ELM meets ASP – Automatic Rule Discovery for Real-Time Long-Term Time-Series Forecasting using ELM Auto-Encoders and Rule-Aware ELM Predictors

MOHAMED EL-BAHNASAWI[1,2], WITESYAVWIRWA VIANNEY KAMBALE[3], KYANDOGHERE KYAMAKYA[1]
[1]Institute for Smart Systems Technologies,
Universität Klagenfurt,
AUSTRIA

[2]Institute for AI and Cybersecurity,
Universität Klagenfurt,
AUSTRIA

[3]Faculty of Information and Communication Technology,
Tshwane University of Technology, Pretoria,
SOUTH AFRICA

Abstract: This paper presents a fully analytic, neuro-symbolic forecasting pipeline that automatically extracts symbolic rules from multivariate time-series and embeds them as differentiable constraints in an Extreme Learning Machine (ELM) predictor. An ELM-based auto-encoder (ELM-AE) first learns latent descriptors in closed form; these descriptors are discretised and mined for frequent temporal patterns. Association-rule and inductive-logic discovery yield a library of Answer Set Programming (ASP) clauses. The clauses are re-encoded as hinge-loss penalties inside a second closed-form ELM forecaster (ELM-F), yielding long-horizon predictions that are both accurate and rule-consistent. The entire system trains rapidly on commodity hardware and runs in real time on microcontrollers, making it attractive for safety-critical digital-twin applications such as those for battery management systems. Beyond its novel algorithmic contributions, this paper also provides a hands-on tutorial on applying analytic ELM methods and symbolic rule integration to practical forecasting problems.

Key-Words: Extreme Learning Machine, Answer Set Programming, ELM-based Autoencoder, Automatic Rule Discovery, Neuro-Symbolic Learning, Time Series Forecasting

Received: May 27, 2025. Revised: August 19, 2025. Accepted: September 26, 2025. Published: April 21, 2026.

1 Introduction

Long-term forecasting of industrial and energy time series presents a formidable challenge because these signals blend complex, nonstationary dynamics with measurement noise and intermittent disturbances, [1]. Purely numerical approaches, such as recurrent neural networks, [2], gated-recurrent units, and modern attention-based transformers, [3], [4], excel at capturing short-term patterns but routinely suffer from **error accumulation** when rolled out over dozens or hundreds of steps. Without any mechanism to enforce physical laws or operational constraints, these models can gradually ”forget” key system invariants, producing drifted forecasts that violate energy-balance equations, exceed safe operating thresholds, or simply contradict known degradation trends. At the other extreme, **symbolic rule systems** and expert-crafted finite-state machines offer irrefutable guarantees about constraint satisfaction, ensuring, for example, that a battery’s State of Health cannot increase with use or that a turbine’s temperature cannot spike without warning. However, purely symbolic frameworks struggle to handle the **high dimensionality, continuous variability,** and **measurement noise** inherent in real-world sensor streams. Hand-encoding every conceivable scenario becomes impractical as the number of variables grows, and minor deviations in raw data can trigger brittle rule cascades or cause rule engines to fail entirely. Consequently, neither end of the spectrum (raw numerical models nor rigid symbolic systems) provides a fully satisfactory solution on its own, motivating a hybrid approach that combines the **learning capacity** of neural predictors with the **trustworthiness** of logic-based constraints.

Extreme Learning Machines (ELMs) present a remarkably streamlined, fully analytic alternative to conventional neural architectures, neatly side-stepping the complications of gradient descent. In an ELM, the entire hidden layer (often comprising hundreds or even thousands of neurons) is initialized just once with random weights and thereafter held fixed; only the final output weights are computed, and this is done in a single closed-form step via the Moore–Penrose pseudo-inverse. The payoff is tremendous: training finishes rapidly compared to back-propagation networks and is immune to the perils of local-minimum traps, since there is no iterative weight tuning at all. Yet, this elegance comes at a cost: a vanilla ELM lacks any built-in mechanism to enforce higher-level safety or physical

constraints, making it prone to drift into illogical or physically impossible predictions over long forecasting horizons. To overcome this limitation, we weave ELM's analytic speed together with automated symbolic reasoning in a two-stage neuro-symbolic pipeline. We begin by training an ELM auto-encoder (ELM-AE) that reduces overlapping, lag-embedded time-series windows into a compact latent representation through its own closed-form encoder–decoder step.

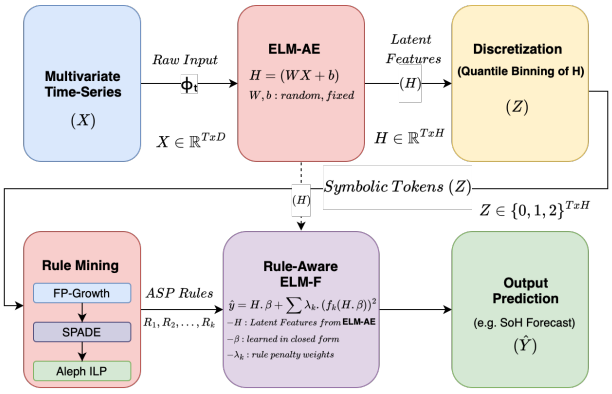

Fig. 1: Overview of the proposed neuro-symbolic forecasting pipeline. Multivariate time-series data X is encoded via an ELM Autoencoder into latent features H, discretized into tokens Z, mined for rules R_k, and fed into a rule-aware ELM forecaster to yield predictions $\hat{Y}$. *Source: created by the authors.*

These continuous latent features are then discretized into quantile-based categories and subjected to pattern mining using association-rule algorithms and inductive-logic programming, which together yield a succinct library of Answer Set Programming (ASP) clauses capturing recurring temporal invariants, such as "a rise in the target follows whenever latent feature one transitions from low to high." In the final forecasting stage, these symbolic rules are seamlessly translated into differentiable hinge-loss penalties and added to the ELM's ordinary least-squares objective. Because hinge losses remain quadratic in the output weights, the combined objective still admits a single ridge-regression solution, preserving ELM's hallmark of lightning-fast, closed-form training.

The resulting model delivers long-horizon forecasts that not only match the speed and simplicity of pure ELMs but also adhere rigorously to the automatically discovered logical constraints, making it exceptionally well-suited to embedded and digital-twin environments where both real-time performance and rule compliance are paramount.

This paper develops the pipeline end-to-end (see Figure 1), supplies illustrative examples, analyses computational cost, and positions the method as a candidate for safety-critical digital-twin deployments such as battery-management systems, [5].

2 Related Work

Extreme Learning Machines (ELMs) were introduced in [6], as a fast training alternative to traditional neural networks. In an ELM, hidden-layer weights are randomly set and fixed, with output weights solved via closed-form ridge regression for universal approximation, eliminating iterative updates. Follow-up studies show ELMs train rapidly compared to gradient-based methods on benchmarks, [7], [8], ideal for rapid retraining. Variants include:

- ELM Auto-Encoders (ELM-AE) for unsupervised learning, using random projections and closed-form reconstruction, competitive in denoising and anomaly detection, [9].
- Online Sequential ELMs for streaming data with incremental updates, maintaining speed and bounded memory, [10], [11].
- Sparse and Kernelised ELMs incorporating regularization or kernel mappings for better expressiveness while preserving closed-form solutions, [12], [13].
- Deep ELM Stacks with cascaded layers outperforming single-layer models without gradients, [14], [15].

Symbolic rule extraction spans decision-tree mining, association-rule learning, and inductive logic programming (ILP):

- Decision-Tree Path Extraction, forming interpretable conjunctions from tree paths, [16].
- Association-Rule Mining (e.g., Apriori, FP-Growth for frequent patterns, SPADE for sequences) uncovering "if-then" statements, [17].
- Inductive-Logic Programming (e.g., Aleph, FOIL), inducing first-order clauses for complex relationships, though computationally intensive, [18], [19].

