Distributed Tsetlin Machine-Based Learning for Energy-Efficient Wireless Sensor Networks*

Rayan Azzam, Yandi Liu, Fabien Courrèges, Romain Négrier, Jean-Pierre Cances and Raymond Quéré

***Abstract*— This work-in-progress paper proposes a two-level intelligence framework for improving energy efficiency and service continuity in large-scale wireless sensor networks (WSN) operating under QoS constraints and asynchronous node behavior. At the edge level, each sensor node embeds lightweight, interpretable decision logic based on Tsetlin Machine clauses to regulate sensing and communication actions. At the global level, a centralized base station supervises learning indirectly by reconstructing the underlying spatio-temporal sensed field from sparse transmissions using a deep learning model, and by using reconstruction quality as a performance signal for network-level optimization through reinforcement learning. As an initial concrete step toward this paradigm, we present a current implementation where a Tsetlin Machine is instantiated as a sampling policy that learns sparse acquisition masks under a quality reconstruction–sparsity trade-off. We also introduce a dedicated, controllable WSN simulator designed for high-throughput experimentation and report preliminary validation results, including an ns-3-aligned comparison showing comparable network-level statistics and a promising runtime speedup.**

I. INTRODUCTION

Technological advances have accelerated the development of smart connected systems, making wireless sensor networks (WSN) a key component of many IoT applications such as smart cities, environmental monitoring, industry, and healthcare. However, large-scale WSN deployments face major challenges in energy efficiency, data management, network lifetime, service continuity, and quality of service (QoS), particularly under unreliable wireless conditions and asynchronous node operation [1]. Battery-powered nodes are strongly constrained by radiofrequencies activity, sensing, and processing costs, while the growing volume of sensed data further stresses communication and storage resources.

Existing solutions, such as compressive sensing for reduced acquisition [2], clustering-based communication [3], [4], and duty-cycling/scheduling have shown effectiveness at different levels, but often address these issues in isolation and rely on predefined roles or centralized assumptions. Some approaches are also computationally demanding [5], and many do not explicitly capture network-level interactions, including topology effects. Thus, designing an adaptive strategy that jointly manages energy, continuity, and data utility at the network level remains challenging.

*This work has benefited from a government grant managed by the Agence Nationale de la Recherche under the France 2030 program, reference "ANR-22-PEFT-0008". It is conducted at XLIM Laboratory (RUBI team, SRI axis) within the IAD-RCSF initiative, funded by the Nouvelle-Aquitaine Region. CISTEME is involved as a regional technology transfer partner.

All authors are with XLIM Research Institute, Université de Limoges, France {rayan.azzam, fabien.courreges}@unilim.fr

In this context, we propose a two-level learning paradigm combining lightweight distributed Artificial Intelligence (AI) with global supervision. Sensor nodes embed interpretable Tsetlin Machine (TM) clause logic for local, energy-aware decisions [6], while a central base station (BS) evaluates network performance via deep-learning (DL) based signal reconstruction and guides learning through a global reinforcement learning (RL) framework. The objective is to foster an emergent, adaptive network behavior that autonomously maximizes energy efficiency and service continuity, moving beyond predefined roles and complex synchronization.

As a work-in-progress (WiP), this paper focuses on two practical foundations: (i) instantiating the TM as a sampling policy to reduce acquisition/transmission while preserving reconstructability at the BS, and (ii) developping a lightweight WSN simulator designed for RL-scale experimentation. Preliminary results show promising computational efficiency of this simulator, notably an observed execution speed-up compared to ns-3 in comparable scenarios.

II. RELATED WORKS

Existing work on large-scale WSNs improves energy efficiency, data handling, and service continuity through isolated separate axes, such as base-station data reconstruction, clustering-based communication, and activity scheduling, often relying on fixed roles or centralized assumptions that limit adaptability and holistic optimization of energy, continuity, and QoS.

