Master Thesis Planning Report

Tokenization as a Neural Compression Strategy in Automotive Embedded Systems

Tim Boleslawsky, gusbolesti@student.gu.se

Emrik Dunvald, gusdunvem@student.gu.se

February 2026

Suggested Supervisor at CSE: Yinan Yu

Suggested Supervisor at Company: Dhasarathy Parthasarathy

1 Background

The In-Vehicle Embedded System: An in-vehicle embedded system is a specialized computer system integrated within a vehicle to perform dedicated functions, often in real-time, and is essential for controlling, monitoring, and enhancing various automotive operations. These systems typically consist of both hardware and software components, such as electronic control units (ECUs), sensors, actuators, and communication interfaces, which are responsible for tasks like engine management, safety features, infotainment, and advanced driver assistance systems [Navet and Simonot-Lion, 2017, Fairley, 2019].

In-Vehicle Networks and Event-Triggered Logging: Modern vehicles may contain dozens or even hundreds of these embedded systems, interconnected through in-vehicle networks (e.g., CAN, LIN, FlexRay, Ethernet), enabling efficient communication and coordination among different vehicle subsystems [Bello et al., 2019, Navet and Simonot-Lion, 2017, Fairley, 2019]. Event-triggered logging and diagnostic frameworks, which record data only when anomalies or threshold crossings occur, are often adopted to reduce data transmission and avoid bus saturation in complex systems, such as the in-vehicle embedded system. However, this selective approach can reduce holistic visibility of system health, as it may miss subtle degradation patterns or early warning signs that do not cross predefined thresholds. This complicates the detection of emerging faults and comprehensive condition monitoring [Nunes et al., 2023, Montero Jiménez et al., 2020, Azar et al., 2022]. Additionally, the need to carefully tune event thresholds and diagnostic criteria introduces maintenance challenges, as improper settings can lead to missed events or excessive false positives, further complicating system upkeep and reliability [Nunes et al., 2023, Azar et al., 2022].

Downstream Machine Learning Tasks and Data Quantity: Two developments in recent years further underline the shortcomings of event-triggered logging in automotive systems: the massive increase in signal-based data in the in-vehicle network and the growing relevance of downstream ML tasks.

Recent industry and research reports indicate that the data quantity generated by ADAS (Advanced Driver Assistance Systems) sensors in vehicles is growing at an extremely rapid pace. According to a 2023 technical paper referencing McKinsey's 2021 automotive electronics report, by 2030, about 95% of new vehicles will be connected, up from around 50% today, and a single car can generate up to 1 terabyte (TB) of data per hour from its sensors [Bertoncello et al., 2021, Samantaray, 2023]. This explosive growth is driven by the increasing number and sophistication of sensors—such as cameras, radars, and lidars—required for advanced safety and autonomous driving features, with the complexity and volume of data presenting significant challenges for storage, processing, and transmission within embedded automotive systems [Samantaray, 2023].

Modern vehicles increasingly rely on data-driven intelligence to enhance safety, reliability, and efficiency. Beyond perception and control, downstream ML tasks — those leveraging collected vehicle and sensor data for offline analysis, optimization and predictive functions — have become central to automotive-system design. These tasks include predictive maintenance [Theissler et al., 2021], anomaly and intrusion detection [Özdemir et al., 2024], and fleet-level analytics like fuel consumption or maintenance scheduling [Chen et al., 2025].

Recent reviews highlight that while event-triggered and anomaly-based data collection can optimize resource use, they often result in fragmented or incomplete datasets, making it harder to implement robust predictive maintenance strategies and limiting the effectiveness of ML models that rely on continuous, high-resolution data streams [Nunes et al., 2023, Montero Jiménez et al., 2020]. Multi-model and hybrid approaches are being explored to address these limitations, but the trade-off between data reduction and diagnostic completeness remains a significant challenge in both industrial and automotive contexts [Montero Jiménez et al., 2020, Azar et al., 2022]. How modern research approaches this trade-off is discussed in more detail in Section **??**.