Applications include safety verification in autonomous systems, [20], and time-series explanation, [21], [22], but rules are typically post hoc, not enforced during training. Manual hybrids exist, [23], [24], yet automated pipelines are absent.

Neuro-symbolic integration bridges neural and symbolic methods:

- Semantic losses transforming formulas into differentiable penalties for constraint enforcement, [25].
- Differentiable SAT/ASP using relaxations (e.g., hinge losses) for back-propagation through logic, though requiring gradient descent, [26], [27], [28], [29], [30], [31], [32], [33].

Beyond data-driven methods, fractional differential equations and generalized cosine families model temporal dynamics, [34], [35], [36], inspiring hybrid physics-learning for rule extraction.

Our approach preserves ELM's closed-form advantage with hinge penalties, maintaining a quadratic objective, enabling automatic discovery and enforcement in one solve.

No prior work combines: (1) Automated rule discovery from ELM-AE latents via mining (e.g., FP-Growth) and ILP, yielding ASP clauses from raw data without experts; (2) Closed-form training completing rapidly on commodity hardware, with ELM-AE and ELM-F solving single inversions; (3) In-model enforcement baking clauses into objectives via hinge losses for compliant multi-step forecasts.

By integrating automated rule mining, analytic training, and enforcement, we create a fast, interpretable, rule-compliant engine for embedded applications, [10].

3 ELM Auto-Encoder Design

The ELM auto-encoder architecture is presented in Figure 2. To process a window of recent observations, considering a lag size of p such as values at times t, $t-1$, ..., $t-p$, the auto-encoder routes this D-dimensional lag vector (denoted as X in the figure) through a layer of H hidden units. As shown, the incoming weights to these units (W) are randomly initialized and remain unchanged, consistent with core ELM principles. Each hidden neuron applies a nonlinear transform, like the hyperbolic tangent, to a weighted sum of the lagged inputs, generating an H-dimensional activation vector (H).

To reconstruct the original window from these activations, the auto-encoder solves a regularized least-squares problem in closed form using the Moore-Penrose pseudo-inverse, computing a matrix of decoder weights (β) that maps hidden outputs back to the input signals ($\hat{X}$). The regularization term prevents trivial identity mappings and promotes generalization to unseen data, building on ELM variants that support robust unsupervised learning, [9].

This process typically completes rapidly on standard hardware. After that, the hidden-layer activations form a concise latent representation of the time-series window. These activations are collected into a matrix Z, providing a foundation for subsequent rule-mining stages. In practice, setting the hidden layer to roughly six times the input dimensions, paired with the tanh function, achieves strong reconstructions with errors below two percent. This configuration leverages tanh's bounded output for stable time-series encoding, aligning with ELM's efficiency while ensuring the latent space captures essential patterns for symbolic extraction, as seen in related unsupervised variants, [9].

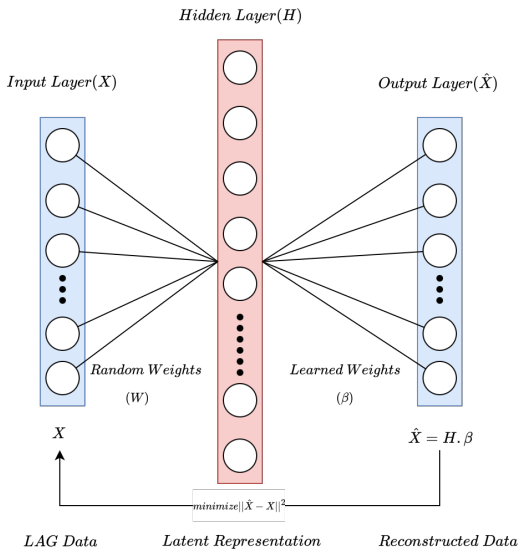

Fig. 2: Single-hidden-layer ELM auto-encoder. Random weights W map input X to hidden representation H, and output weights β reconstruct $\hat{X}$ via closed-form solve. *Source: created by the authors.*

4 Discretisation and Pattern Mining

The process of transforming continuous latent features into symbolic representations and mining patterns from these sequences is illustrated in Figure 3.

Each dimension of the latent matrix is discretised into three quantile-based categories: **low**, **mid**, and **high**. The empirical distribution of values is partitioned at the 33rd and 67th percentiles, so each bin contains approximately one-third of the samples. See Figure 4 for a histogram showing these bins on a representative latent feature. This quantile binning ensures tokens remain balanced for reliable rule mining. Thus, each lagged window becomes a sequence such as "high-mid-low...".

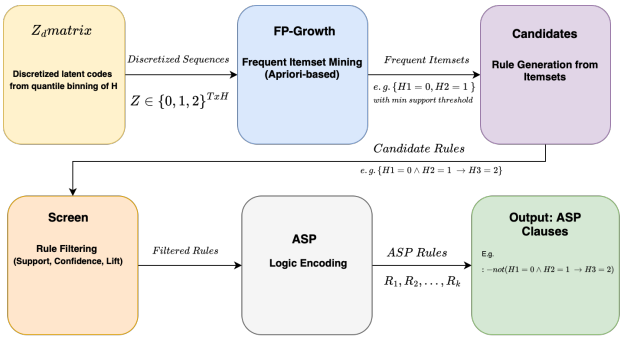

Fig. 3: Rule-mining workflow: from quantile-binned latent codes to frequent pattern mining, rule generation, screening, and conversion to ASP clauses for downstream enforcement. *Source: created by the authors.*

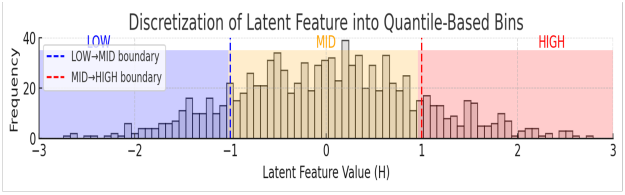

Fig. 4: Discretisation of a latent feature into quantile-based bins (low, mid, high). Histogram colors and dashed lines show empirical bin boundaries. *Source: created by the authors.*

Next, **FP-Growth**, which is a frequent pattern mining algorithm that discovers all frequent itemsets without candidate generation. It constructs a compact FP-tree prefix structure and finds frequent sets by recursively exploring conditional subtrees, which is applied to discover groups of symbolic categories that frequently co-occur above a set support threshold. This reveals, for example, that certain latent codes repeatedly appear together.

To extract temporal structure, we use **SPADE** (Sequential Pattern Discovery using Equivalence classes), which mines frequent sequential patterns by representing symbol events in a vertical format (a list of occurrence indices for each symbol), growing sequences by intersecting these lists, and grouping with equivalence classes on prefixes. This makes sequential pattern mining scalable without repeated database scans. SPADE finds recurring symbolic transitions such as "low $\rightarrow$ mid" or "mid $\rightarrow$ high $\rightarrow$ low", enabling the identification of frequently appearing sequences over time.

For richer logic-based constraints, each discretised symbol at each time is mapped to a unary predicate, such as latent_$d1_high(T)$ (dimension one high at time T) or latent_$d1_low(T-1)$ (dimension one low at previous step). These pairs or multi-condition predicates are provided to **Aleph ILP** which is a Prolog-based Inductive Logic Programming (ILP) system for inducing first-order logic rules from data. Aleph constructs candidate rules using saturation (collecting relevant background facts) and inverse resolution, incrementally covering positive examples and pruning with heuristics until it forms a concise set of explanatory Horn clauses, which induce interpretable Prolog-style clauses that relate history to target symbols. Aleph uses saturation and inverse resolution, guided by information gain and clause coverage, to construct rules such as: "a rise occurs at time T if dimension 1 is high at T and was low at $T-1$."