A first major axis relies on Compressive Sensing (CS) to reduce acquisition and transmission by exploiting sparsity and reconstructing signals from sub-Nyquist measurements. It includes spatio-temporal CS that leverages correlations across sensors and time [1], [2], mostly iterative recovery, and does not directly control reconstruction-error deviations under wireless impairments, which complicates QoS guarantees. To face these limitations, deep learning has been introduced for reconstruction from compressed data [5]. In particular, Rousseau's spatio-temporal architecture combining Recurrent Neural Network (RNN)-based prediction, autoencoders, and a discriminator-inspired refinement reports accurate reconstruction of highly undersampled signals with reduced execution time compared to classical CS [7], making it suitable as a base-station module for evaluating the utility of sparse observations.

A second major direction focuses on clustering-based protocols, where cluster heads aggregate and forward data to reduce long-range transmissions [3]. While classical schemes such as LEACH balance energy through random cluster-head

rotation, they can neglect factors such as residual energy, node-to-base-station distance, and topology, leading to early degradation [4]. Numerous variants enhance cluster-head selection using centralized rules, fuzzy logic, energy/density criteria (e.g., HEED), or optimization/game-theoretic formulations (e.g., PSO/ACO) [3], [4]. Nevertheless, comparative studies show persistent trade-offs and strong dependence on scenario and topology, with no method consistently dominating across performance and reliability [4].

Complementary to clustering, activity scheduling reduces radio duty cycle but is often coupled with predefined roles or centralized control, which may reduce flexibility and fail to capture interactions between routing dynamics and continuity under disruptions.

Achieving network-level adaptive behavior that jointly optimizes energy, service continuity, and QoS remains challenging, motivating distributed learning-driven strategies that couple lightweight node decisions with global performance assessment.

III. PRELIMINARIES: TSETLIN MACHINE

This section presents the classical (standard) Tsetlin Machine (TM), which serves as a foundational component of our proposed methodology. Originally introduced in 2018 [6], the TM is a lightweight interpretable learning model, primarily designed for supervised classification. It is a rule-based framework inspired by finite-state automata and boolean logic, learning human-readable propositional logic rules (if-then rules) with low computational complexity. This makes it attractive for resource-constrained environments such as distributed and embedded systems.

Through this section, we describe the classical TM formulation used for classification, but the TM framework is general and includes several variants tailored to different objectives (e.g., regression through Regression Tsetlin Machine (RTM)), while preserving the same clause-automata learning principle.

A. Clause Based-Representation and Voting

The TM operates on a binarized input vector $X = [x_1, x_2, \ldots, x_n]$, where $x_k \in \{0, 1\}$. For each variable, the model considers two literals: the positive literal x_k and its negation $\neg x_k$. The selected literals are combined using logical AND conjunctions to form a set of clauses. The output of a clause C_j for input X is defined as:

$$C_j(X) = \bigwedge_{k \in I_j^1} x_k \ \wedge \ \bigwedge_{k \in I_j^0} \neg x_k \,, \tag{1}$$

where I_j^1 and I_j^0 in (1) represent the indices of the variables whose positive and negative literals are included in clause j, respectively. A clause returns a value of 1 if all included literals evaluate to 1, otherwise it evaluates to 0.

Clauses are divided into two polarities, denoted as positive clauses $\{C_j^+\}$ and negative clauses $\{C_j^-\}$. In a standard binary classification problem, positive clauses vote in favor of the target class, while negative clauses vote against it. The final prediction $\hat{y}$ is obtained by aggregating positive and negative votes (n^+ and n^-), and applying a threshold function as shown in (2):

$$\hat{y} = u\left(\sum_{j=1}^{n^+} C_j^+(X) - \sum_{j=1}^{n^-} C_j^-(X)\right) \tag{2}$$

with $u(.)$ a function that returns 1 if its argument is positive and 0 otherwise. This structure is interpretable since decisions can be traced back to fired clauses and their literals.