Traditional Compression: Traditional compression methods, based on information theory principles, often fall short in automotive applications, especially as a precursor for downstream ML tasks. For video/image compression, traditional methods like JPEG or MP3 are optimized for human perception (e.g., visual quality) rather than ML tasks or efficient downstream data use [Ma et al., 2019]. For time series data, algorithmic approaches like CHIMP or Gorilla depend on manually chosen parameters like window size and are sensitive to data characteristics such as entropy and signal variability. This limits their effectiveness in capturing the nuances required for accurate ML model performance in automotive contexts [Johnsson, 2025]. These algorithmic approaches were investigated by Johnsson [2025] in a previous Master's thesis project. This work builds upon this thesis by exploring an alternative approach to compressing vehicle telemetry data.

Rate-Utility Trade-off and Related Research: Constructing downstream ML models for automotive systems, or in fact Internet-of-Things (IoT) systems in general, is a constant trade-off between handling large quantities of data and maximizing model performance. Traditional compression techniques can reduce data volume, but often at the cost of losing critical information necessary for accurate ML tasks such as predictive maintenance, anomaly detection, and fleet analytics. The impact of this trade-off is well-documented in the literature. Muniz-Cuza et al. [2024], for example, study the impact of lossy compression techniques on time series forecasting tasks and observe a constant trade-off between compression ratio and forecasting accuracy.

Existing research approaches these challenges from three different angles: utility-aware adaptive telemetry, neural compression, and task-aware compression.

- First, utility-aware adaptive telemetry methods aim to employ policy learning methods to dynamically adjust telemetry parameters to reduce maintenance costs while preserving data utility for downstream tasks. Although this approach is still emerging, recent research has demonstrated promising results [Zhang et al., 2023].
- Second, neural compression techniques learn data representations optimized for both compression efficiency and ML task performance. This research is heavily inspired by deep generative models like GANs, VAEs, and autoregressive models, but focuses on compressing the data, instead of generating realistic data samples [Yang et al., 2022]. Neural compression techniques extend the introduced lossy compression principles in two key ways. First, they offer an alternative to traditional distribution modeling by leveraging deep neural networks to learn complex data distributions directly from the data, capturing intricate patterns and dependencies that traditional statistical models may miss. Second, they substitute traditional approaches to transform coding and quantization with learned representations [Yang et al., 2022]. Studies as early as 2019 have shown that neural compression methods can outperform traditional compression techniques for image and video data, especially at low bitrates [Löhdefink et al., 2019]. The same has been shown for time series data [Zheng and Zhang, 2023, Liu et al., 2024].
- Lastly, task-aware compression techniques focus on optimizing compression algorithms to retain information that is most relevant for specific tasks [Yang et al., 2022]. This idea has shown promise in handling time-series data more efficiently in IoT systems. Azar et al. [2020] and Sun et al. [2025] for example explore task-aware compression algorithms that adaptively prioritize data features based on their relevance to downstream tasks, demonstrating improved performance in resource-constrained environments.

2 Aim

The aim of this project will be to investigate the use of neural compression techniques, specifically tokenization-based approaches, for compressing automotive time series data. The goal is to develop a compression framework that effectively balances the trade-off between compression ratio and utility for downstream machine learning (ML) tasks, such as predictive maintenance and anomaly detection.

The motivation behind using tokenization as a neural compression strategy for automotive time series data is to produce discrete latent representations that simplify the entropy modeling task within the compression pipeline. By constraining the data to a finite set of tokens, the complexity of modeling the underlying data distribution is reduced, enabling the use of lightweight entropy models that are computationally efficient. This is particularly advantageous in automotive applications where computational resources are limited, and real-time processing is often required.

The goal will be to design, implement and evaluate a tokenization-based neural compression framework tailored for time-series data. Evaluation will be conducted by comparing relevant parameters between the proposed framework and traditional compression methods.