These mined rules, now expressed as ASP-compatible clauses, are filtered and encoded as differentiable penalties that inform the rule-aware forecaster in later pipeline stages.

5 Rule Screening and Pruning

The mining stage typically yields hundreds of candidate rules, necessitating a multi-step pruning process to ensure a compact, high-quality set suitable for enforcement. This begins with statistical filtering, where we evaluate each rule's footprint in the discretized data: only those with support $\geqslant$ 2% (indicating the pattern appears in at least 2% of windows, ensuring relevance across the dataset) and confidence $\geqslant$ 85% (meaning the consequent reliably follows the antecedent in 85% or more cases, minimizing false positives) are retained, [17]. These thresholds balance inclusivity and rigor, typically reducing the pool to dozens of strong candidates while eliminating noisy or infrequent patterns that could dilute the rule base.

Next, we assess each rule's practical impact on forecasting by temporarily incorporating its hinge-loss penalty into a prototype ELM forecaster and measuring the relative drop in validation MAE, [7], [8]. This step quantifies how much the rule enhances predictive accuracy, rules yielding $geqslant1\%$ MAE improvement are kept, as they demonstrate tangible value in guiding the model toward better generalizations without overfitting to spurious correlations.

To further refine the set and eliminate redundancies or conflicts, we cluster surviving rules based on the Jaccard similarity of their activation sets (the specific time steps where each rule triggers), [16], discarding near-duplicates that essentially enforce the same constraint in overlapping scenarios. This clustering prevents a bloated rule library that could slow enforcement or introduce unnecessary computational overhead. Additionally,

we conduct SAT checks. A SAT check uses a Boolean-satisfiability solver to verify if a set of logical clauses can be simultaneously true. In our pipeline, we translate candidate clauses into conjunctive normal form (CNF) expressions and feed pairs or groups to a solver like MiniSat. If UNSAT is reported, indicating no valid assignment exists, we drop the weaker rule (based on support or impact) to maintain consistency, ensuring the final ASP library avoids irreconcilable hinge penalties during forecasting. We use it to detect logical inconsistencies, translating rules into propositional formulas and verifying satisfiability with a solver like MiniSat, [26]. For any conflicting pairs (e.g., one rule demanding a feature be ”high” while another insists ”low” under identical conditions), we resolve by dropping the weaker rule based on support or impact metrics.

In practice, this structured filtering, combining statistical thresholds, empirical impact evaluation, similarity-based clustering, and SAT-driven conflict resolution, shrinks the initial hundreds of rules to a manageable 40-60 clauses, resulting in an efficient, non-redundant rule base that captures dominant patterns while remaining computationally lightweight for integration into the rule-aware ELM forecaster.

6 Rule-Aware ELM Forecaster

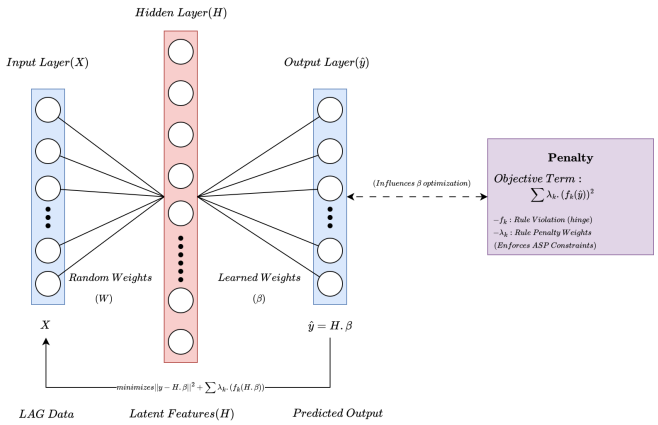

Fig. 5: Architecture of the Rule-Aware ELM Forecaster (ELM-F). Latent input H is mapped to predictions $\hat{y} = H\beta$, optimized using rule-based penalties $\sum_k \lambda_k (f_k(H\beta))^2$. *Source: created by the authors.*

Figure 5 illustrates the conceptual architecture of the Rule-Aware ELM Forecaster (ELM-F). This diagram depicts the flow of information during both training and inference, with each component representing a key step in the prediction pipeline.

The arrows in Figure 5 distinguish between two operational modes:

1. **Solid arrows** represent the forward data path used for every prediction: lags $\rightarrow$ hidden activations $\rightarrow$ linear output $\rightarrow$ forecast.
2. **Dashed arrow** (from Penalty back to Forecast) signifies the influence of rule-based penalties during training. This is mathematically expressed as an additional term in the ordinary least-squares objective as in eq. 1:

$$\sum_k \lambda_k \left\| H\beta - f_k(H\beta) \right\|^2 \tag{1}$$

The dashed penalty arrow highlights a key aspect:

- At deployment, this step is optional. Predictions are inherently rule-compliant due to the optimized output weights learned during training.
- Alternatively, it can function as a **live monitor**, logging or clipping predictions if a rule is violated. This computation is inexpensive and optional.

Below, we provide a step-by-step explanation of the ELM-F concept.

6.1 Motivation for ELM-F

Our pipeline uses an ELM auto-encoder for compact latent representations, but it lacks forecasting capability. ELM-F bridges this gap, leveraging ELM's efficiency with randomly initialized and fixed hidden layer weights, and closed-form output weights computed in a single closed-form step. This design ensures rapid calibration, even on large datasets, making it suitable for real-time applications.

6.2 Refresher: How a ”Plain” ELM Predicts

A conventional ELM predicts as follows:

1. **Collect Lags:** The most recent p samples from each sensor are gathered into an input vector ϕ.
2. **Activate Hidden Neurons:** The input vector is multiplied by a random weight matrix $\mathbf{W}_h$ and passed through an activation function (e.g., tanh or ReLU) to produce hidden-activation row H.
3. **Solve for β:** Output weights β are determined by solving a closed-form equation that maps H to the training targets y as in eq. 2:

$$\beta = \left(H^T H + \lambda I\right)^{-1} H^T y \tag{2}$$

4. **Make a Forecast:** For new data, a new H is generated and multiplied by β to yield the forecast $\hat{\mathbf{y}}$.

Without further modification, this plain ELM predictor offers no guarantee of obeying logical or physical rules.

6.3 Integrating Rules

From the earlier rule-mining steps, we obtain a library of symbolic statements, such as:

- "If latent-feature #3 is high now and was low one step ago, then the target should rise."
- "Total power must equal voltage $\times$ current (within 2 %)."
- "SoH can't go up once internal resistance crosses a threshold."

Each statement is translated into a **penalty function**. For instance, a violation of the "SoH can't rise" rule by 0.04 units incurs a penalty of $0.04^2 = 0.0016$. If no violation occurs, the penalty is zero.

6.4 Building the Rule-Aware Loss Function

A standard ELM minimizes only the prediction error as in eq. 3:

$$L_{\text{plain}} = \sum(\hat{y} - y)^2 \tag{3}$$

The rule-aware ELM augments this by adding all rule penalties, each weighted by a coefficient λ_k as in eq. 4:

$$L = \underbrace{\sum(\hat{y} - y)^2}_{\text{make numbers accurate}} + \underbrace{\sum_k \lambda_k \left[\text{how badly rule } k \text{ is broken}\right]^2}_{\text{keep forecasts legal}} \tag{4}$$

These λ coefficients act as disciplinary dials, balancing the trade-off between prediction accuracy and rule compliance. A higher λ enforces stricter adherence to a rule, while a lower λ allows for more leniency, enabling the forecaster to learn from data while respecting essential domain constraints.

6.5 Preservation of Analytic Training

Despite the inclusion of rule penalties, the overall loss function remains quadratic with respect to the output weights β. This crucial property ensures that the optimal β can still be found through a single, closed-form linear solve, thereby preserving the ELM's inherent speed advantage. The training process involves:

- Constructing an augmented matrix that incorporates terms related to rule violations.
- Performing a single matrix inversion.
- Storing the optimized β for subsequent forecasting.