B. Learning With Tsetlin Automata

Learning is performed by teams of Tsetlin Automata (TA), one TA per literal, that control clause composition by deciding whether each literal should be included or excluded from this clause. During training, TAs receive stochastic feedback that drives clauses toward patterns supporting correct classification while suppressing misleading activations, and thus increasing the tendency to include useful literals and exclude harmful ones [8]. This feedback is represented as two complementary types, often referred to as Type I and type II feedback. Type I feedback reinforces useful patterns, thus correcting false negative cases and encouraging clauses that should fire for the target class. On the other hand, type II feedback penalizes misleading patterns, thus correcting false positive cases and discouraging clauses that fire when they should not. This feedback depends on a threshold parameter T, which is responsible of controlling granularity and learning, and prevents too many clauses from firing simultaneously. The following formulation in (3) and (4) represents these probabilities as functions of the clause vote sum, where the clause vote sum is typically clipped to [-T, T]:

$$P_{\text{Type I}} = \frac{T - \max\big(-T,\ \min\big(T,\ \sum_{n=1}^{m} C_n^j\big)\big)}{2T}, \tag{3}$$

$$P_{\text{Type II}} = \frac{T + \max\big(-T,\ \min\big(T,\ \sum_{n=1}^{m} C_n^j\big)\big)}{2T}. \tag{4}$$

IV. PROPOSED METHODOLOGY

In this section, we propose a hybrid learning architecture designed to optimize network performance in large-scale WSNs while naturally accommodating asynchronous node operations. This methodology follows a two-level intelligence structure as shown in Fig. 1.

At the first local level, the edge level, each sensor node embeds lightweight decision logic in the form of Tsetlin Clauses (TC) governed by TA. These clauses map local context to operational actions controlling the node's activitiy and communication strategy. At the global level, a central powerful base station monitors the overall network state indirectly through signal reconstruction quality, and provides a feedback signal used to train the centralized TA-driven clause logic through a global RL loop.

To avoid the practical constraints of physical deployments, training is conducted primarily in simulation environment

and then transferred to a real platform with limited fine-tuning via transfer learning, enabling extensive policy exploration while targeting resource-constrained devices.

A. Node Level Intelligence Design (Edge AI)

At this level, each sensor executes local decision-making using TC that receive binary inputs derived from the node's observable context. These inputs include:

- Local sensor measurements m_s: discretized sensor readings (e.g., temperature: above/below a threshold, humidity level: high/low.)
- Remaining energy E_r: the node's current battery level, typically quantized into states (e.g., high/medium/low.)
- Neighborhood information (limited): simple messages received from nearby nodes (e.g., basic channel state or coordination signals.) Complex interaction is avoided due to asynchronicity, but remains important to enable emergent, adaptive collective behavior.

Based on these inputs, the TC output a set of operational actions determining the node's behavior during the next cycle. This includes the following:

- Sleep duration / duty cycle: how long to remain in low-power sleep mode.
- Transmission power: Optimizing wireless transmission power to balance reachability and energy consumption.
- Communication behavior: deciding whether to transmit directly, relay through neighbors, or adjust communication range.
- Adaptive sampling and message flow optimization: to reduce redundancy while preserving reconstruction quality and network efficiency.

Overall, the node-level component implements a lightweight, interpretable policy that couples sensing, communication, and energy conservation through local rule-based decisions.

B. Base Station-Level Supervision and Reconstruction

The BS acts as a global supervisor of the distributed learning process without imposing direct real-time control over individual nodes. It continuously receives messages from active nodes, potentially via multi-hop relaying, resulting in sparse and incomplete spatio-temporal observations of the sensor field. To evaluate the effectiveness of the network, it uses a deep learning reconstruction model proposed by Rousseau [7], which exploits spatio-temporal and cross-signal correlations to reconstruct the underlying sensor field from the partial data stream. This BS then computes a reconstruction quality score Q_c and compares it to a predefined QoS threshold Q_s to assess whether service is maintained. In practice, Q_c is derived from application-relevant criteria such as (i) a reconstruction error [7], and/or (ii) a data-availability rate. For comparability across scenarios, these quantities can be normalized and mapped to a bounded score. Q_s is then set according to the application requirement, either from domain specifications or via calibration on a validation set.

