A multi-step finite-state automaton for arbitrarily deterministic Tsetlin Machine learning

Kuruge Darshana Abeyrathna | Ole-Christoffer Granmo | Rishad Shafik
Lei Jiao | Adrian Wheeldon | Alex Yakovlev | Jie Lei | Morten Goodwin

1Centre for Artificial Intelligence Research, University of Agder, Grimstad, Norway
2Microsystems Research Group, School of Engineering, Newcastle University, Newcastle upon Tyne, UK

Abstract
Due to the high arithmetic complexity and scalability challenges of deep learning, there is a critical need to shift research focus towards energy efficiency. Tsetlin Machines (TMs) are a recent approach to machine learning (ML) that has demonstrated significantly reduced energy compared to neural networks alike, while providing comparable accuracy on several benchmarks. However, TMs rely heavily on energy-costly random number generation to stochastically guide a team of Tsetlin Automata (TA) in TM learning. In this paper, we propose a novel finite-state learning automaton that can replace the TA in the TM, for increased determinism. The new automaton uses multi-step deterministic state jumps to reinforce sub-patterns, without resorting to randomization. A determinism parameter \( d \) finely controls trading off the energy consumption of random number generation, against randomization for increased accuracy. Randomization is controlled by flipping a coin before every \( d \)th state jump, ignoring the state jump on tails. For example, \( d = 1 \) makes every update random and \( d = \infty \) makes the automaton completely deterministic. Both theoretically and empirically, we establish that the proposed automaton converges to the optimal action almost surely. Further, used together with the TM, only substantial degrees of determinism reduce accuracy. Energy-wise, random number generation constitutes switching energy consumption of the TM, saving up to 11 mW power for larger datasets with high \( d \) values. Our new learning automaton approach thus facilitates low-energy ML.

KEYWORDS
learning automata, low-power machine learning, Tsetlin machine

1 | INTRODUCTION

State-of-the-art deep learning (DL) requires massive computational resources, resulting in high-energy consumption (Strubell et al., 2019) and scalability challenges (Chen & Ran, 2019). Thus, there is a critical need to shift research focus towards dealing with energy efficiency (Garcia...
Martin et al., 2019; Shafik et al., 2018. Tsetlin Machines (Granmo, 2018) (TM) are a recent approach to machine learning (ML) that has demonstrated significantly reduced energy usage compared to neural networks alike (Lei et al., 2020; Wheeldon et al., 2020). Using a linear combination of conjunctive clauses in propositional logic, the TM has obtained competitive performance in terms of accuracy (Abeyrathna, Granmo, Zhang, Jiao, & Goodwin, 2019; Berge et al., 2019; Granmo et al., 2019), memory footprint (Granmo et al., 2019), energy (Wheeldon et al., 2020) and learning speed (Granmo et al., 2019; Wheeldon et al., 2020) on diverse benchmarks (image classification, regression and natural language understanding). Furthermore, the rules that TMs build seem to be interpretable, similar to the branches in a decision tree (e.g., in the form if X satisfies condition A and not condition B then Y = 1: Berge et al., 2019). The reported small memory footprint and low-energy consumption make the TM particularly attractive for addressing the scalability and energy challenge in ML.

