed directly on the device and transmitted via OPC UA or MQTT. Feature extraction involves identifying and selecting specific attributes or patterns from a signal that are relevant to a particular task or analysis. This approach makes sense, if the AE data must be merged with other sensor data for a combined evaluation and decision-making.
3. **Use case C**: Decision can be made solely based on AE signals/features. Trained machine learning models are deployed on the device for inference at the edge. Only the outputs of the model (and optionally the features) are transmitted via OPC UA or MQTT.

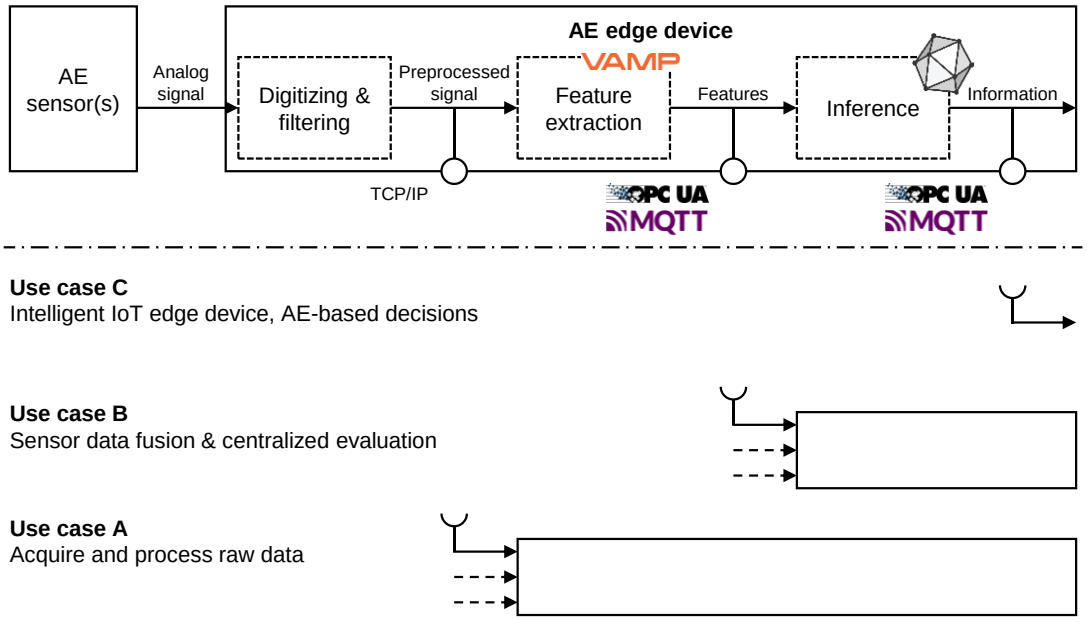

Figure 1. Integration use-cases of an acoustic emission edge device.

Challenge 1: Application-specific feature extraction algorithms

Feature extraction techniques in acoustic emission analysis can vary depending on the specific application and the characteristics of the signals being analyzed. Common methods include time-domain analysis (amplitude, duration, rise time, zero crossings), frequency-domain analysis (spectral centroid, bandwidth), time-frequency analysis (STFT, wavelet transforms), statistical features (variance, skewness, kurtosis) waveform/envelope analysis and pattern recognition algorithms [4][5][6][7]. These techniques aim to transform the raw acoustic emission data into a set of informative features that can be used for further analysis, classification, or anomaly detection.

An AE edge device with integrated feature extraction must be flexible and allow users to embed their own algorithms. A possible solution can be a plugin system like Vamp[1]. Vamp is a well-established audio processing plugin system that provides a standardized interface for analyzing audio data, but can also be used for AE and structure-borne sound signals. It allows developers to create plugins in C/C++ that can extract various types of information from signals. A lot of common algorithms are already implemented as Vamp plugins and available for download. Vamp plugins can also be used in Python environments[2].