C. Global Reinforcement Learning Framework

Global optimization is formulated through an RL process in which centralized TA act as learning agents whose Include/Exclude decisions determine the literals forming node-level clauses, and thereby shaping the distributed policies executed by sensor nodes (Fig. 1.) In this scheme, the WSN covering the wireless communication, sensing dynamics and energy dynamics constitutes the RL environment. The global state is not assumed to be directly fully observable. It is captured indirectly through the aggregated data reconstructed at the base station, while nodes act based on local observations.

We define the episode reward as the simulated duration (τ_s) during which the network satisfies the QoS constraint, (i.e $Q_c \geq Q_s$.) An episode ends when reconstruction quality drops below the threshold, indicating loss of service continuity. The objective is to maximize the expected reward, which corresponds to maximizing network operational lifetime under QoS constraints. Over repeated episodes, longer τ_s reinforces TA configurations that sustain performance, while shorter durations discourage ineffective clause logic, with TA Type I/II feedback driving clause adaptation.

Although Q_c (and thus τ_s) is evaluated globally at the BS, learning requires attributing this outcome to individual local decisions. To achieve this, we use counterfactual evaluation where the BS perturbs a small subset of local decisions and assigns Type I/II feedback to the corresponding clauses based on the sign of the resulting change in the global objective. This yields an approximate credit signal in a coupled WSN where node effects are interdependent (see section V).

The same formulation can be extended to incorporate topology-aware decisions by augmenting clause inputs with lightweight topological indicators and allowing relay/forwarding or clustering actions. Improvements in connectivity efficiency are then reflected through increased τ_s.

V. CURRENT TSETLIN MACHINE SAMPLING IMPLEMENTATION

Building on the end-to-end methodology introduced in section IV, our current work-in-progress instantiates the node-level TM as a local sampling decision policy that learns when to acquire and transmit measurements under a sparsity-reconstruction quality trade-off. To make the control flow clear and explicit, we represent this current instantiation using a Specification and Description Language (SDL) representation (Fig. 2), which is a state-machine and signal flow formalism commonly used to describe reactive and distributed systems. In our context, our SDL representation is used to explicitly capture the sequence of local node states, the information exchanged with the base station, and the training feedback loop that updates the TM over episodes.

As illustrated in Fig. 2, an episode starts from an initial sparse observation state and iteratively builds the measurement stream that will be used for reconstruction. The process is initialized with an internal state $z = z_0$ and an empty set of acquired samples. At each decision cycle, a TM-based policy that is implemented through TC receives the current state z with lightweight time/index features and outputs