3 Problem Formulation

The problem to be solved with this thesis is that of efficiently compressing vehicle telemetry data for downstream machine learning tasks. The issue is that vehicles generate a vast amount of telemetry data from various sensors and systems, which can be very challenging to store and transmit due to constraints on edge devices and communication bandwidth [Samantaray, 2023]. While techniques for data compression do exist, they are either not optimized for the specific characteristics of vehicle telemetry data or do not take into account the requirements of downstream machine learning tasks.

Traditional compression techniques such as event-triggered logging and algorithmic time series compression methods have limitations when applied to vehicle telemetry data. Event-triggered logging can miss important information that does not cross predefined thresholds, leading to incomplete data for machine learning models [Nunes et al., 2023, Montero Jiménez et al., 2020, Azar et al., 2022]. Algorithmic compression methods often rely on manually chosen parameters and may not effectively capture the complex patterns present in vehicle telemetry data, which can negatively impact the performance of downstream machine learning tasks.

More recent approaches such as utility-aware adaptive telemetry and task-aware compression have shown promise in addressing some of these challenges. However, these methods often require complex policy learning or adaptive algorithms that may not be feasible for real-time applications in vehicles [Zheng and Zhang, 2023, Löhdefink et al., 2019, Kawawa-Beaudan et al., 2022, Liu et al., 2024].

One less explored but promising approach to compressing data which could be suitable for vehicle telemetry data is that of neural compression in combination with tokenization. A system which incorporates a lightweight neural encoder and a small tokenizer should in theory be able to produce a lossy low entropy representation of the data which can be efficiently stored and transmitted. The main challenge will be to ensure that the compressed representation retains as much relevant information as possible for downstream machine learning tasks while still maintaining a high compression ratio.

4 Limitations

Due to time constraints and the scope of this thesis, there will be several limitations to the work presented:

- The work will be limited to only testing a single head / type of loss for the neural compression model. While it would be interesting to explore multiple heads, this would be time consuming in testing when running all the model variations.
- The work will only implement the encoder and tokenizer part of the pipeline while forgoing the entropy coding. This is done due to the focus of the thesis being on the neural compression step and while entropy coding is important in a full compression pipeline, it is less relevant for the research question at hand.
- The work will also forgo testing on real hardware as that would be very time consuming and difficult to set up while contributing relatively little to the main research question.
- The work will be limited to only testing regression tasks and anomaly detection as downstream tasks due to time constraints and structure of the available datasets. These two tasks should however be a good representation of common downstream tasks and adding more tasks would likely not drastically change the conclusion of the thesis.
- Due to wanting to keep the compressed data representation as versatile as possible, no task-aware compression techniques will be implemented. While task-aware compression could potentially yield better performance for specific tasks, it would limit the generalizability of the compressed representation and add significant complexity to the work. It also results in a more relevant comparison to other compression techniques which are not task-aware.

5 Methodology

Within this section, we will discuss the methodology employed in our project to achieve the outlined aim. Specifically, we will discuss how the architecture of the baseline established compression model and the proposed tokenization-based compression framework are designed and implemented. We additionally discuss the experiment setup we will use to compare these two approaches.

5.1 Implementation Details

Baseline Model: First, we want to outline the architecture of the established compression method we will use as a baseline. We will use the autoencoder approach introduced by [Zheng and Zhang, 2023] as an inspiration for our autoencoder baseline model. It is important to note, that the paper introduces a sophisticated framework consisting of two models, the TCN-RNN model and the TCN-ARNN model, as well as a model selection mechanism. For simplicity sake, we will only focus on one of these models, the TCN-RNN model. The TCN-RNN model described in the paper is a representative of state-of-the-art continuous-latent neural time-series compression methods. Compared to vanilla autoencoders, the models architecture is explicitly designed for long temporal dependencies and has demonstrated stronger rate-distortion performance. This allows us to evaluate our tokenization-based approach against a competitive neural compressor rather than a toy model. Another point to note about the TCN-RNN model, is its costly architectural design, namely deep temporal convolutional stacks with large receptive fields, recurrent (often sequential) decoding, and dense continuous latent representations. These components typically lead to increased FLOPs, higher activation memory, and longer inference latency compared to discrete, token-based compression methods.