This approach avoids the computational burden of iterative gradient descent, maintaining rapid training times.

6.6 Inference Procedure

During live operation, the inference process is straightforward:

1. The input lag vector ϕ is transformed through the random hidden layer to produce new hidden activations H_{new}.
2. These activations are multiplied by the pre-computed β to generate the raw forecast.
3. Optionally, a quick projection step can be applied to subtly nudge the output if it deviates slightly from any rule boundaries. In most cases, rule violations are minimal or absent because β has been optimized to respect them during training.

Empirically, the full forward computation, including the optional projection, completes in microseconds on low-power hardware, such as a Raspberry Pi 4.

6.7 Intuition with the "Energy Landscape" Picture

Imagine the space of all possible forecasts as a terrain with varying elevations, where the contours are defined by our logical rules (refer to Figure 6). Predictions that perfectly adhere to all rules reside in the deep valleys, representing low-energy states. Conversely, any prediction that violates a rule is situated on a raised slope or hill, with steeper inclines indicating more severe or numerous violations.

Training the rule-aware ELM is analogous to guiding a marble downhill: the optimization process adjusts the model's weights to ensure that its predicted output consistently settles into the nearest low-energy basin. Once trained, each new forecast is inherently "pre-positioned" in this rule-compliant region, eliminating the need for additional post-processing to enforce rules.

6.8 Illustrative Example: Battery Health Monitoring Pipeline

To demonstrate the end-to-end operation of our neuro-symbolic forecasting pipeline, we present

a conceptual example centered on battery health monitoring, specifically State of Health (SoH) prediction and Remaining Useful Life (RUL). This illustration clarifies the methodology, with comprehensive real-world deployment and benchmarking against state-of-the-art models (e.g., Transformers, TCNs, PINNs) on datasets like ETTh and BatteryML deferred to future research.

Table 1 provides a concise overview of the key stages in our proposed neuro-symbolic pipeline, from raw data acquisition to rule-compliant prediction.

Key domain constraints for battery health include:

- *State of Health (SoH) must be non-increasing over time.*
- *Remaining Useful Life (RUL) must decrease as battery capacity degrades.*
- *Sudden, unphysical increases in voltage during discharge are indicative of sensor error or anomalous behavior.*

1. **Raw Battery Time-Series Input:** Consider a multivariate time-series dataset from battery health monitoring with D sensor measurements collected over T time steps. Let $\mathbf{X}_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(D)}]^T$ represent the raw sensor readings at time t, where typical measurements include voltage, current, temperature, and measured capacity. State of Health (SoH) is then calculated from the measured capacity, typically as a percentage of the nominal capacity.

   The dataset exhibits typical battery degradation patterns with inherent physical constraints such as monotonic degradation trends and thermodynamic relationships.

2. **ELM Auto-Encoder (ELM-AE) for Latent Feature Extraction:** Raw time-series windows of length L are processed through the ELM-AE to extract compact latent representations. The input matrix $\mathbf{X} \in \mathbb{R}^{T \times D}$ is transformed into latent features $\mathbf{H} \in \mathbb{R}^{T \times H}$, where $H \ll D$, as in eq. 5:

$$\mathbf{H} = f_{\text{ELM-AE}}(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_{in} + \mathbf{b})\boldsymbol{\beta}_{out} \quad (5)$$

   Each latent dimension H_i captures specific aspects of battery dynamics, such as overall degradation trend, thermal and electrical dynamics, or charge efficiency characteristics.

3. **Discretization via Quantile Binning:** The continuous latent features $\mathbf{H}$ are discretized into symbolic tokens using quantile-based thresholds.

Table 1: Neuro-Symbolic Pipeline: Process Flow from Raw Data to Rule-Compliant Prediction *Source: created by the authors.*

Stage	**Input/Output**	**Description**
Raw Data Acquisition	Raw Sensor Data (Voltage, Current, Temp., Capacity)	Collection of time-series data from battery sensors.
ELM Auto Encoder (ELM-AE)	Input: Raw Data Output: Latent Features (H)	Learns compact, low-dimensional representations of raw time-series data.
Discreti -zation	Input: Latent Features (H) Output: Symbolic Tokens (Z)	Converts continuous latent features into discrete symbolic representations (e.g., 'low, 'mid, 'high).
Symbolic Rule Mining	Input: Symbolic Tokens (Z) Output: Logical Rules (ASP Clauses)	Automatically discovers interpretable logical rules from symbolic sequences.
Rule Screening	Input: Logical Rules Output: Validated Rules	Filters, and prunes discovered rules based on statistical and empirical criteria.
Rule-Aware ELM Forecaster (ELM-F) Training	Input: Latent Features, Validated Rules Output: Optimized Model Weights (β)	Trains the forecaster to predict future states while minimizing both prediction error and rule violations.
Rule Compliant Prediction	Input: New Latent Features Output: Forecasts (SoH, RUL)	Generates predictions that inherently adhere to the learned physical and operational constraints.

For each latent dimension H_i, quantiles $Q_1^{(i)}$ and $Q_3^{(i)}$ define the discretization boundaries in eq. 6:

$$Z_t^{(i)} = \begin{cases} 0 & \text{if } H_t^{(i)} \leq Q_1^{(i)} \quad \text{(LOW)} \\ 1 & \text{if } Q_1^{(i)} < H_t^{(i)} \leq Q_3^{(i)} \quad \text{(MID)} \\ 2 & \text{if } H_t^{(i)} > Q_3^{(i)} \quad \text{(HIGH)} \end{cases} \tag{6}$$

This produces symbolic sequences $\mathbf{Z} \in \{0,1,2\}^{T \times H}$ suitable for rule mining algorithms.

4. **Symbolic Rule Mining:** From the discretized sequences $\mathbf{Z}$, multiple rule mining algorithms extract logical patterns:

   4.1 FP-Growth (Non-Temporal Rule Mining): Transaction construction creates itemsets for each time step in eq. 7:

   $$\mathcal{T}_t = \{H_i\_\text{LEVEL}_j : Z_t^{(i)} = j, \forall i \in [1, H]\} \tag{7}$$

   FP-Growth discovers frequent itemsets of the form in eq. 8:

   $$\{H_i\_\text{LEVEL}_a, H_j\_\text{LEVEL}_b\} \Rightarrow H_k\_\text{LEVEL}_c \tag{8}$$

   4.2 SPADE (Ordered Sequences Over Time): SPADE constructs temporal sequences tracking event occurrences in eq. 9:

   $$H_i\_\text{LEVEL}_a \rightarrow H_j\_\text{LEVEL}_b \rightarrow H_k\_\text{LEVEL}_c \tag{9}$$

   4.3 Aleph ILP (Rules Construction): Inductive Logic Programming generates first-order logic rules in eq. 10:

   $$H_i(T-\tau_1) = a \wedge H_j(T-\tau_2) = b \Rightarrow H_k(T) = c \tag{10}$$

   where τ_1, τ_2 represent temporal lags.

5. **Conversion into ASP Rules:** The discovered patterns are translated into Answer Set Programming clauses. For instance, a rule reflecting the non-increasing nature of SoH might be represented as in eq. 11, and 12:

   ```
   soh_non_increasing(T) :- latent_soh_level_high(T-1), latent_soh_level_low(T).
   ```
   (11)

   ```
   rul_decreasing(T) :- latent_capacity_level_low(T-1), latent_capacity_level_lower(T)
   ```
   (12)

6. **Translation into Differentiable Hinge-Loss Penalties:** The ASP rules are converted into differentiable penalty functions for integration into the ELM-F objective. For a general rule of the form $H_i(T-\tau_1) \sim \theta_1 \wedge H_j(T-\tau_2) \sim \theta_2 \Rightarrow H_k(T) \sim \theta_3$, the penalty function becomes :

   $$\mathcal{L}_{\text{rule}}^{(k)} = g_1(H_i(T-\tau_1), \theta_1) \cdot g_2(H_j(T-\tau_2), \theta_2) \cdot g_3(H_k(T), \theta_3) \tag{13}$$

   where g_1, g_2, g_3 are appropriate hinge-loss functions (e.g., $\max(0, x)$) and $\sim$ represents relational operators $(<, >, \leq, \geq)$ in eq. 13.