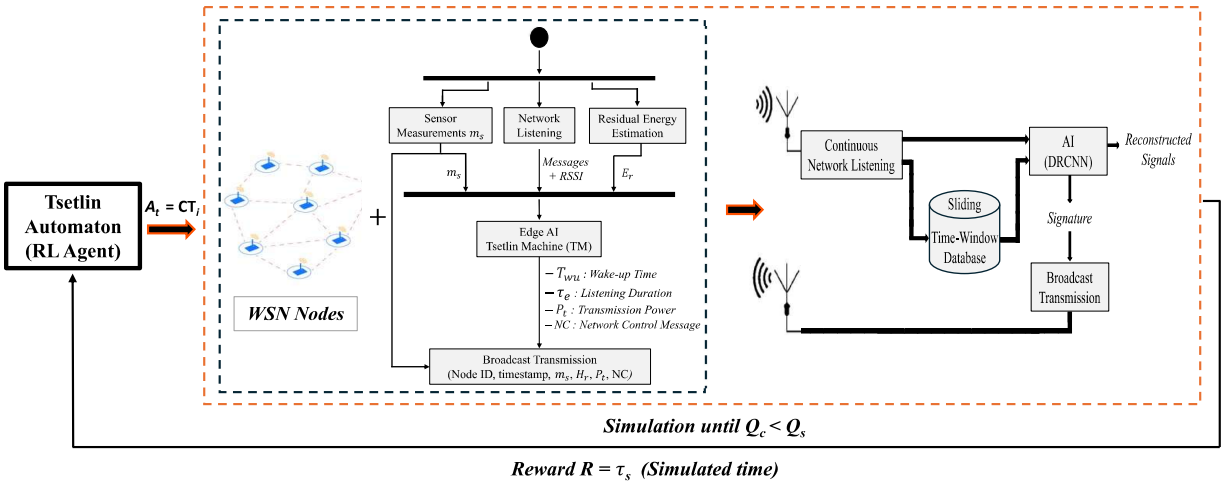

Fig. 1. Overview of the Proposed Approach

the next sampling interval Δ_n. This Δ_n is equivalent to the next sampling instant. The node then enters a sleep state for $(\Delta_n T)$, after which it performs a measurement m and records it with its corresponding index. This acquired information is combined into a packet $y = (m, \Delta_n)$ and transmitted to the base station. The internal state z is then updated and as long as the cumulative progress has not exceeded the signal length N (i.e., while the current index remains within the horizon), the node returns to the TM computation state to predict the next Δ_n. Repeating this loop produces a sparse set of measurements that implicitly defines the sampled vector $y \in \mathbb{R}^N$ which will be assembled and then reconstructed at the base station.

An important and distinctive element of our implementation is that, in addition to producing and determining the immediate next interval Δ_n, the TM may be configured in order to predict a horizon of future intervals $(\Delta_n, \Delta_{n+1}, ..., \Delta_{n+k})$ at each computation step. In execution, only the first predicted interval is applied by the sampler, while the additional predictions are retained as latent variables (Fig. 2). This mechanism is supposed to provide internal temporal context, which is similar to the latent state in recurrent models, thus stabilizing and improving policy learning without changing the actual sampling constraint. Also, although Δ_n is adapted online, we do not expect persistent oscillations under stationary conditions because TA updates are incremental and driven by an episode-level objective that averages performance over multiple decisions. So repeated exposure to similar dynamics yields consistent feedback, reinforcing the same clauses over time and reducing update variance.

At the base station, received packets are stacked to form the sparse observation vector, which is reconstructed as $\hat{x} = R(y)$, where $R(.)$ is intended to be the deep learning reconstruction model introduced previously [7]. The resulting performance is summarized through the following objective: $J(\mathbf{a}) = \alpha\, \tilde{A}(\mathbf{a}) + (1-\alpha)\, \tilde{S}(\mathbf{a})$, where $\tilde{A}(\mathbf{a})$ is a normalized reconstruction error based on $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$, and $\tilde{S}(\mathbf{a})$ is the normalized sparsity factor derived from $S = \|\mathbf{y}\|_0 = \sum_n a_n$, representing the number of revealed samples. The factor $\alpha \in [0, 1]$ controls the trade-off, where larger α emphasizes reconstruction fidelity, while smaller α emphasizes sparsity (lower sampling rate.) To update the TM based on this objective, J is converted into Type I/II feedback applied to the underlying TA (Fig. 2). Since J is defined at the episode level while decisions are taken locally, we propose to derive per-step supervision by evaluating a *limited* set of counterfactual sampling configurations within a forecast horizon W, where each counterfactual candidate requires one reconstruction and loss computation accordingly. Precisely, after computing the baseline $J(\mathbf{a})$, we perturb at most W selected sampling decisions to obtain $J(\mathbf{a}^{(t)})$ and define $\Delta J_t = J(\mathbf{a}^{(t)}) - J(\mathbf{a})$. The sign of ΔJ_t yields a label $z_t \in \{0, 1\}$ for the corresponding feature vector $\mathbf{f}(t)$, which is then used to update the TM. This makes the update cost dominated by $O(W)$ reconstructions evaluations per update, while TM inference/update remains lightweight (clause/feature dependent). Consequently, this SDL-specified sampling loop provides a first operational instantiation of the distributed TM decision component of section IV, while keeping the base-station evaluation mechanism explicit through reconstruction-based loss and TA feedback. Our current implementation is a reproducible and modular stepping stone toward TM–RL training paradigm, linking local decisions to global feedback and updates.

