Revisions based on Reviewers' Comments

Paper: Learning Semantic Association Rules from Internet of Things Data

We would like to thank the reviewers for their constructive feedback! We have addressed the reviewer's comments and answered their questions as described below. In our revised **paper**, we used blue font for the revised parts.

For the rest of this document:

Blue tone: refers to the feedback given by the reviewers.
Green tone: describes our actions and answers.

Reviewer 1 - Andreas Martin

The paper presents a well-structured study on semantic association rule mining for IoT data, combining static knowledge graphs and dynamic sensor data. The proposed Autoencoder-based Neurosymbolic ARM method effectively reduces the number of rules while maintaining high data coverage. The experiments are rigorous and provide convincing evidence of the method's effectiveness.

Strengths:
- The integration of static and dynamic IoT data for association rule mining is a novel and well-motivated contribution.
- The evaluation is extensive, including comparisons with exhaustive and optimization-based ARM methods.
- The discussion on execution time, scalability, and variations of the method is thorough and relevant for future research.

We would like to thank the reviewer for the kind words and for highlighting the strengths of our paper.

1. The paper does not conform to the journal's layout and template. The formatting should be adjusted to align with the journal's requirements, including section numbering, citation style, and figure placement.

We are aware that our paper has a different template from the previously reviewed papers in the NeSy AI journal. We have contacted the editors and confirmed that there has been a template change, and the template that our paper uses is the correct one, as it follows the SAGE journal template on the Author Guidelines (https://neurosymbolic-ai-journal.com/content/author-guidelines) page of the NeSy AI journal (as of 26.03.2025).

2. The methodology and experimental setup are dense, making it difficult for non-expert readers to follow. More intuitive explanations or a simplified example pipeline would improve accessibility.

As suggested, we have updated the methodology (Semantic Association Rules from IoT Data) and experimental setup sections and made them easier to follow for non-expert readers.

The Semantic Association Rules from IoT Data section now contains both more intuitive explanations of each of the pipeline steps, as well as a running example that we continuously refer to at the end of each step (Data preparation, Training and Autoencoder Architecture, and Rule Extraction from Autoencoders steps).

The Evaluation section now starts with an intuitive explanation of the two experimental settings that we designed to evaluate the two contributions of our paper.

3. The study relies solely on one language model for evaluating the ability to extract rules. A brief discussion on whether results would generalize to other models would strengthen the impact.

We would like to clarify that we do not use any language model in any part of our study. The evaluation of our two contributions is done in 2 different experimental settings: i) we run multiple association rule mining methods (including ours) with and without semantic properties and compare the results to evaluate our semantic rule mining pipeline and to show how semantics can help learn generalizable association rules; ii) we compare ours and the state-of-the-art methods using standard association rule mining quality criteria (number of rules, data coverage, execution time, average support, confidence and association strength (Zhang's metric)) to show that our rule mining method learns a more concise set of high-quality association rules with full data coverage (addressing the well-known rule explosion problem).

4. The execution time analysis is detailed, but the discussion on real-world scalability could be expanded. The applicability of the method to large-scale IoT systems with a high number of sensors should be addressed.

As suggested, we expanded the discussion subsection on the real-world scalability of our approach:

"Real-world scalability. Real-world large-scale IoT data differs from the tabular datasets that most state-of-the-art ARM methods focus on. Each sensor is treated as a different data dimension, hence resulting in potentially extremely high-dimensional data especially upon including semantic properties. Therefore, utilizing neural networks' capability to process high-dimensional data is essential. Both time complexity and execution time analyses (Experiments 1.2 and 2.1) show that our Neurosymbolic rule mining approach is scalable on large-scale IoT data. Extrapolating the execution times (training + rule extraction) shown in Figure 6, Aerial can scale up to tens of thousands of sensors on a laptop (see Hardware) in a day. The training is linear over the number of features (sensor measurements and associated semantic properties) and the number of transactions, and the rule extraction stage is polynomial over the number of feature classes. Algorithm 1 is parallelizable as test vectors per feature subsets are created and processed independently. Another advantage of our Neurosymbolic ARM approach over the commonly used algorithmic approaches, such as FP-Growth, is the option to leverage neural network–specific optimizations that significantly speed up execution and improve scalability. Some examples are batch normalization, both in training and when performing forward runs with the test

vectors, and quantization and pruning to reduce model size and inference time without compromising performance."