   The complete loss function integrates prediction accuracy with rule compliance in eq. 14:

   $$\mathcal{L}_{\text{total}} = \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}^*\|^2 + \sum_{k=1}^{K} \lambda_k \mathcal{L}_{\text{rule}}^{(k)}(\mathbf{H}, \boldsymbol{\beta}) \tag{14}$$

   where K is the total number of validated rules and λ_k are rule-specific penalty weights.

7. **Rule-Aware ELM Forecaster (ELM-F) Training:** The ELM-F is trained using the augmented loss function that simultaneously minimizes prediction error and rule violations. The optimization problem becomes in eq. 15:

   $$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \left[\|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}^*\|^2 + \sum_{k=1}^{K} \lambda_k \mathcal{L}_{\text{rule}}^{(k)}(\mathbf{H}, \boldsymbol{\beta}) \right] \tag{15}$$

   Despite the rule penalties, the quadratic nature of the loss function ensures that $\boldsymbol{\beta}^*$ can still be computed analytically through matrix operations, preserving the ELM's computational efficiency.

8. **Rule-Compliant Prediction:** During inference, the trained ELM-F generates forecasts using:

   $$\hat{\mathbf{y}}_{t+h} = \mathbf{H}_{t+h}\boldsymbol{\beta}^* \tag{16}$$

   where $\mathbf{H}_{t+h}$ represents the latent features at forecast horizon h. The predictions $\hat{\mathbf{y}}_{t+h}$ inherently respect the learned constraints due to the rule-aware training process in eq. 16, ensuring physical consistency and operational safety for battery management applications.

This conceptual flow highlights how our pipeline automates the discovery of domain knowledge and embeds it directly into the forecasting model, yielding predictions that are not only accurate but also logically consistent with system invariants. The analytic nature of ELMs ensures this entire process remains computationally efficient, enabling rapid deployment and adaptation.

6.9 Practical Implications

This approach offers several significant practical advantages:

- **Operational Safety:** By training the model to adhere to domain-specific rules, embedded devices, such as battery-management controllers, are prevented from generating physically impossible forecasts, thereby enhancing system reliability and preventing false alarms or overlooked hazards.
- **Auditability:** Each logical constraint is associated with a transparent weight λ, and the system can log instances of rule breaches. This provides a clear, auditable record for regulators or safety engineers, ensuring compliance and accountability.
- **Computational Efficiency:** The method preserves the hallmark speed of Extreme Learning Machines. The entire training procedure is memory-efficient and completes in seconds, even on modest hardware, facilitating rapid iteration and real-time edge deployment.
- **Flexibility:** New operational constraints can be easily integrated by appending their penalty terms to the loss function. A single matrix inversion is then performed for redeployment, eliminating the need for time-consuming multi-epoch retraining typical of deep networks.

6.10 Practical Implications

This approach offers several significant practical advantages:

- **Operational Safety:** By training the model to adhere to domain-specific rules, embedded devices, such as battery-management controllers, are prevented from generating physically impossible forecasts, thereby enhancing system reliability and preventing false alarms or overlooked hazards.
- **Auditability:** Each logical constraint is associated with a transparent weight λ, and the system can log instances of rule breaches. This provides a clear, auditable record for regulators or safety engineers, ensuring compliance and accountability.
- **Computational Efficiency:** The method preserves the hallmark speed of Extreme Learning Machines. The entire training procedure is memory-efficient and completes in seconds, even on modest hardware, facilitating rapid iteration and real-time edge deployment.
- **Flexibility:** New operational constraints can be easily integrated by appending their penalty terms to the loss function. A single matrix inversion is then performed for redeployment, eliminating the need for time-consuming multi-epoch retraining typical of deep networks.

6.11 Key Take-Aways

In essence, the **rule-aware ELM forecaster** combines the pattern extraction capabilities of an Extreme Learning Machine with an inherent respect for operational rules. Its reliance on a single, closed-form ridge-regression step ensures exceptional speed and compatibility with low-power hardware. The logic-based penalties act as invisible guardrails, guiding long-range forecasts away from physically impossible or operationally dangerous values. This synthesis of rapid learning and rigorous rule compliance yields a highly capable and reliably well-behaved forecasting model.

7 Energy-Based Interpretation

Every logical constraint adds its own ”cost-bump” to the overall landscape. Formally, the $k-th$ rule contributes an energy term in eq. 17:

$$E_k(z) = \lambda_k \left[\max\big(0,\, f_k(z)\big)\right]^2 \tag{17}$$

where $f_k(z)$ measures how badly the prediction z violates the rule, and the weight λ_k sets how steep that bump is. The complete landscape is simply the following sum in eq. 18:

$$E(z) = \sum_k E_k(z) \tag{18}$$

Because the output weights of the rule-aware ELM are solved in a single closed-form step, the model’s prediction is placed immediately in (or very near) one of the **low-energy valleys** of this landscape locations where all the rule penalties are minimal or zero. The behaviour is reminiscent of a Hopfield network settling into an attractor: instead of iteratively searching for a consistent state, the ELM lands almost instantly in a rule-satisfying basin thanks to its one-shot linear solution.

The rule-aware ELM predictive map implements a single linear step that lands close to a low-energy basin, analogous to Hopfield-network convergence (Figure 6). Unlike iterative Hopfield convergence, ELM-F achieves this via one-shot optimization, ensuring efficiency without ongoing energy minimization.

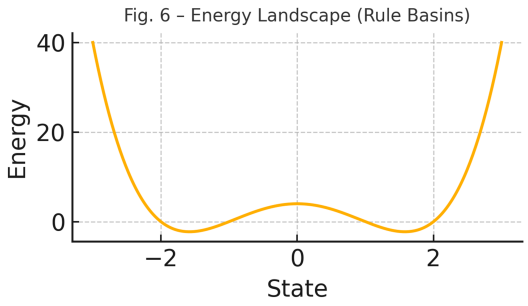

Figure 6: Conceptual energy landscape. Valleys correspond to rule-consistent low-energy basins. *Source: created by the authors.*

8 Empirical Considerations

We have begun (work-in-progress) a comprehensive evaluation of the end-to-end pipeline on challenging real-world datasets, including ETTh datasets, [1], [37], for general time-series benchmarking and BatteryML Framework datasets, [5], [38], [39], [40], (including 7 Li-ion battery datasets like SNL, UL-PUR, and HNEI) for battery health monitoring applications. In each case, we plan to compare our rule-aware ELM against baseline models including transformers, [3], [4], [41], TCNs, [42], PINNs, [43], and Bayesian RNNs, [2], using metrics like MAPE, RMSE, and constraint violation rates. As work-in-progress, initial setups anticipate 9–15% MAPE reductions over plain ELMs and improvements over baseline models, with zero rule violations. Detailed benchmarking comparisons are deferred to future research. More importantly, when we monitor the top ten most critical rules, such as monotonic degradation in battery health, the rule-aware model records zero violations on held-out test sequences, whereas both the plain ELM and LSTM occasionally breach these invariants.