VI. PRELIMINARY RESULTS

To support our proposed TM–RL framework, we developed a lightweight Python-based WSN simulator providing fine-grained control over asynchronous node behavior, energy dynamics, and wireless propagation, while being efficient enough for large numbers of learning episodes. The initial simulator models nodes with local clocks and a state cycle (wake-up → reception → transmission → sleep), message-based communication, and a realistic channel model

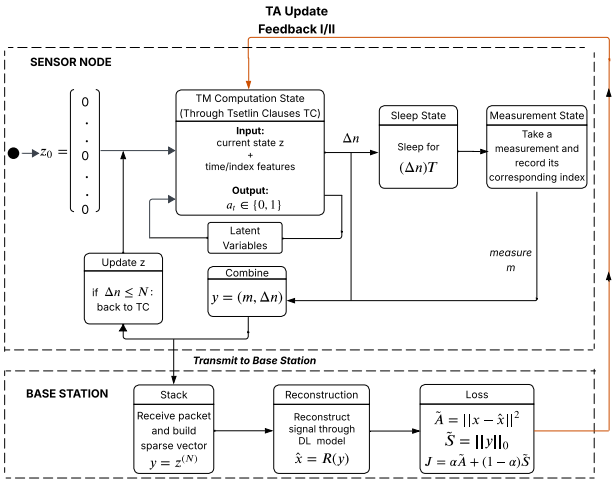

Fig. 2. Tsetlin Machine as a sampling Policy

(path loss, log-normal shadowing, and quasi-static Rayleigh fading.) Packet reception is decided using a received-power sensitivity threshold, and the base station follows the same criterion. Per-node energy is updated using state-dependent costs until depletion. This environment supports future integration of embedded clause-based node logic and BS reconstruction evaluation, while assessing metrics such as network lifetime, node energy consumption and service continuity.

To validate realism and enable comparison with a reference tool, we aligned the simulator with ns-3 under a LoRa-like uplink configuration. This alignment introduces a star topology (end devices $\rightarrow$ gateway), time-on-air (ToA) timing with overlap detection, gateway constraints including spreading-factor-dependent sensitivity, an 8-path demodulation limit with first-come-first-served selection (FCFS), and an energy-based signal-to-interference ratio (SIR) interference test. In addition, it considers that the end devices follow a simplified cycle (standby $\rightarrow$ transmission $\rightarrow$ sleep), and that only gateway performs reception. The energy model was also adapted to ns-3-style accounting using $E = V.I.\Delta t$ with configurable parameters (e.g., $V = 3.3V$, $I_{tx} \approx 28mA$, $I_{standby} \approx 1.4mA$, $I_{sleep} \approx 1.5\mu A$.)

Preliminary comparative benchmarking across multiple network sizes, shows that our Python simulator produces network-level statistics close to ns-3. For conciseness, Table I reports the smallest evaluated scenario where packet delivery ratios (PDR) and lifetime remain comparable and close to ns-3. Large-scale scenarios with higher number of nodes exhibit the same agreement trend. For example, with 200 nodes scenario, the Python simulator yields PDR comparable to ns-3 (about 4.2% and 3.3% until first and last node death accordingly) with almost identical lifetimes. Additional experiments up to 1000 nodes show also the same agreement trend with ns-3 in terms of PDR and network lifetime.