5. The study is theoretically rigorous, but a more explicit discussion of practical implications would enhance its relevance.

As recommended, we added a new discussion point to deliberate on the practical implications of our approach:
"Practical implications. Besides knowledge discovery, ARM is a cornerstone of interpretable machine learning models such as rule list classifiers, which are the standard approach to high-stakes decision-making (Rudin 2019). Such models process a given set of association rules to find a small subset that can be used to explain a certain class label. One example of such high-stakes decision-making in the scope of IoT systems, continuing our WDN example, is leakage detection in WDNs. In this case, our proposed pipeline can be used to learn association rules with a class label (e.g., 0: no leakage, 1: leakage) on the consequent side (ARM with item constraints as explained in the Variations of Aerial discussion point). The rules are then passed to a rule-based classifier such as CORELS (Angelino et al. 2018) to build a classifier that can detect leakages. This example can easily be extended to other anomaly detection tasks, including digital twins of IoT systems. In addition, our proposed pipeline and rule mining method Aerial is integrated into a digital twin architecture of a WDN (Degeler et al. 2024) to detect such abnormalities."

Reviewer 2 - Marvin Schiller

As far as I can judge, the presented methodology is well-motivated and executed and the results (in particular, the clear effect of the inclusion of semantics) appear highly relevant. I find the very detailed experimental analyses very instructive and easy to follow. The overall presentation, in terms of structure and language is excellent (nonwithstanding some minor comments below), and I appreciate the Figures and examples that illustrate the main workflow and the steps taken by the algorithm (like Figure 2 or Table 6 with examples of actual association rules). The source code of Aerial being available, well described and also at first glance quite clear is highly appreciated.

We would like to thank the reviewer for the kind words and for highlighting the strengths of our paper.

However, I noticed a gap in the otherwise convincing and clear presentation of the ideas, when putting myself in the shoes of someone who is trying to follow or replicate the presented work or adapt it to a potentially new domain (which I think would be the prime target group for this paper).

First of all, it was not clear to me how exact numeric (sensor) measurement values in the input data relate to the intervals "s1 must measure between 23-31" shown as part of the mined association rules or the comparisons like shown in Table 2 (e.g. "p1.length > 100"). In Section "Pipelines" I noted that "the proposed approach is applied to, such as the type of discretization...", so I assume that such details have been left out, but at least in the case of the discussed water sensor networks, it would have been instructional to see how this was done.

Based on the reviewer's feedback, we expanded the explanation of the data preparation steps, including discretization, under the Pipeline section. Furthermore, we used the water distribution network example given in the Problem Definition section throughout the methodology section (Semantic Association Rules from IoT data) to exemplify various steps of the pipeline.

Similarly, I understand that the semantics of classes in the ontology should not be understood in the sense of e.g. RDFS or OWL class hierarchies, but more like sets/tags. Still, I assume that the granularity of modeling applied to the ontology can have an effect on the performance of Aerial, so it would have been nice to see what was done here with a short example. Luckily I found some previous papers by the authors (e.g. https://arxiv.org/pdf/2310.07348) where this is more clear (showing a little part of the generated knowledge graph). Taking the practicioner perspective, I would assume that if the ontological modeling was done naively (e.g. one class per sensor instance) the value of the semantics would be null. Therefore, I would have found it highly valuable to see a coherent running example included, not only for the syntax of expressions (like Table 2), but also of the translation to the enriched sensor data. Such a running example could help to pinpoint how the positive effect of semantics unfolds during the course of processing, which would put the cherry on the cake. From my perspective, this would be more crucial than exhaustive formal definitions of the underlying set theoretic (graph, language) structures.