On a high level, the TCN-RNN architecture consists of the following components:

- Input handling and preprocessing
- TCN encoder
- RNN bottleneck
- RNN decoder
- TCN decoder

For the input handling and preprocessing it is important that the time-series data is segmented into fixed-length windows. These windows are treated as inputs to the model. Overlapping windows can be used to increase the effective training data size and improve reconstruction quality at window boundaries. Additionally, each channel is normalized independently using statistics computed on the training set.

The TCN encoder consists of a stack of Temporal Convolutional Network (TCN) blocks with increasing dilation factors. Each TCN block contains: A 1D convolution operating along the temporal dimension, normalization (BatchNorm or LayerNorm), a nonlinear activation function (ReLU or LeakyReLU), optional dropout for regularization, and a residual connection to stabilize training. The outcome is a continuous latent representation whose size is controlled by the downsampling factor and channel width.

The encoder output is processed by a recurrent neural network, specifically a gated recurrent unit (GRU), the RNN bottleneck. The RNN captures longer-range temporal dependencies that are not easily modeled by convolutions alone. At this stage, we produce a sequence of hidden states that serves as the core compressed signal. The hidden state dimensionality and number of layers define the strength of the bottleneck. The compression is achieved implicitly by restricting temporal resolution and latent dimensionality rather than by discretization. Afterwards, the RNN decoder reconstructs a latent sequence from the bottleneck representation. The decoder mirrors the bottleneck RNN in depth and hidden size.

Finally, a mirrored TCN stack upsamples the latent sequence back to the original temporal resolution. Transposed convolutions or interpolation-based upsampling are used. A final linear projection maps features back to the original channel dimension. The overall training objective is the reconstruction loss (e.g., MSE or MAE) between input and reconstructed signal.

Tokenization-Based Compression Framework: Next, we outline the architecture of our proposed tokenization-based compression framework. The architecture is inspired by recent advances in neural compression for speech and audio data, namely the Wavtokenizer paper. The key idea is to replace continuous latent representations with discrete tokens drawn from a learned codebook.

We again want to first outline the high-level components of the architecture:

- Input handling and preprocessing
- Lightweight encoder
- Vector quantization / tokenization
- Decoder head (for reconstruction) or downstream head (for task-specific loss)

The first component is (and should be) identical to the TCN-RNN baseline model. This ensures that any performance differences can be attributed to the compression method rather than preprocessing variations.

The encoder layer functionally has the same goal s the TCN encoder used in the baseline model, but has a different architecture. The encoder is designed to be lightweight, avoiding heavy temporal convolutions or recurrent layers. Instead, a shallow neural network (e.g., small CNN or MLP applied per timestep) maps the input signal to intermediate embeddings. The encoder implements the learned transform stage of a lossy compression pipeline. Its purpose is not to predict labels or reconstruct the input per se, but to reshape the data into a representation that can be efficiently discretized. Specifically, the encoder learns a mapping that removes redundancy and correlations present in the raw time series, concentrates important information into a low-dimensional, stable representation, and aligns the geometry of the representation space with the constraints of quantization. The objective is usually joined with the quantization and decoder stages.

In contrast to autoencoder-based methods, where the encoder is free to produce high-entropy continuous latents, the encoder here is explicitly shaped to produce quantization-friendly representations that support discrete tokenization and entropy coding.

The next component is the vector quantization / tokenization stage. Here, intermediate embeddings are discretized using a learned codebook. The codebook very simply looks like this: codebook = nn.Parameter(torch.randn(K, D)), with K = number of tokens and D = embedding dimension. Each embedding vector is replaced by the index of its nearest codebook entry. This step introduces the only explicit source of information loss. The output is a sequence of discrete token IDs drawn from a finite vocabulary.

Lastly, we implement a head to enable loss calculation and backpropagation. Depending on the use case, this head can be a decoder for reconstruction tasks or a downstream head for task-specific losses (e.g., classification or regression). The head architecture is flexible and can be adapted to the specific application. The calculated loss is then used to jointly optimize the encoder, codebook, and head parameters.