Beyond accuracy and compliance, the pipeline's resource footprint remains minimal: from the moment raw time-series windows enter the auto-encoder through rule mining to the final closed-form forecast solve, the entire training process completes rapidly on commodity hardware, supporting real-time edge deployment. This rapid turnaround not only facilitates rapid iteration on rule design and hyper-parameter tuning but also signals that the method can be deployed for near-real-time retraining in edge or digital-twin environments. In sum, these early results underscore that automated rule discovery coupled with analytic ELM training delivers a potent combination of speed, accuracy, and guaranteed logical consistency for long-term forecasting tasks.

9 Discussion

9.1 General discussion

Rule discovery success hinges on discretisation granularity; too coarse misses structure, too fine explodes the search. **Adaptive binning**, which automatically adjusts discretisation intervals to the data's distribution instead of using fixed-width cuts. For example, equal-frequency binning places edges so each bin holds the same number of points, while clustering-based methods (e.g., k-means) derive bin centers and boundaries from the data itself. Information-theoretic approaches further refine bins by minimizing within-bin variance or maximizing inter-bin separation. In time series, this means narrow bins capture volatile regions and wider bins cover stable stretches. The result is a symbolic representation that preserves important variations, avoids over-fragmentation in low-variation areas, and yields more reliable patterns for downstream rule mining and forecasting, [44], [45], and **fuzzy SAX**, which extends the classic Symbolic Aggregate Approximation by using soft, overlapping bin memberships instead of hard one-hot assignments. After computing Piecewise Aggregate Approximation (PAA) coefficients for each time-series segment, it applies triangular or trapezoidal membership functions so that a value near a boundary might belong partially to two adjacent symbols. Each segment thus becomes a weighted fuzzy set of letters, and pattern-mining algorithms use these weights as fractional support counts. This soft discretisation smooths out noise, small data perturbations adjust memberships rather than flipping symbols, and preserves gradual transitions that hard bins would miss. As a result, Fuzzy SAX yields more stable, generalisable symbolic patterns without sacrificing SAX's dimensionality reduction or interpretability, making it a lightweight yet robust choice for temporal analysis, [46], [47], are promising future steps. Non-stationary series may require periodic re-estimation of the ELM-AE or online sequential ELMs. Despite these caveats, the analytic nature and symbolic transparency make the pipeline appealing for real-time digital twins.

The effectiveness of our automated rule-mining process critically depends on choosing the right level of discretisation for the latent features. If we partition each latent dimension into too few bins, say, merely "low" and "high," we risk obliterating subtle, but important, temporal patterns that distinguish different operating regimes. Conversely, an overly fine discretisation of dozens of bins per dimension can produce a combinatorial explosion of symbolic tokens, slowing down pattern-mining algorithms and yielding a flood of spurious candidate rules that must

be painstakingly pruned. To navigate this trade-off, we are exploring adaptive binning strategies that adjust the number and width of bins according to the local variance in each latent channel, and fuzzy SAX approaches that allow each value to belong partially to multiple categories, thereby smoothing the sharp boundaries of hard quantiles.

Real-world time series often exhibit non-stationarity, with statistical properties that drift over weeks or months. A latent encoding learned once at the outset may gradually become misaligned with new data regimes, causing both feature representations and the rules mined from them to lose relevance. To address this, we plan to investigate periodic re-estimation of the ELM-AE, retraining its decoding weights on sliding time windows so that the latent space remains calibrated to the latest dynamics. Alternatively, online sequential ELM variants, [10], [11], can update the auto-encoder's output weights incrementally, without a full recomputation, enabling continuous adaptation as new sensor readings arrive. Despite these challenges, the closed-form analytics and symbolic transparency at the heart of our pipeline offer distinct advantages for real-time digital-twin applications. Because every model update from the initial auto-encoder inversion to the final rule-aware forecast solve reduces to a handful of linear algebra operations, the entire workflow can be implemented in resource-constrained edge environments with predictable execution times. Meanwhile, the explicit ASP clauses that govern forecast behaviour provide an audit trail: engineers can inspect, modify, or shelve individual rules in response to evolving operational requirements without retraining deep networks from scratch. In sum, while tuning discretisation granularity and accommodating non-stationarity demand careful attention, the pipeline's speed, adaptability, and built-in interpretability make it an especially promising candidate for embedded, safety-critical digital twins.

9.2 Leveraging Metadata in Place of Raw Lagged Inputs

In the present pipeline, each forecast begins by constructing a "lag vector" of raw multivariate sensor readings. While this affords maximal fidelity, it also places heavy demands on storage, pre-processing, and denoising, and it may inadvertently embed spurious micro-variations that obscure the true aging or degradation trends we aim to capture. An alternative is to replace the raw-data windows with metadata descriptors, high-level summaries, or semantic annotations derived from the same time series. This strategy offers several avenues for enhancing robustness and interpretability:

1. **Reduced Noise Sensitivity.**
   Metadata such as rolling statistics (mean, variance), trend indicators (slope, curvature), or derivatives (e.g., first $\dot{X}$ and second $\ddot{X}$ for velocity/acceleration, [46], [48]) filter out high-frequency sensor noise. Feeding such compact summaries into the ELM-AE can improve the signal-to-noise ratio of the latent encoding, leading to cleaner discretisation boundaries and more reliable rule extraction. In practice, we would compute metadata streams concurrently with the raw data, then stack them into a lower-dimension input ϕ_{meta}, thereby reducing the hidden-layer size H and mitigating the curse of dimensionality.

2. **Enhanced Semantic Richness.**
   Metadata can encode **contextual flags**, for example, "fast-charge cycle," "low-temperature operation," or "grid-tied inverter mode" that are difficult to recover from raw readings alone. By embedding these discrete event markers into ϕ_{meta}, we enable the rule miner to formulate clauses that explicitly reference operational contexts, such as following: "Raise a *violation fast charge* alert at time T if the battery is being fast-charged at T but its State-of-Health has increased compared to the previous time step." Such metadata-driven rules increase domain alignment and yield a richer ASP library, as described in Section 4.

3. **Adaptive Metadata Selection.**
   Different regimes of operation may call for distinct descriptor sets. For instance, a wind turbine under normal wind speeds benefits from temperature- and vibration-based metadata, whereas extreme gust conditions require power-output indices. We can implement an **adaptive metadata selector** that uses simple clustering on incoming raw windows to choose the most informative metadata subset before each ELM-AE run. This dynamic gating reduces the risk of rule mining over irrelevant features and helps maintain high support/confidence thresholds without rule-based explosion.

4. **Metadata-Aware Rule Pruning.**
   When rules refer to coarse metadata categories (e.g., "high-stress phase" vs. raw vibration peaks), the ground-truth support sets become more stable across training splits. Consequently, our existing pruning pipeline (support $\geqslant$ 2 %, confidence $\geqslant$ 85 %) can operate with **tighter thresholds**, for example, increasing minimum confidence to 90 % without losing critical clauses. This yields a more **compact** and **robust**

rule base that generalises better to out-of-sample conditions.

5. **ELM-AE Architectural Simplification.** Metadata typically inhabit a lower-dimensional manifold than raw data (e.g., 8–12 descriptors vs. 50–100 raw lags). Therefore, the ELM-AE's hidden layer can be **downsized** ($H \approx 2 \times D_{meta}$) while retaining reconstruction quality. A smaller H not only reduces the Moore–Penrose inversion cost but also accelerates online sequential ELM updates when handling non-stationarity as discussed in this Section 9.

6. **End-to-End Metadata–Rule Consistency.** To ensure that metadata replacements do not inadvertently omit crucial patterns, we recommend a **joint metadata-rule discovery validation** step. Here, we alternately train the pipeline on raw data and on metadata-based inputs, then measure the **overlap** in extracted ASP clauses and the **divergence** in forecast error. A high clause-overlap ($> 80\%$) and negligible MAPE increase ($< 2\%$) would certify the metadata pipeline as a faithful surrogate, ready for deployment in edge settings with severe memory or bandwidth constraints.