We agree with the reviewer's assumption that the granularity of modelling applied to the ontology can have an effect on the performance of not only our Aerial rule mining method but also any other rule mining method that utilizes our pipeline for semantic rule learning.

In line with the reviewer's previous comment, we used the water distribution network running example, initially given in the Problem Definition section throughout the paper. We expanded the example and included a figure of our knowledge graph, and described how the properties from the knowledge graph would be used to enrich sensor data in the data preparation step (semantic enrichment step), and continued to use this example throughout the other subsections of the Pipeline section. The source code for constructing the knowledge graphs can be found in our repository. We further described at the end of the second-to-the-last paragraph of the Data Preparation section that the granularity of semantic modeling in the knowledge graph can impact the rule quality and listed it among the future works in the Conclusion section.

Also, I was wondering to what degree typical knowledge-graph expressivity (like e.g. symmetry or transitivity of relations) would be picked up by the Aerial approach, to understand what degree of "semantification" is supported at all (again, thinking of making a dataset "Aerial-ready").

Indeed, we find the idea of capturing (and measuring) different levels of semantic expressivity using rules highly intriguing. We included it as a future research direction in the Conclusion and Future Work Section.

Also, to me it is not entirely clear if a transaction refers to all data in a given timespan of all sensors

together with the properties from the entire related knowledge graph (as it seems according to "Input", and also the sets presented as "Input transactions" for the autoencoder), or if transactions are structured or even grouped/cut according to individual sensors and their vicinity (as suggested by; "Property values from neighbors of node v can also be in the transaction set depending on the application."; from "Pipeline").

Transactions refer to all data in a timeframe of all sensors together with the properties from the related part of the knowledge graph, hence, one transaction per time frame. The latter sentence, “Property values from neighbors of node v can also be in the transaction set depending on the application”, refers to the fact that it depends on the application area to decide how much semantics to include in the transactions, i.e., up to first or second neighbors of where a sensor is placed in a knowledge graph.

We further elaborated and exemplified this in the Data Preparation Section, second and the third paragraphs.

Minor comments:

- The abbreviation DL (for deep learning) is never spelled out (to distinguish, e.g. from "description logic", since semantic technologies are also mentioned)
- In "Autoencoder Architecture": "lost function" --> "loss function"
- In the Discussion: "Semantic enrichment increases execution time by 2-3 times for Aerial and 3-12 times for exhaustive "methods, as shown in Experiment 2.1. --> Should read: "Experiment 1.2"
- In the conclusion: "lii" --> "ii"
- "Note that the Aerial" --> skip "the"

The comments are incorporated into the paper.

Reviewer 3 - Savitha Sam Abraham

Summary:
The paper addresses a very relevant research problem, that of mining a concise set of rules from dynamic sensor data leveraging static semantic knowledge about the domain. The proposed solution employs an autoencoder for rule mining, controls the number of rules mined by using additional parameters like semantic threshold and number of antecedents, and the static semantic knowledge is incorporated by enriching the input to the auto encoder.

We would like to thank the reviewer for recognizing the relevance and contributions of our paper.

What is not clear to me is the semantic enrichment step in the pipeline. I understand that eventually the input to the auto encoder is just a vector that does not have information about the semantic class or relations. Is the auto encoder aware of what these values in the vector actually represent semantically? If so, it has not been explained well in the paper.

The autoencoder is only aware of the vector representation of each feature category. Our rule extraction algorithm (Algorithm 1) keeps track of which category values are fed into which neuron of the autoencoder as a map of feature value pairs, and this information is used in rule extraction. For instance, line number 5 uses 'X.features' (X is the input transactions) to initialize an 'initial test vector' with equal probabilities per feature category, and line 6 marks feature categories in C (feature tuples), which are created in line 3 again using 'X.features', on the initial test vectors.