Challenge 2: Machine learning interoperability

Machine learning models can be developed and trained in various frameworks, for example TensorFlow, PyTorch, Caffee, MATLAB, MXNet or Scikit-learn. How can those models be deployed on an edge device in a framework-agnostic manner?

ONNX [3], the open standard for machine learning interoperability, is a further puzzle piece on this journey. ONNX is an open format designed for representing machine learning models [8][9]. ONNX describes a model using a computational graph that represents the model's structure and operations. The computational graph is composed of nodes and edges, where nodes represent operations or computations, and edges represent the flow of data between nodes. ONNX allows models trained in one framework to be used in another framework without the need for extensive model reimplementation or conversion. Models saved in the ONNX format can be deployed and executed on a variety of platforms and devices, including cloud servers, mobile devices, edge devices, and specialized hardware accelerators, e.g. by using the ONNX runtime by Microsoft[4] (see Figure 2).

[1] Vamp (https://vamp-plugins.org). An open-source C++ SDK suitable for real-time applications is available via GitHub (https://github.com/lukasberbuer/rt-vamp-plugin-sdk).
[2] Vamp host Python bindings (https://pypi.org/project/rtvamp)
[3] ONNX (https://onnx.ai)
[4] ONNX Runtime (https://onnxruntime.ai)

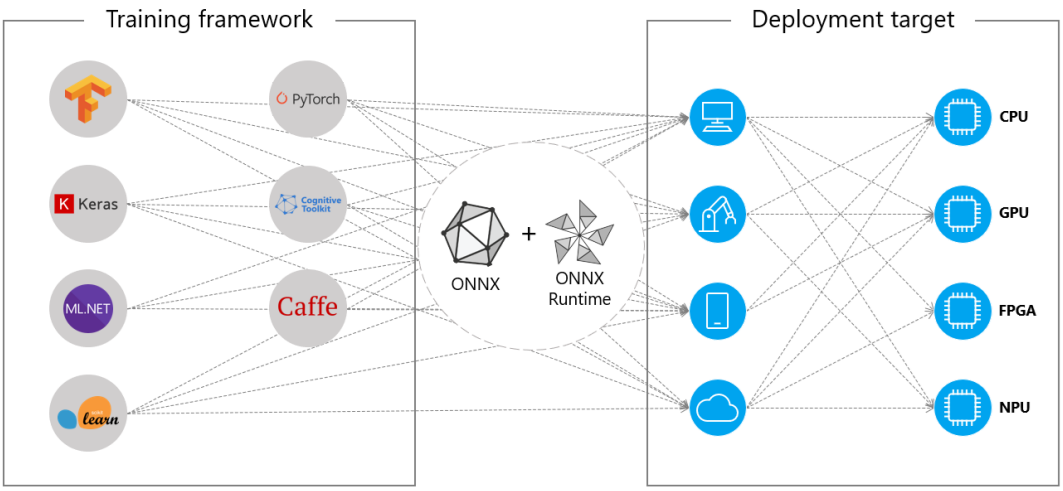

Figure 2. Machine Learning model interoperability with ONNX (by https://microsoft.com)

Overall, ONNX simplifies the process of model development, deployment, and collaboration by providing a common format that bridges the gap between different frameworks, tools, and deployment environments in the machine learning ecosystem.