More importantly, and in terms of computational efficiency, our Python simulator shows a promising speedup that increases with network size. To illustrate, for 30 nodes configuration, it completes in $\approx 33ms$ versus $\approx 204ms$ in ns-3 (about 6× faster), while for 200 nodes configuration, it completes in ≈ 210 ms versus ≈ 4210 ms in ns-3 (about $20\times$ faster). This is critical for the large number of episodes required later by RL-based training.

TABLE I

COMPARISON BETWEEN THE PYTHON SIMULATOR AND NS-3.

Simulation Parameters: 30 nodes | 1-byte packet size	**Simulator**	
	Python	**ns-3**
Network Lifetime (Simulation Clock)	18499	18495
Total packets sent	4190	4215
Total packets received	167	167
Packets lost (interference)	0	0
Packets lost (under sensitivity)	4023	4048
Packets lost (no more receivers)	0	0
PDR Until First Node death (%)	4.5	4.59
PDR Until Last Node death (%)	3.98	3.96
Execution duration (ms)	32.304	204

Ongoing work extends the benchmark across broader configurations and finalizing validation before integrating the full TM-driven sampling and end-to-end TM–RL framework.

VII. CONCLUSION AND FUTURE WORK

This WiP paper introduced a two-level intelligence framework for large-scale WSNs, combining lightweight TM-based node decisions with base-station supervision for network optimization. As an initial concrete instantiation of this paradigm, we presented a TM-driven sampling policy that learns sparse acquisition strategies under a quality reconstruction-sparsity trade-off. We also showed preliminary results showing our simulator consistent network statistics with ns-3 and a promising runtime speedup.

Future work includes full simulator validation, integration of the TM logic within nodes, and defining the full RL training loop with QoS–energy optimization metrics. The approach will then be benchmarked against established protocols, scaled to larger and heterogeneous networks (e.g., ZigBee/LoRaWAN), and evaluated on representative smart-city/IoT monitoring scenarios.

REFERENCES

[1] F. Itoua, "Réalisation d'une plate-forme pour l'optimisation de réseaux de capteurs sans fil appliqués au bâtiment intelligent," Ph.D. dissertation, Université de Limoges, Brive-la-Gaillarde, France, Mar. 2018.

[2] S. Yan, C. Wu, W. Dai, M. Ghanem, and Y. Guo, "Environmental monitoring via compressive sensing," in *Proc. Sixth Int. Workshop Knowledge Discovery from Sensor Data (SensorKDD '12)*, 2012, pp. 61–68.

[3] A. I. Al-Sulaifanie, B. K. Al-Sulaifanie, and S. Biswas, "Recent trends in clustering algorithms for wireless sensor networks: A comprehensive review," *Computer Communication*, vol. 191, pp. 395–424, 2022.

[4] M. F. Alomari, M. A. Mahmoud, and R. Ramli, "A systematic review on the energy efficiency of dynamic clustering in a heterogeneous environment of wireless sensor networks (wsns)," *Electronics*, vol. 11, no. 18, p. 2837, 2022.

[5] A. L. Machidon and V. Pejovic, "Deep learning techniques for compressive sensing-based reconstruction and inference – a ubiquitous systems perspective," arXiv preprint, 2021.

[6] O.-C. Granmo, "The tsetlin machine – a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," arXiv preprint, 2018.

[7] S. Rousseau, "Développement de méthodes d'apprentissage profond pour la reconstruction de données spatio-temporelles manquantes – application aux réseaux de capteurs sans fil et aux simulations de structures colloïdales," Ph.D. dissertation, Université de Limoges, Brive-la-Gaillarde, France, 2024.

[8] K. D. Abeyrathna, O.-C. Granmo, X. Zhang, and M. Goodwin, "A scheme for continuous input to the tsetlin machine with applications to forecasting disease outbreaks," arXiv preprint, June 2019.