5.2 Experiment Setup

To compare and evaluate the two methods outlined above, we will conduct a series of experiments. In this subsection we outline how we ensure a fair comparison, what metrics we will use, and what data we will use for the experiments.

Comparison and Metrics: There are three dimensions along which we want to compare the two compression methods. First, we want to evaluate the *compression performance* of both methods. This includes the metrics compression rate and reconstruction error. These metrics align with the evaluation standards in the neural compression literature. We will measure these metrics by training the two models on a reconstruction task and evaluating the reconstruction quality and effective compression rate. Second, we want to evaluate the *downstream task performance* when using compressed representations from both methods. This includes metrics such as accuracy for classification tasks or mean squared error for regression tasks. This evaluation reflects the practical utility of the compressed data for real-world applications. Third, we want to compare the *model efficiency* of both methods. This includes metrics such as parameter count, computational cost (FLOPs), inference latency, and memory footprint. These metrics are important for assessing the feasibility of deploying the compression methods in resource-constrained environments.

An important note to make regarding comparability of these two metrics, especially regarding the downstream task performance, is that the two methods produce different types of compressed representations. The TCN-RNN baseline produces a continuous latent representation and the tokenization-based method produces discrete tokens. To ensure a fair comparison, we will need to design a lightweight unobtrusive adapter for each representation type that maps them into a common format suitable for the downstream model. The adapters should have minimal capacity to avoid introducing bias. This way, we can isolate the effect of the compression method itself on downstream performance. The envisioned design for the adapters is as follows:

- The adapter for the continuous latent representation will take the sequence of real-valued vectors output by the TCN-RNN model, average them over time to produce a single vector, and then pass this vector through a linear layer to match the input size of the downstream model.
- The adapter for the token-based representation will take the sequence of token IDs output by the tokenization model, map each token ID to a small vector using an embedding lookup, average these embeddings over time to produce a single vector, and then pass this vector through a linear layer to match the input size of the downstream model.

Data: The goal is to use industry relevant data, while also ensuring that the methods are tested on publicly available data. For that reason, we will use three datasets: Two Volvo datasets, the Volvo test fleet dataset and the Volvo electric mobility (EMOB) dataset, and a third publicly available dataset called X-CANIDS [Jeong et al., 2024]. Each of these datasets contains multivariate time-series data collected from vehicles, including sensor readings, telemetry data, and operational parameters. The test fleet dataset will be used to construct a downstream regression task focused on predicting a continuous target variable. The EMOB dataset will be used to construct a downstream classification task, namely anomaly detection in battery systems. The publicly available dataset will be used as an additional benchmark to validate the generalizability of the findings across different data sources.

6 Risk Analysis

To prepare for potential delays and challenges in the project, we have identified several risks that could impact our timeline and deliverables. These risks are:

- **Downstream model training takes longer than expected:** While there exist some fragments of downstream models within the Volvo environment, we will need to construct our own downstream models to test the compression methods. As this is not our main focus within this project, we hope to complete this relatively fast and with straightforward methods. But this, if e.g. data handling becomes more complicated than expected, this could become more time consuming than expected..
- **Baseline compression method is harder to implement than expected:** Fortunately [Zheng and Zhang, 2023] describe the implemented method in detail, but there is still a risk, that our implementation of the baseline compression method may need finetuning and adjustments to work properly with our data and downstream models. This could lead to delays in the project timeline.
- **Proposed method does not outperform baselines:** There is a risk that the proposed tokenization-based compression method may not outperform the baseline methods in terms of compression efficiency or downstream task performance. This could lead to challenges in demonstrating the value of our approach and may require additional iterations or adjustments to the method.
- **Proposed method does outperform baseline, but is computationally not better:** Even if the proposed method achieves better compression or downstream performance, it remains to be seen, if the method actually outperforms the baseline in terms of computational efficiency. This could limit its practical applicability and may require further optimization or adjustments to make it viable for real-world use cases.
- **Computational efficiency cannot be proven without doubt because actual hardware is not available for testing:** As the actual in-vehicle technology will not be available for testing, the

computational efficiency may be proven in theory not in practice. If the proposed method is actually computationally feasible in practice remains to be evaluated when it is deployed on actual hardware.