1.1 | Recent progress on TMs

Recent research reports several distinct TM properties. The TM can be used in convolution, providing competitive performance on MNIST, Fashion-MNIST and Kuzushiji-MNIST, in comparison with CNNs, K-Nearest Neighbour, SVMs, Random Forest, Gradient Boosting, BinaryConnect, Logistic Circuits and ResNet (Granmo et al., 2019). The TM has also achieved promising results in natural language processing, such as text classification (Berge et al., 2019), word sense disambiguation (Yadav et al., 2021a) and sentiment analysis (Yadav et al., 2021b). By introducing clause weights, it has been demonstrated that the number of clauses can be reduced by up to 50×, without loss of accuracy (Phoulady et al., 2020). Further, hyper-parameter search can be simplified with multi-granular clauses, eliminating the pattern specificity parameter (Gorji et al., 2019). By indexing the clauses on the features that falsify them, up to an order of magnitude faster inference and learning has been reported (Gorji et al., 2020). Additionally, regression TMs compare favourably with Regression Trees, Random Forest Regression and Support Vector Regression (Abeyrathna, Granmo, Zhang, & Goodwin, 2019). In Abeyrathna et al. (2021), stochastic searching on the line automata (Oommen, 1997) learn integer clause weights, performing on-par or better than Random Forest, Gradient Boosting and Explainable Boosting Machines. While TMs are binary throughout, thresholding schemes open up for continuous input (Abeyrathna, Granmo, Zhang, & Goodwin, 2019). Finally, TMs have recently been shown to be fault-tolerant, completely masking stuck-at faults (Shafik et al., 2020). The convergence property of TM has recently been studied in (Jiao et al., 2021; Zhang et al., 2020).

1.2 | Paper contributions

TMs rely heavily on energy-costly random number generation to stochastically guide a team of TAs to a Nash Equilibrium of the TM game. In this paper, we propose a novel finite-state learning automaton that can replace the TAs of the TM, for increased determinism. The new automaton uses multi-step deterministic state jumps to reinforce sub-patterns. Simultaneously, flipping a coin to skip every dth state update ensures diversification by randomization. The d-parameter thus allows the degree of randomization to be finely controlled. Both theoretically and empirically, we establish that the proposed new automaton converges to the optimal action almost surely, when it is trained over an infinite time horizon while having infinite number of memory states. We further evaluate the performance of TM with this new automaton empirically on five datasets, demonstrating that the d-parameter can be used to trade-off accuracy against energy consumption.

1.3 | Paper Organization

In Section 2, we introduce our new type of Learning Automaton (LA)—the multi-step variable-structure finite-state LA (MVF-LA). The convergence of the MVF-LA is studied both theoretically and empirically in Section 3. Replacing the TA with MVF-LA, we describe the Arbitrarily Deterministic TM (ADTM) in Section 4. Then, in Section 5, we evaluate ADTM empirically using five datasets. The performance of ADTM is investigated by varying the d-parameter, contrasting against the regular TM and five other state-of-the-art ML algorithms. Effect of determinism on energy consumption is discussed in Section 6. We conclude our work in Section 7.

2 | A MULTI-STEP FINITE-STATE LEARNING AUTOMATON

The origins of LA (Narendra & Thathachar, 2012) can be traced back to the work of M. L. Tsetlin in the early 1960s (Tsetlin, 1961). The objective of an LA is to learn the optimal action through trial and error in a stochastic environment. Various types of LAs are available depending on the nature of the application (Thathachar & Sastry, 2004). Due to their computational simplicity, we here focus on two-action finite-state LA, which we extend by introducing a novel periodically changing structure (variable structure).
An LA interacts with its environment iteratively. In each iteration, the action that a finite-state LA performs next is decided by its present state (the memory). The environment, in turn, randomly produces a reward or a penalty according to an unknown probability distribution, responding to the action selected by the LA. If the finite-state LA receives a reward, it reinforces the action performed by moving to a ‘deeper’ state. If the action results in a penalty, it instead changes state towards the middle state, to weaken the performed action, ultimately switching to the other action. In this manner, with a sufficient number of states, a finite-state LA converges to selecting the action with the highest probability of producing rewards—the optimal action—with probability arbitrarily close to 1.0 (Narendra & Thathachar, 2012).

The transitions between states can be deterministic or stochastic. Deterministic transitions occur with probability 1, while stochastic transitions are randomly performed based on a preset probability. If the transition probabilities are changing, we have a variable-structure automaton, otherwise, we have one with fixed structure. The pioneering TA, depicted in Figure 1, is a deterministic fixed-structure finite-state automaton (Tsetlin, 1961). The state transition graph in the figure depicts a TA with 2N states. States 1 to N maps to Action 1 and states N + 1 to 2N maps to Action 2.