Challenge 3: Mapping of features and model inputs

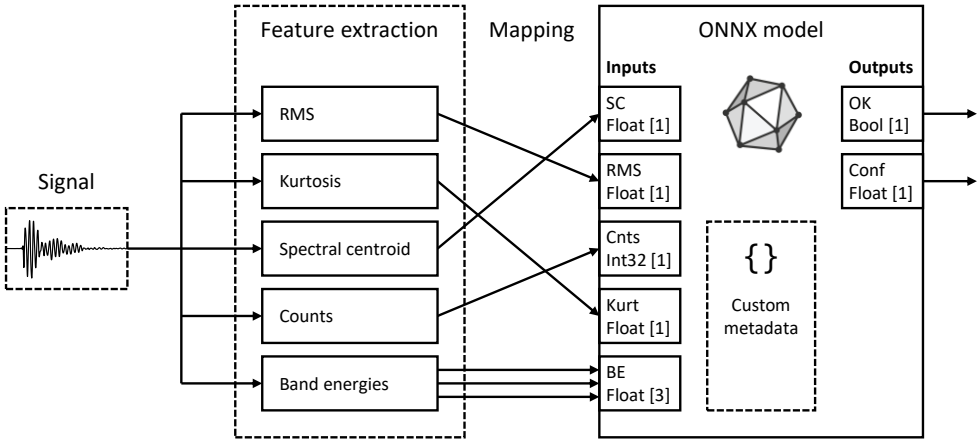

Figure 3. Mapping of features and model inputs with ONNX.

The inputs of the machine learning model must be mapped to the computed features – the feature vector (see Figure 3). This task is easy, if both the features and the model are developed and deployed in the same environment, e.g. with Python or MATLAB. If the model is exported as an ONNX model and deployed on another target, the input processing must be reproduced on the target device and mapped correctly to the model input. Otherwise, the model yields garbage data. ONNX provides named, typed and dimensioned inputs and outputs to simplify the mapping. Additionally, custom metadata can be stored in a model with additional setup information, e.g. in JSON format. This can be used to store the information of the full processing pipeline in the model: Acquisition settings, digital filters, FFT settings and the used feature extraction algorithms with its parameters.

FURTHER CHALLENGES AND GOALS

How can we facilitate advanced signal analysis and machine learning for acoustic emission applications? What do we need to utilize modern techniques like transfer learning and build monitoring solutions by using and adapting already existing models?

Transfer learning is a common practice in other domains like computer vision. With the rise of deep learning, modern computer vision models directly use the raw image data as the input without relying on handcrafted features or explicit domain knowledge. Only minor pre-processing (resizing, normalization and color space transformation) of the images is necessary to feed them into the deep learning model. Specific models can be built by utilizing a pre-trained model, such as ResNet, that has been trained on a large-scale dataset like ImageNet, which contains millions of images across various categories. Only the last layers (classification layers) of the model have to be re-trained for the new task. This approach allows the model to benefit from the large-scale pre-training dataset, even with limited labeled data. Overall, transfer learning accelerates training, improves model performance, and enhances generalization on the target task.

The world of SHM and AE-based monitoring is very different. Data sets of failure cases are usually rare and not sufficient to train models with the raw sensor data directly (deep learning). Instead, handcrafted features and domain knowledge are necessary to utilize machine learning. Overall, the signal path from the sensor to the decision-making is influenced by a lot of factors (Figure 4):

a) structure and dispersion
b) sensor sensitivity and mounting
c) acquisition settings, e.g. digital filters
d) feature extraction algorithms and parameters