7 Timeline

We want to roughly divide the project into four stages, each with specific goals and deadlines. First, we will set up a working environment, and put time into understanding the problem and plan for the project. This would be **Stage 1: Setup and Preparation**. Within this stage we have the following sub-goals:

- Planning the project with appropriate subtasks and a timeline. Then writing the project plan.
- Creating a working environment. This include setting up necessary hardware (at Volvo), git repositories, and coding environments.
- Defining and Training a downstream ML model on uncompressed data that will serve as a benchmark for evaluating the performance of different compression methods. Right now, we aim for one regression and one classification task based on automotive telemetry data.
- Implementing a baseline model based on an established neural compression frameworks such as [Zheng and Zhang, 2023]. The planned architecture is discussed in section 5.

We envision, that this stage will take roughly five weeks and set a preliminary deadline for the 20th of February.

The second stage, **Stage 2: Development**, will focus on developing the core components of the proposed compression approach leveraging tokenization. This includes:

- Develop a compression framework that leverages tokenization. This includes implementing an encoder, a quantization method using a codebook, and a decoder as discussed in section 5.
- Conducting some first initial experiments to validate the implementation.
- Writing the halftime report.
- Start writing on the final thesis report. At this stage we aim to have the theory part of the report done.

The second stage should take around five weeks as well, with a deadline for the 27th of March.

After that, we will start **Stage 3: Experiments**, which will focus on evaluating the performance of the proposed compression method against the baseline model. This stage will include:

- Compressing the data using both the baseline compression method and the proposed tokenization approach.
- Training the downstream model on the compressed data using both compression methods.
- Evaluating the experiments by capturing the defined metrics for both compression methods.

This stage is expected to take three weeks, with a deadline for the 17th of April.

Finally, we will enter **Stage 4: Finalizing**, which will involve:

- Conducting an ablation study of the proposed tokenization compression framework.
- Writing the methodology, results, discussion, and conclusion part for the final report.

This stage is expected to take four weeks. We plan to have a first draft of the final report ready with a deadline for the 15th of May.

References

Joseph Azar, Abdallah Makhoul, Raphaël Couturier, and Jacques Demerjian. Robust iot time series classification with data compression and deep learning. *Neurocomputing*, 398, 02 2020. doi: 10.1016/j.neucom.2020.02.097.

Kamyar Azar, Zohreh Hajiakhondi-Meybodi, and Farnoosh Naderkhani. Semi-supervised clustering-based method for fault diagnosis and prognosis: A case study. *Reliability Engineering & System Safety*, 222: 108405, 2022. doi: 10.1016/j.ress.2022.108405.

L. L. Bello, R. Mariani, S. Mubeen, and S. Saponara. Recent advances and trends in on-board embedded and networked automotive systems. *IEEE Transactions on Industrial Informatics*, 15:1038–1051, 2019. doi: 10.1109/tii.2018.2879544.

Michele Bertoncello, Christopher Martens, Timo Möller, and Tobias Schneiderbauer. Unlocking the full life-cycle value from connected-car data. Technical report, McKinsey & Company, McKinsey Center for Future Mobility, Feb 2021. URL `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/unlocking-the-full-life-cycle-value-from-connected-car-data`. White paper.

Fanghua Chen, Hong Jia, and Wei Zhou. Vehicle maintenance demand prediction: A survey. *Applied Sciences*, 15(20), 2025. ISSN 2076-3417. doi: 10.3390/app152011095. URL `https://www.mdpi.com/2076-3417/15/20/11095`.

Richard E. Fairley. *Automobile Embedded Real-Time Systems*, pages 377–389. Wiley-IEEE Press, 2019. doi: 10.1002/9781119535041.app2.

Seonghoon Jeong, Sangho Lee, Hwejae Lee, and Huy Kang Kim. X-canids: Signal-aware explainable intrusion detection system for controller area network-based in-vehicle network. *IEEE Transactions on Vehicular Technology*, 73(3):3230–3246, 2024. doi: 10.1109/TVT.2023.3327275.