While the TA changes state in single steps, the deterministic Krinsky Automaton introduces multi-step state transitions (Narendra & Thathachar, 2012). The purpose is to reinforce an action more strongly when it is rewarded, and more weakly when penalized. The Krinsky Automaton behaves as a TA when the response from the environment is a penalty. However, when it is a reward, any state from 2 to N transitions to state 1, and any state from N + 1 to 2N − 1 transitions to state 2N. In effect, N consecutive penalties are needed to offset a single reward.

Another variant of LA is the Krylov Automaton. A Krylov Automaton makes both deterministic and stochastic single-step transitions (Narendra & Thathachar, 2012). The state transitions of the Krylov Automaton are identical to those of a TA for rewards. However, when it receives a penalty, it performs the corresponding TA state change randomly, with probability 0.5.

We now introduce our new type of LA, the MVF-LA, shown in Figure 2. The MVF-LA has two kinds of feedback, strong and weak. As covered in the next section, strong feedback is required by the TM to strongly reinforce frequent sub-patterns, while weak feedback is required to make the TM forget infrequent ones. To achieve this, weak feedback only triggers one-step transitions. Strong feedback, on the other hand, triggers s-step transitions. Thus, a single strong feedback is offset by s instances of weak feedback. Further, MVF-LA has a variable structure that changes periodically. That is, the MVF-LA switches between two different transition graph structures, one deterministic and one stochastic. The deterministic structure is as shown in the figure, while the stochastic structure introduces a transition probability 0.5, for every transition. The switch between structure is performed so that every d’th transition is stochastic, while the remaining transitions are deterministic.

### 3 | PROOF OF THE CONVERGENCE OF MVF-LA

In this section, we discuss the convergence of the proposed MVF-LA. In Section 3.1, we use a Markov chain model to analyse the convergence property of the MVF-LA. Thereafter, we simulate the MVF-LA and illustrate its convergence in different conditions in Section 3.2.
3.1 | Proof of the convergence of MVF-LA using Markov chain

To build the Markov chain, we utilize the memory states of the MVF-LA, that is, 1 to 2N, to represent the state space of Markov chain. The transition probability matrix, \( P \), for the Markov chain of MVF-LA is then to be established. The transition from any state \( i \) to another state \( j \) in MVF-LA can happen due to one of four types of feedback: strong reward, strong penalty, weak reward and weak penalty. Apart from boundary conditions, the state transition from \( i \) to \( j \) may also not happen since the every \( d^{th} \) update is made with probability of 0.5. Considering these conditions, the probability of making the transition from \( i \) to \( j \), \( p_{ij} \) can be calculated as follows.

Transition probability due to a strong reward, \( P_{sr} \) can be calculated as:

\[
P_{sr} = P_{Trans} \times (1 - c) \times P_s
\]  

(1)

Here, \( P_{Trans} \) is the probability that any transition to other states happens. It includes two possibilities. (1) Transitions happen \( d - 1 \) times for every \( d \) iteration. (2) Transitions happen with probability 0.5 at the remaining 1 of \( d \) iterations. Therefore, the overall probability of any transition, \( P_{Trans} \), can be calculated as,

\[
P_{Trans} = \frac{d - 1}{d} + 0.5 \times \frac{1}{d}
\]  

(2)

The variable \( c \) in (1) is the penalty probability. The penalty probability \( c \) is the penalty probability of action 1 if the starting state \( i \) in a transition from \( i \) to \( j \) is located in the state space of the action 1, that is, \( 0 < i \leq N \). The penalty probability \( c \) on the other hand is the penalty probability of action 2 if state \( i \) is in the state space of action 2, that is, \( N < i \leq 2N \). The probability \( P_s \) in the same equation is the probability of getting a strong feedback.