We agree with and appreciate the reviewer's comment that this connection between the vector representations and the tracking of their corresponding feature categories could be elaborated and emphasized more. We updated the **Pipeline** section and emphasized this connection under the first paragraph of **Training and Autoencoder Architecture subsection,** and in the Algorithm description paragraph of the **Rule Extraction from Autoencoders subsection**.

I would like to clarify: I understand that the output is structured to capture the semantics of the values being predicted - by using softmax that predicts probabilities per class values. Could you explain this further. For instance, as in the example provided: feature f1 can take values a, b and feature 2 (f2) takes three possible values. In this case the output would have two probability distributions, one for f1 across values an and b and the other for f2. Is this right? If so, do you discretize the input feature values?

The reviewer described the process correctly. The softmax function is applied to categories (classes) of each feature separately. As an example, assuming that there are only two features, f1 and f2, with two and three possible categories respectively, {a, b} and {c, d, e}, the output of our Autoencoder would be 2 probability distributions, one for each feature. Therefore, 'probability(a) + probability(b) = 1', and 'probability(c) + probability(d) + probability(e) = 1'. We then apply binary cross-entropy loss as the loss function to each probability distribution separately, aggregate the results, and propagate the loss back (during training). We restructured our **Pipeline** section and formally introduced this process in the **Training and Autoencoder Architecture** subsection.

Yes, we do discretize numerical values (please see the last part of the **Training and Execution subsection**, under the Setup section). Furthermore, in the new version of our paper, we dedicated a **Data Preparation subsection** under the **Pipeline section** to elaborate on all data preparation step including the discretization. Each subsection also includes a running example from the water distribution networks domain that we used in the Problem Definition earlier.

Why is the complexity only depending on number of features and number of antecedents? Shouldn't it depend on the number of values possible for each feature as well?

Yes, the reviewer is right in their assumption that the complexity also depends on the number of categories (possible values) per feature. In practice, the number of categories per feature (in comparison to the number of features) is often a small number, hence, we treated them as constants when considering the worst-case performance. As recommended by the reviewer, we re-formulated our complexity analysis to include the number of feature categories. We moved the analysis to the Appendix **Time Complexity Analysis of Aerial** to preserve the clarity and flow of the Methodology section. If the reviewer considers it more appropriate to include this analysis in the main text, we are open to reinstating it.

More explanation required: It would be nice to explain the choice of auto encoder - why undercomplete and not over complete. Did the ARM-AE paper also use under complete AE? Also, the paper mentions that ARM-AE considers the input as consequent and output as antecedent. Why does it matter? Is the proposed method differentiating between causality and correlation? The AE predicts the feature that is highly likely to co-occur with another feature. How do you conclude that it is f1-->f2 and not f2-->f1. Do you provide both of these instances as input to the AE and test?

We agree with the reviewer that the aforementioned questions are important to explain further in our paper, and we elaborated on them in the **Training and Autoencoder Architecture** and the **Rule Extraction from Autoencoders** sections. We further elaborated the differences of ARM-AE in **Experiment 6** in the Appendices. Following is a brief summary of our revisions:

Autoencoder Architecture. Under-complete autoencoders create a lower-dimensional representation of their input data and better capture the more significant aspects of the input data. Since the goal in association rule mining is to capture associations (co-occurrence) between feature categories, and also the goal of our paper is to capture a smaller number of rules to address the well-known rule explosion problem, we utilized an under-complete autoencoder and aimed to learn more prominent associations in the data, rather than all.