Simon Johnsson. Large scale efficient data readout for vehicle fleets. Master's thesis, Chalmers University of Technology, 2025.

Maxime Kawawa-Beaudan, Ryan Roggenkemper, and Avideh Zakhor. Recognition-aware learned image compression. *Electronic Imaging*, 34(14):220–1–220–5, January 2022. ISSN 2470-1173. doi: 10.2352/ei.2022.34.14.coimg-220. URL `http://dx.doi.org/10.2352/EI.2022.34.14.COIMG-220`.

Jinxin Liu, Petar Djukic, Michel Kulhandjian, and Burak Kantarci. Deep dict: Deep learning-based lossy time series compressor for iot data, 2024. URL `https://arxiv.org/abs/2401.10396`.

Jonas Löhdefink, Andreas Bär, Nico M. Schmidt, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. Gan- vs. jpeg2000 image compression for distributed automotive perception: Higher peak snr does not mean better semantic segmentation. *arXiv preprint arXiv:1902.04311*, 2019. doi: arXiv:1902.04311v1.

Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30:1683–1698, 2019. doi: 10.1109/tcsvt.2019.2910119.

Juan José Montero Jiménez, Sébastien Schwartz, R. Vingerhoeds, B. Grabot, and M. Salaün. Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, 2020. doi: 10.1016/j.jmsy.2020.07.008.

Carlos Enrique Muniz-Cuza, Søren Kejser Jensen, Jonas Brusokas, Nguyen Ho, and Torben Bach Pedersen. Evaluating the impact of error-bounded lossy compression on time series forecasting. In *Advances in Database Technology - EDBT*, number 3 in Advances in Database Technology - EDBT, pages 650–663. OpenProceedings, March 2024. doi: 10.48786/edbt.2024.56.

N. Navet and F. Simonot-Lion. *Automotive Embedded Systems Handbook*. CRC Press, 2017. doi: 10.1201/9780849380273.

P. Nunes, J. Santos, and E. Rocha. Challenges in predictive maintenance – a review. *CIRP Journal of Manufacturing Science and Technology*, 2023. doi: 10.1016/j.cirpj.2022.11.004.

Rojalin Samantaray. Adas sensor data handling in the world of autonomous mobility. *SAE Technical Paper Series*, 2023. doi: 10.4271/2023-01-0993.

Guoyou Sun, Panagiotis Karras, and Qi Zhang. Highly efficient direct analytics on semantic-aware time series data compression, 2025. URL `https://arxiv.org/abs/2503.13246`.

Andreas Theissler, Judith Pérez-Velázquez, Marcel Kettelgerdes, and Gordon Elger. Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. *Reliability Engineering & System Safety*, 215, 2021. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2021.107864. URL `https://www.sciencedirect.com/science/article/pii/S0951832021003835`.

Yibo Yang, S. Mandt, and Lucas Theis. An introduction to neural data compression. *Found. Trends Comput. Graph. Vis.*, 15:113–200, 2022. doi: 10.1561/0600000107.

Penghui Zhang, Hua Zhang, Yibo Pi, Zijian Cao, Jingyu Wang, and Jianxin Liao. Adapint: A flexible and adaptive in-band network telemetry system based on deep reinforcement learning. *IEEE Transactions on Network and Service Management*, 21:5505–5520, 2023. doi: 10.1109/tnsm.2024.3427403.

Zhong Zheng and Zijun Zhang. A temporal convolutional recurrent autoencoder based framework for compressing time series data. *Applied Soft Computing*, 147:110797, 2023. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2023.110797. URL `https://www.sciencedirect.com/science/article/pii/S1568494623008153`.

Övgü Özdemir, M. Tuğberk İşyapar, Pınar Karagöz, Klaus Werner Schmidt, Demet Demir, and N. Alpay Karagöz. A survey of anomaly detection in in-vehicle networks, 2024. URL `https://arxiv.org/abs/2409.07505`.