Similarly, transition probabilities due to strong penalty: \( P_{sp} \), weak reward: \( P_{wr} \), and weak penalty: \( P_{wp} \) are calculated as in (3), (4) and (5), respectively.

\[
P_{sp} = P_{Trans} \times c \times P_s
\]  

(3)

\[
P_{wr} = P_{Trans} \times (1 - c) \times (1 - P_s).
\]  

(4)

\[
P_{wp} = P_{Trans} \times c \times (1 - P_s).
\]  

(5)

Using the above transition probabilities, we form the transition probability matrix, \( P \) for the MVF-LA in Figure 2. Matrix \( P \) exhibits the Markov chain property that the sum of the probabilities of each raw equals to one: \( \sum_{j} p_{ij} = 1 \). For instance, consider the MVF-LA in Figure 2. Here, when the starting state of a transition is \( N \), a strong reward moves the state from \( N \) to \( N - 3 \) (\( s = 3 \)). Similarly, a weak reward moves the state from \( N \) to \( N - 1 \). Weak and strong penalties, on the other hand, move the state \( N \) in the state space of action 1 to the state space of action 2. While a weak penalty moves the state from \( N \) to \( N + 1 \), a strong penalty moves it to \( N + 3 \). The state \( N \) stays on the same state with probability \( P_{non} \), where \( P_{non} \) is equal to \( 1 - P_{Trans} \). At boundaries, if any of the above transitions cannot be made, that transition probability is accommodated in \( P_{non} \).

The Algorithm 1 shows the step-by-step procedure of building the transition matrix of MVF-LA.
Algorithm 1

Calculating the stationary distribution of the Markov chain for MVF-LA

1: Input: Number of states per action, \( N \); Number of strong jumps, \( s \); Deterministic parameter, \( d \); Probability of getting a strong feedback, \( P_s \); Penalty probability for action 1, \( c_1 \); Penalty probability for action 2, \( c_2 \).
2: Output: Limiting Matrix
3: Initialize: Transition Probability Matrix \( P \) \( \triangleright \) \( P \) requires 2N by 2N space
4: Function:
5: for \( i = 1, ..., 2N \) do
6: if \( i \leq N \) then
7: \( c = c_1 \) \( \triangleright \) \( c \) = Penalty probability for action 1
8: else
9: \( c = c_2 \) \( \triangleright \) \( c \) = Penalty probability for action 2
10: end if
11: for \( j = 1, ..., 2N \) do
12: Compute \( P_{\text{Trans}} \) \( \triangleright \) (2)
13: if \( (i \leq N \) and \( i - s = j \) or \( (i > N \) and \( i + s = j \) then
14: \( P_{ij} = P_{\text{Trans}} \times P_s \times (1 - c) \) \( \triangleright \) strong reward
15: else if \( (i \leq N \) and \( i - s = j \) or \( (i > N \) and \( i + s = j \) then
16: \( P_{ij} = P_{\text{Trans}} \times P_s \times c \) \( \triangleright \) strong penalty
17: else if \( (i \leq N \) and \( i - 1 = j \) or \( (i > N \) and \( i + 1 = j \) then
18: \( P_{ij} = P_{\text{Trans}} \times (1 - P_s) \times (1 - c) \) \( \triangleright \) weak reward
19: else if \( (i \leq N \) and \( i + 1 = j \) or \( (i > N \) and \( i - 1 = j \) then
20: \( P_{ij} = P_{\text{Trans}} \times (1 - P_s) \times c \) \( \triangleright \) weak penalty
21: else (staying on the same state)
22: if \( i = j = 1 \) or \( i = j = 2N \) then \( \triangleright \) both weak and strong updates cannot be made
23: \( P_{ij} = (1 - P_{\text{Trans}}) + P_{\text{Trans}} \times (1 - c) \times (1 - P_s) + P_{\text{Trans}} \times (1 - c) \times P_s \)
24: else if \( i = j \) then
25: if \( (i \leq N \) and \( i - s < 0 \) or \( (i > N \) and \( i + s > 2N \) then \( \triangleright \) only weak updates can be made
26: \( P_{ij} = (1 - P_{\text{Trans}}) + P_{\text{Trans}} \times (1 - c) \times P_s \)
27: else \( \triangleright \) both weak and strong updates can be made
28: \( P_{ij} = 1 - P_{\text{Trans}} \)
29: end if
30: else
31: \( P_{ij} = 0 \)
32: end if
33: end if
34: end for
35: \( P[j,j] \leftarrow \text{Update} \) \( \triangleright \) \( P[j,j] = p_{ij} \)
36: end for
37: 
38: End Function
39: Return: \( (P)^\pi \) \( \triangleright \) Return the stationary distribution