Rule Extraction. The field of association rule mining aims to find co-occurrence of feature categories, and does not use 'correlation' or 'causation' in its terminology. In line with the literature, our approach learns co-occurrences (associations) of feature categories. In the rule extraction stage, our approach creates test vectors with marked feature categories, while the rest of the categories for other features have equal probabilities per category. The input test vector represents a partially defined environment via the marked features, and the autoencoder reconstructs the co-occurrences with the rest of the environment. Therefore, we hypothesize that the marked feature classes in the test vector represent the antecedents of a rule, while the reconstructed feature classes represent the consequents. However, note that our algorithm (Algorithm 1) tests both probabilities, continuing the previous example, whether $f1(x) \rightarrow f2(y)$ and also $f2(y) \rightarrow f1(x)$, $x = \{a, b\}$, $y = \{c, d, e\}$.

The extensive evaluations showed that this approach indeed resulted in a smaller number of high-quality rules with full data coverage (Experimental Setting 2).

ARM-AE is different both in terms of autoencoder architecture and rule extraction methodology. Its autoencoder is not under-complete but has equal dimensions in each layer, has a different loss function (MSE), and the loss function is not applied per feature, but for the entire output. In the rule extraction stage, it does not assign equal probabilities to the unmarked features, but leaves them as 0, and assumes that the output is the antecedent while the input is the consequent. We argue that due to these differences, ARM-AE resulted in low rule quality with a higher number of rules, both in their paper (33% confidence on the Nursery dataset and 50% confidence on the chess dataset) and also based on our results (experiment 6 in the appendices).

Clarification: Is it right if I say that the proposed approach results in a concise set of rules only by controlling the input parameters - threshold and number of antecedents? If so, how do you ensure that high quality or significant rules are given priority to less significant rules?

The major reason why our approach results in a concise number of rules is the usage of an under-complete autoencoder that captures more prominent features in the data in its code layer rather than all features which may lead to obvious rules. On the other hand, it is true that the input parameters, both threshold and number of antecedents, affect the number of rules, similar to other state-of-the-art association rule mining methods. As shown in Experiment 3, selecting higher similarity thresholds tends to result in an even more concise set of rules with higher confidence and association strength (Zhang's metric). We reiterated this point in the Discussion section, in the **Neurosymbolic methods can help learning a concise set of high-quality rules** paragraph..

In Table 5, it is seen that FP-G method with semantics has better support and coverage for all datasets except LeakDB. It also has better confidence than Aerial with semantics in all but one dataset (LBNL). How do you explain this?

Table 5 alone is not enough to judge the ability of algorithms for mining high-quality rules.

This table is the result of an experiment that aimed to show that including semantics results in rules that are more generically applicable in comparison to having no semantics, which results in rules being applicable to specific sensors only. The table shows that both the support and average rule coverage increase upon including semantics for both approaches, hence demonstrating that our hypothesis holds.

The quality of the rules learned by our method and the baselines are compared in Experiment 2.2. Both Table 5 and Table 8 show that FP-Growth (Exhaustive) resulted in 10 to 100 times more rules than Aerial, leading to the well-known rule explosion problem. The data coverage of Aerial is 100% (1) despite its low number of rules (Table 8, note that the coverage in Table 8 refers to the data coverage of all rules, while the coverage in Table 5 refers to average rule coverage as stated in the title of each table). Furthermore, rules mined by FP-Growth have lower association strength (Zhang's metric), especially in L-Town and LBNL datasets, meaning less informative (obvious) rules, which tend to have higher confidence values. We argue that this also explains the slightly lower (but still compatible) confidence score of Aerial in comparison to FP-Growth. Support levels, on the other hand, are not by themselves informative as for instance high support levels can be good at explaining trends in the data, while low support levels might be better at detecting anomalies.

We revised the Experimental **Setting 1: Semantics vs without Semantics subsection** (first paragraph) to emphasize the fact that this setting does not compare the ARM algorithms but the impact utilizing semantics in rule mining.

Writing: I feel the form given in Table 2 is difficult to understand - may be a better representation can be used.

We shortened the item forms in Table 2 and in the Output section (first paragraph) to make them easier to understand.