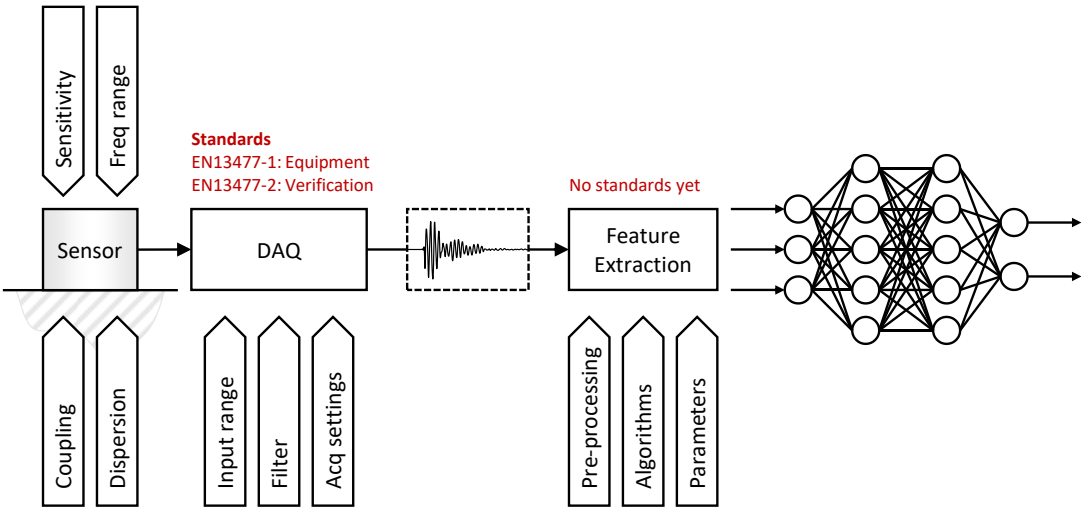

Figure 4. AE data and machine learning pipeline.

Time and frequency feature extraction is a crucial element in the signal path. Features like spectral centroid, band powers or ASL are widely used, but can differ in implementation details. Consequently, models fed with those features yield other results depending on the implementation of the mathematical algorithm.

We want to address this challenge with OpenAE[5], a community-driven project to empower data-based acoustic emission and structure-born sound applications (see Figure 5). The first goal is to build an open standard of feature algorithm definitions including reference implementations. Each feature is identified by a URI and a version. The model can use those identifiers to unambiguously specify its inputs. With reference implementations in different programming languages, those exact features can be computed in different environments and on different targets. Trained models based on those standardized features can be shared in the model zoo to be either reused in similar applications or new applications by utilizing transfer learning. GitHub is used as the main platform to allow easy and transparent collaboration between researchers, students and engineers. To join us and become part of the community, visit openae.io.

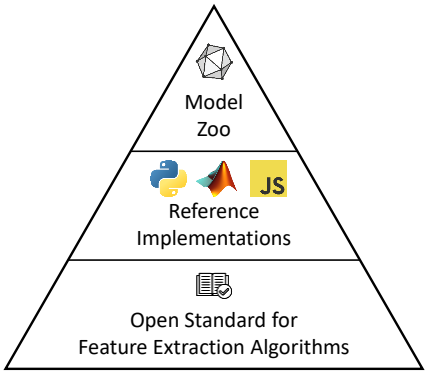

Figure 5. OpenAE building blocks for transferable AE analysis and models.

REFERENCES

1. Berns, K., Köpper, A., Schürmann, B., Berns, K., Köpper, A., & Schürmann, B. (2019). Sensordatenverarbeitung. *Technische Grundlagen Eingebetteter Systeme: Elektronik, Systemtheorie, Komponenten und Analyse*, 237-264.
2. Tränkler, H. R. (2014). Einführung in die Sensortechnik. *Sensortechnik: Handbuch für Praxis und Wissenschaft*, 3-20.
3. Leitner, S. H., & Mahnke, W. (2006). OPC UA–service-oriented architecture for industrial applications. *ABB Corporate Research Center*, *48*(61-66), 22.
4. Eyben, F. (2015). *Real-time speech and music classification by large audio feature space extraction*. Springer.
5. Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications* (Vol. 5). Springer.
6. Caesarendra, W., & Tjahjowidodo, T. (2017). A review of feature extraction methods in vibration-based condition monitoring and its application for degradation trend estimation of low-speed slew bearing. *Machines*, *5*(4), 21.
7. Sause, M. G. (2016). *In situ monitoring of fiber-reinforced composites: theory, basic concepts, methods, and applications* (Vol. 242). Springer.
8. Jajal, P., Jiang, W., Tewari, A., Woo, J., Lu, Y. H., Thiruvathukal, G. K., & Davis, J. C. (2023). Analysis of Failures and Risks in Deep Learning Model Converters: A Case Study in the ONNX Ecosystem. *arXiv preprint arXiv:2303.17708*.
9. Shridhar, A., Tomson, P., & Innes, M. (2020, August). Interoperating Deep Learning models with ONNX. jl. In *Proceedings of the JuliaCon Conferences* (Vol. 1, No. 1, p. 59).

[5] https://openae.io, https://github.com/openae-io