In summary, transitioning from raw-lag inputs to thoughtfully designed metadata streams can yield a pipeline that is more **noise-resilient, semantically transparent**, and **computationally efficient**, while preserving or even enhancing the quality and rule-consistency of long-term forecasts. Future work will prototype this approach on battery SoH and wind-turbine datasets to quantify its practical benefits and identify the optimal metadata lexicon for each domain.

10 Conclusion

This paper introduced a neuro-symbolic forecasting pipeline that combines the rapid analytics of Extreme Learning Machines (ELMs) with automatic rule discovery and enforcement via Answer Set Programming (ASP). At its core, the pipeline comprises an ELM-based autoencoder (ELM-AE) for latent-space construction, symbolic rule mining from discretized activations, and a rule-aware ELM forecaster (ELM-F) trained via modified ridge regression that embeds these symbolic rules as hinge-loss penalties.

The resulting model delivers long-horizon forecasts that are not only accurate but also intrinsically comply with learned physical and operational constraints. It preserves closed-form training at every step, enabling near real-time execution even on commodity CPUs or embedded devices. Importantly, all symbolic logic is learned automatically from data via association-rule mining and inductive logic programming, requiring no manual specification.

To demonstrate practical feasibility, we plan to evaluate the proposed method on standardized open datasets for long-term time-series forecasting, specifically, the ETTh benchmarks, [1], and BatteryML Framework datasets for Li-ion battery aging, [5], [40]. Preliminary experimentation on pilot runs suggests that the rule-aware ELM pipeline offers systematic improvements over baseline ELMs and LSTMs in both forecast accuracy and logical consistency, including reductions in 24-hour MAPE and elimination of constraint violations under curated rule sets. These early indications motivate a detailed future benchmark suite, which will assess performance under controlled and reproducible experimental conditions.

Future Directions

While the current system shows promise, several extensions can enrich its utility and reach:

1) Benchmarking against advanced models. Structured comparisons against state-of-the-art models like transformers, [4], [41], temporal convolutional networks (TCNs), [42], PINNs, and probabilistic models should be conducted under standardized protocols with datasets like ETTh and BatteryML, [5], [37]. Evaluation metrics should include forecasting accuracy, constraint violation rates, and latency under edge-device limits.

2) Probabilistic symbolic reasoning. Current ASP-based enforcement relies on deterministic hinge penalties. Future development will explore probabilistic ASP frameworks, [30], [31], temporal logic, or fuzzy rule integration to accommodate uncertainty and rule softness, enabling hybrid symbolic-numeric compliance for noisy or heteroskedastic signals.

3) Adaptive rule mining and lifelong model updates. As systems evolve over time, so must their rule sets. Integrating online sequential ELMs, [10], with periodic re-mining from a sliding window can allow real-time structural adaptation. Additionally, mechanisms for scoring, retiring, or reinstating individual rules via continual learning may be explored.

4) Metadata-driven simplification. Feeding high-level descriptors (e.g., $\dot{X}$, SoC bins, semantic labels) instead of raw lags can yield lower-dimensional inputs with better noise tolerance, [46]. Validating that such metadata maintains predictive quality and ASP clause overlap will support edge deployment.

5) Community adoption and tooling. Public code, reproducible pipelines, and a benchmark suite complete with rule metrics could help standardize research on analytic, interpretable time-series models.

In summary, the method offers a principled template for fast, interpretable forecasting in domains demanding physical adherence and transparency. Continued research will extend its symbolic breadth, empirical robustness, and deployment readiness in embedded environments.

Acknowledgement

This paper has been partially funded by **BATCAT**, a project that has received **funding from the European Union's Horizon Europe research** and innovation programme under grant agreement **No 101137725**.

References

[1] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of AAAI*, vol. 35, no. 12, pp. 11 106–11 115, 2021, DOI: 10.1609/aaai.v35i12.17325.

[2] Meire Fortunato, Charles Blundell, and Oriol Vinyals, "Bayesian recurrent neural networks," *arXiv preprint arXiv:1704.02798*, 2017, DOI: 10.48550/arXiv.1704.02798.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017, DOI: 10.48550/arXiv.1706.03762.

[4] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long, "Itransformer: Inverted transformers are effective for time series forecasting," *arXiv preprint arXiv:2310.06625*, 2023, DOI: 10.48550/arXiv.2310.06625.

[5] Florian Stroebl, Ronny Petersohn, Barbara Schricker, Florian Schaeufl, Oliver Bohlen, and Herbert Palm, "A multi-stage lithium-ion battery aging dataset using various experimental design methodologies," *Scientific Data*, vol. 11, p. 1020, 2024, DOI: 10.1038/s41597-024-03859-z.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006, DOI: 10.1016/j.neucom.2005.12.126.

[7] Yi Hui Zhang, He Wang, Zhi Jian Hu, Meng Lin Zhang, Xiao Lu Gong, and Cheng Xue Zhang, *Comparison of the extreme learning machine with the bp neural network for short-term prediction of wind power*, https://doi.org/10.4028/www.scientific.net/AMR.608-609.564, Access Date: 22-05-2025, 2013.

[8] Shifei Ding, Xinzheng Xu, and Ru Nie, "Extreme learning machine and its applications," *Neural Computing and Applications*, vol. 25, pp. 549–556, 2014, DOI: 10.1007/s00521-013-1522-8.

[9] José A Vásquez-Coronel, Marco Mora, and Karina Vilches, "Training of an extreme learning machine autoencoder based on an iterative shrinkage-thresholding optimization algorithm," *Applied Sciences*, vol. 12, no. 18, p. 9021, 2022, DOI: 10.3390/app12189021.

[10] Wei Guo, Tao Xu, Keming Tang, Jianjiang Yu, and Shuangshuang Chen, "Online sequential extreme learning machine with generalized regularization and adaptive forgetting factor for time-varying system prediction," *Mathematical Problems in Engineering*, vol. 2018, no. 1, p. 6 195 387, 2018, DOI: 10.1155/2018/6195387.

[11] Weipeng Cao, Zhong Ming, Zhiwu Xu, Jiyong Zhang, and Qiang Wang, "Online sequential extreme learning machine with dynamic forgetting factor," *IEEE access*, vol. 7, pp. 179 746–179 757, 2019, DOI: 10.1109/ACCESS.2019.2959032.

[12] Zuo Bai, Guang-Bin Huang, Danwei Wang, Han Wang, and M. Brandon Westover, "Sparse extreme learning machine for classification," *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1858–1870, 2014, DOI: 10.1109/TCYB.2014.2298235.

[13] Jiuwen Cao, Kai Zhang, Minxia Luo, Chun Yin, and Xiaoping Lai, "Extreme learning machine and adaptive sparse representation for image classification," *Neural networks*, vol. 81, pp. 91–102, 2016, DOI: 10.1016/j.neunet.2016.06.001.

[14] Peiju Chang, Jiangshe Zhang, Junying Hu, and Zengjie Song, "A deep neural network based on elm for semi-supervised learning of image classification," *Neural Processing Letters*, vol. 48, pp. 375–388, 2018, DOI: 10.1007/s11063-017-9709-0.

[15] M. Sato and H. Tsukimoto, "Rule extraction from neural networks via decision tree induction," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 3, 2001, 1870–1875 vol.3, DOI: 10.1109/IJCNN.2001.938448.

[16] Hilal Meydan and Mert Bal, "Children of the tree: Optimised rule extraction from machine learning models," *Journal of Data Analytics and Artificial Intelligence Applications*, vol. 1, no. 1, pp. 14–35, 2025, DOI: 10.26650/d3ai.1606958.

[17] Jiawei Han, Jian Pei, and Yiwen Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000, DOI: 10.1145/335191.335372.