Clearly, this Markov chain is indeed recurrent and non-periodical, and thus it is an ergodic Markov chain. As in Jiao (2020), the probability of staying at a particular state at time \( n \), \( \pi(n) \), in the Markov chain can be computed as,

\[
\pi(n) = \pi(n-1)P = \pi(n-2)P^2 = ... = \pi(0)P^n
\] (6)
Here, \( \pi(n) \) represents all the states in MVF-LA, \( \pi(n) = [\pi_1(n), \pi_2(n), ..., \pi_N(n)] \) and \( \sum \pi_i(n) = 1 \). When \( n \) goes to infinity, we obtain the steady state probabilities, \( \pi^* \). The steady state probabilities \( \pi^* \) are independent of the initial state. Hence, we can also calculate \( \pi^* \) by,

\[
\pi^* = \pi^* P
\]

Theoretically, from (6) and (7), \( \pi^* \) can be obtained by multiplying \( P \) itself for infinite number of times, \( \pi^* = (P)^\infty \). In practice, we multiply \( P \) itself for a sufficiently large number of times until its entries converge and then the steady state probabilities are obtained. Once we know the steady state probabilities, we sum up the steady state probabilities that correspond to the action that has the lowest probability of penalty. If the sum of the probabilities converges to 1 when \( N \) approaches to infinity, we can conclude that, with sufficiently large number of memory states per actions, the LA converges to the correct action. For MVF-LA, based on the calculation of Algorithm 1, we conclude that as long as the best action's penalty probability is less than 0.5, the action selection probability converges to 1 when \( N \) goes to infinity.

To illustrate the convergence property of MVF-LA, we form different transition matrices with distinct parameter configurations using Algorithm 1. We keep \( s, P, \) and \( d \) as constant at 3, 0.67 and 10, respectively for the analysis. Without loss of generality, we always set action 1 as the best action. The steady state probability distribution over the states of MVF-LA can be seen in Figure 3 when \( N = 10 \). The sum of the steady state probabilities of action 1, \( Pr[\text{Action 1}] \), for different penalty probability configurations are also illustrated in the figure. Although with only \( N = 10 \) memories, the action selection probability for action 1 is convincingly higher (\( > 0.94 \)) when \( c_1 = 0.1, c_2 = 0.9 \) and \( c_1 = 0.2, c_2 = 0.4 \).

The probability of selecting action 1 for the remaining penalty probability setups are higher than 0.75. However, the probability distribution of these two setups show that the MVF-LA has not made the decision of selecting action 1 confidently as the steady state probabilities of the end states of action 1 are relatively lower than those of the previous two setups. Nevertheless, theoretically, the probability of selecting action 1 increases with \( N \) and it will reach 1 as \( N \) goes to infinity, given that the best action has a penalty probability less than 0.5. This is verified by the plots in Figure 4 where the probability of selecting action 1 reaches 1 when \( N \) increases for all the cases except when \( c_1 = 0.6 \) and \( c_2 = 0.9 \). This is because the lowest penalty probability (\( c_1 \) in this case) is not less than 0.5. Therefore, even though the difference between penalty probabilities of the case \( c_1 = 0.4 \) and \( c_2 = 0.6 \) (0.2) lower than the case \( c_1 = 0.6 \) and \( c_2 = 0.9 \) (0.3), the case of \( c_1 = 0.4 \) and \( c_2 = 0.6 \) can be perfectly learned when \( N \) increases while the other case struggles. Our Markov chain based analysis thus uncovers similar convergence properties as what is achieved with traditional Tsetlin Automata (TA). However, with increasing determinism, the stochasticity of learning is reduced. This reduction makes the individual learning runs more predictable.
Additionally, the Markov property of TM learning has been utilized previously to analyse learning convergence (Jiao et al., 2021; Zhang et al., 2020). Because TM learning can be formulated as a Markov chain, we can mathematically prove convergence properties. Indeed, apart from proving convergence of the individual learning elements (TA or MVF-LA) within the TM, one can also analyse the complete TM as one unit. Other state-of-the-art ML algorithms do not necessarily decompose into a simple Markov chain, making exact convergence analysis more difficult.

3.2 | Simulation analysis of MVF-LA

In this section, we simulate the MVF-LA and see if it behaves similar to the above stated convergence properties. First, we build the MVF-LA and iteratively update its states by stochastically generating feedbacks for MVF-LA’s actions according to known penalty probabilities. Then we analyse the behaviour of the MVF-LA and compare with its theoretical outputs.

Here we introduce the new quantity $M(n)$, which is the average penalty after $n$ training iterations. The $M(n)$ for a two-action automaton is computed as $M(n) = c_1 \Pr[A\text{ction} 1] + c_2 \Pr[A\text{ction} 2]$ (Narendra & Thathachar, 2012). According to the theory stated in Section 3.1, when $n$ and $N$ go to infinity, the probability of selecting the action which has the lowest penalty probability should reaches 1 (consequently, the probability of selecting the other action goes to 0). Therefore, when $n$ and $N$ go to infinity, the average penalty, $M(n)$ should approximate to the lowest penalty probability.

In our simulation, to make the analysis easier, we always set the lowest penalty probability to the action 1. Then, we first analyse the variation of $\Pr[A\text{ction} 1]$ against the number of training iterations, $n$. Figure 5 depicts the 20-iterations moving average of $\Pr[A\text{ction} 1]$ against the number of iterations. At each experiment round, the number of training iteration, $n$ is increased and the final $\Pr[A\text{ction} 1]$ is recorded. The $N$, $s$, $d$ and $P_s$ in this simulation are fixed at 20, 3, 10 and 0.67. As expected, the $\Pr[A\text{ction} 1]$ increases with $n$. The case with $c_1 = 0.1$ and $c_2 = 0.9$ has already approaches probability 1. The probabilities of selecting action 1 in experiments with $c_1 = 0.4$, $c_2 = 0.6$ and $c_1 = 0.2$, $c_2 = 0.4$ are slowly approaching 1. From these two, $\Pr[A\text{ction} 1]$ variation of the case $c_1 = 0.4$ and $c_2 = 0.6$ is more stable than the other. The $\Pr[A\text{ction} 1]$ variation of the experiment with $c_1 = 0.6$ and $c_2 = 0.9$ has stabilized around 0.8.

![Figure 5](image1.png) The variation of the $\Pr[A\text{ction} 1]$ against the number of training iterations, $n$ for different penalty probabilities

![Figure 6](image2.png) The variation of average penalty $M(n)$ against the number of training iterations, $n$ for different penalty probabilities
The change of the average penalty, \( M(n) \) over \( n \) for the same experiment is illustrated in Figure 6. Except for the experiment with \( c_1 = 0.6 \) and \( c_2 = 0.9 \), \( M(n) \) has approximated to the lowest penalty probability with \( n \). The five-iterations moving average for the experiment with \( c_1 = 0.2 \) and \( c_2 = 0.4 \) is again unsteady. The reason here is both \( c_1 \) and \( c_2 \) are lower than 0.5 and therefore, there is a higher chance to get a reward for both the actions.

In the next arrangement, the change of \( \Pr[\text{Action 1}] \) against \( n \) is studied for distinct \( N \) values. For this experiment, the \( c_1 \) and \( c_2 \) are fixed at 0.4 and 0.6, respectively. As expected, Figure 7 displays that \( \Pr[\text{Action 1}] \) of MVF-LA with higher \( N \) reaches highest possible probability faster.