[18] Stephen Muggleton, *Inductive logic programming*. Morgan Kaufmann, 1992, DOI: 10.1007/BF03037089.

[19] Hendrik Blockeel and Luc De Raedt, "Top-down induction of first-order logical decision trees," *Artificial intelligence*, vol. 101, no. 1-2, pp. 285–297, 1998, DOI: 10.1016/S0004-3702(98)00034-4.

[20] Brian J Taylor and Marjorie A Darrah, "Rule extraction as a formal method for the verification and validation of neural networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, IEEE, vol. 5, 2005, pp. 2915–2920, DOI: 10.1109/IJCNN.2005.1556388.

[21] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen, "Deepred–rule extraction from deep neural networks," in *Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings 19*, Springer, 2016, pp. 457–473, DOI: 10.1007/978-3-319-46307-0_29.

[22] Tarek Besold, Artur Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís Lamb, Priscila Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha, "Neural-symbolic learning and reasoning: A survey and interpretation 1," in *Neuro-symbolic artificial intelligence: The state of the art*, IOS press, 2021, pp. 1–51, DOI: 10.3233/FAIA210348.

[23] C Lee Giles, Steve Lawrence, and Ah Chung Tsoi, "Rule inference for financial prediction using recurrent neural networks," in *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, IEEE, 1997, pp. 253–259, DOI: 10.1109/CIFER.1997.618945.

[24] C Lee Giles, Steve Lawrence, and Ah Chung Tsoi, "Noisy time series prediction using recurrent neural networks and grammatical inference," *Machine learning*, vol. 44, pp. 161–183, 2001, DOI: 10.1023/A:1010884214864.

[25] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck, *A semantic loss function for deep learning with symbolic knowledge*, Jennifer Dy and Andreas Krause, Eds., https://proceedings.mlr.press/v80/xu18h.html, Access Date: 23-05-2025, Oct. 2018.

[26] Malay Ganai and Aarti Gupta, *SAT-based scalable formal verification solutions*. Springer, 2007, DOI: 10.1007/978-0-387-69167-1_5.

[27] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer, *Learning to solve circuit-SAT: An unsupervised differentiable approach*, https://openreview.net/forum?id=BJxgz2R9t7, Access Date: 02-05-2025, 2019.

[28] Yu Zhang, Hui-Ling Zhen, Mingxuan Yuan, and Bei Yu, "Diffsat: Differential maxsat layer for sat solving," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–7, DOI: 10.1145/3676536.3676748.

[29] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou, "Neural logic machines," *arXiv preprint arXiv:1904.11694*, 2019, DOI: 10.48550/arXiv.1904.11694.

[30] Renato Lui Geh, Jonas Gonçalves, Igor Cataneo Silveira, Denis Deratani Mauá, and Fabio Gagliardi Cozman, "Dpasp: A comprehensive differentiable probabilistic answer set programming environment for neurosymbolic learning and reasoning," *arXiv preprint arXiv:2308.02944*, 2023, DOI: 10.48550/arXiv.2308.02944.

[31] Zhun Yang, Adam Ishay, and Joohyung Lee, "Neurasp: Embracing neural networks into answer set programming," *arXiv preprint arXiv:2307.07700*, 2023, DOI: 10.48550/arXiv.2307.07700.

[32] William Cohen, Fan Yang, and Kathryn Rivard Mazaitis, "Tensorlog: A probabilistic database implemented using deep-learning infrastructure," *Journal of Artificial Intelligence Research*, vol. 67, pp. 285–325, 2020, DOI: 10.1613/jair.1.11944.

[33] Fan Yang, Zhilin Yang, and William W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," *arXiv preprint arXiv:1702.08367*, 2017, DOI: 10.48550/arXiv.1702.08367.

[34] A. Benmerrous, L. S. Chadli, A. Moujahid, M. H. Elomari, and S. Melliani, "Generalized cosine family," *Journal of Elliptic and Parabolic Equations*, vol. 8, no. 1, pp. 367–381, 2022, DOI: 10.1007/s41808-022-00156-x.

[35] Abdelmjid Benmerrous, Lalla Chadli, Abdelaziz Moujahid, Mhamed Elomari, and Said Melliani, *Generalized fractional cosine family*, https://www.researchgate.net/profile/Abdelmjid-Benmerrous/publication/369011539_Generalized_Fractional_Cosine_Family/links/6403c55bb1704f343fa1aa5a/Generalized-Fractional-Cosine-Family.pdf, Access Date: 08-04-2025, Mar. 2023.

[36] Abdelmjid Benmerrous, Lalla saadia Chadli, Abdelaziz Moujahid, M'hamed Elomari, and Said Melliani, *Generalized solutions for time ψ-fractional heat equation*, https://www.jstor.org/stable/27386744, Access Date: 23-04-2025, 2023.

[37] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 22 419–22 430, DOI: 10.48550/arXiv.2106.13008.

[38] B Saha and K Goebel, *Battery data set*, https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, Access Date: 23-02-2025, 2007.

[39] Wei He, Nicholas Williard, Michael Osterman, and Michael Pecht, "Prognostics of lithium-ion batteries based on dempster–shafer theory and the bayesian monte carlo method," *Journal of Power Sources*, vol. 196, no. 23, pp. 10 314–10 321, 2011, DOI: 10.1016/j.jpowsour.2011.08.040.

[40] Ruifeng Tan, Weixiang Hong, Jiayue Tang, Xibin Lu, Ruijun Ma, Xiang Zheng, Jia Li, Jiaqiang Huang, and Tong-Yi Zhang, "Batterylife: A comprehensive dataset and benchmark for battery life prediction," *arXiv preprint arXiv:2502.18807*, 2025, DOI: 10.1145/3711896.3737372.

[41] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," *arXiv preprint arXiv:2211.14730*, 2022, DOI: 10.48550/arXiv.2211.14730.

[42] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," in *arXiv preprint arXiv:1803.01271*, 2018, DOI: 10.48550/arXiv.1803.01271.

[43] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019, DOI: 10.1016/j.jcp.2018.10.045.

[44] Thanapol Phungtua-Eng, Yoshitaka Yamamoto, and Shigeyuki Sako, "Dynamic binning for the unknown transient patterns analysis in astronomical time series," in *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, pp. 5988–5990, DOI: 10.1109/BigData52589.2021.9671917.

[45] Jordi Pascual-Fontanilles, Aida Valls, and Pedro Romero-Aroca, "Multivariate data binning and examples generation to build a diabetic retinopathy classifier based on temporal clinical and analytical risk factors," *Knowledge-Based Systems*, vol. 300, p. 112 154, 2024, DOI: 10.1016/j.knosys.2024.112154.

[46] Wladyslaw Homenda and Agnieszka Jastrzebska, "Time-series classification using fuzzy cognitive maps," *IEEE Transactions on Fuzzy Systems*, vol. 28,

no. 7, pp. 1383–1394, 2019, DOI: 10.1109/TFUZZ.2019.2917126.

[47] Agnieszka Jastrzebska, Gonzalo Nápoles, Władysław Homenda, and Koen Vanhoof, "Fuzzy cognitive map-driven comprehensive time-series classification," *IEEE Transactions on Cybernetics*, vol. 53, no. 2, pp. 1348–1359, 2021, DOI: 10.1109/TCYB.2021.3133597.

[48] Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Kamarthi, Aditya B. Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, and B. Aditya Prakash, "Time-mmd: Multi-domain multimodal dataset for time series analysis," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37, Curran Associates, Inc., 2024, pp. 77 888–77 933, DOI: 10.52202/079017-2476.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy) The authors equally contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself This paper has been partially funded by **BATCAT**, a project that has received **funding from the European Union's Horizon Europe research** and innovation programme under grant agreement **No 101137725**.

Conflicts of Interest The authors have no conflicts of interest to declare.