### 4 | THE ARBITRARILY DETERMINISTIC TM

In this section, we introduce the details of the ADTM, shown in Figure 8, where the TA is replaced by the MVF-LA. The purpose of the ADTM is to control the amount of stochasticity generated, thus allowing management of energy consumption during learning.

#### 4.1 | ADTM inference

##### 4.1.1 | Input features

Like the TM, an ADTM takes a feature vector of \( o \) propositional variables as input, \( X = [x_1, x_2, x_3, \ldots, x_o] \), to be classified into one of two classes, \( y = 0 \) or \( y = 1 \). These features are extended with their negation, to produce a set of literals: \( L = [l_1, l_2, l_3, l_4, \ldots, l_{2o}] \).
4.1.2 | Clauses

Patterns are represented by m conjunctive clauses. As shown for Clause-1 in the figure, a clause in the TM comprises 2o MVF-LAs, each controlling the inclusion of a specific literal. Let the set \( I_j, I_j \subseteq \{1,\ldots,2o\} \) denote the indexes of the literals that are included in clause \( j \). When evaluating clause \( j \) on input literals \( L \), the literals included in the clause are ANDed: \( c_j = \land_{k \in I_j} l_k, j = 1,\ldots,m \). Note that the output of an empty clause, \( I_j = \emptyset \), is 1 during learning and 0 during inference.

4.1.3 | Classification

In order to identify the sub-patterns associated with both of the classes of a two-class ADTM, the clauses are grouped in two. The number of clauses employed is a user set parameter \( m \). Half of the clauses are assigned positive polarity \( (c_j^+ \) \). The other half is assigned negative polarity \( (c_j^-) \). The clause outputs, in turn, are combined into a classification decision through summation and thresholding using the unit step function \( u(v) = 1 \) if \( v \geq 0 \) else 0:

\[
\hat{y} = u \left( \frac{\sum_{j=1}^{m/2} c_j^+(X) - \sum_{j=1}^{m/2} c_j^-(X)}{2T} \right).
\]

That is, classification is based on a majority vote, with the positive clauses voting for \( y = 0 \) and the negative for \( y = 1 \).

4.2 | The MVF-LA game and orchestration scheme

The MVF-LAs in ADTM are updated by so-called Type I and Type II feedback. Depending on the class of the current training sample \( (X,y) \) and the polarity of the clause (positive or negative), the type of feedback is decided. Clauses with positive polarity receive Type I feedback when the target output is \( y = 1 \), and Type II feedback when the target output is \( y = 0 \). For clauses with negative polarity, Type I feedback replaces Type II, and vice versa. In the following, we focus only on clauses with positive polarity.

4.2.1 | Type I feedback

The number of clauses which receive Type I feedback is controlled by selecting them stochastically according to (9):

\[
T = \max(-T, \min(T,v)) \frac{2T}{\max(-T, \min(T,v))}
\]

Above, \( v = \sum_{j=1}^{m/2} c_j^+(X) - \sum_{j=1}^{m/2} c_j^-(X) \) is the aggregated clause output and \( T \) is a user set parameter that decides how many clauses should be involved in learning a particular sub-pattern. Increasing \( T \) proportionally with the number of clauses introduces an ensemble effect, for increased learning accuracy. Type I feedback consists of two kinds of sub-feedback: Type Ia and Type Ib. Type Ia feedback stimulates recognition of patterns by reinforcing the include action of MVF-LAs whose corresponding literal value is 1, however, only when the clause output also is 1. Note that an action is reinforced either by rewarding the action itself, or by penalizing the other action. Type Ia feedback is strong, with step size \( s \) (Figure 2). Type Ib feedback, on the other hand, combats over-fitting by reinforcing the exclude actions of MVF-LAs when the corresponding literal is 0 or when the clause output is 0. Type Ib feedback is weak (Figure 2) to facilitate learning of frequent patterns.

4.2.2 | Type II feedback

Clauses are also selected stochastically for receiving Type II feedback:

\[
T = \max(-T, \min(T,v)) \frac{2T}{\max(-T, \min(T,v))}
\]
Type II feedback combats false positive clause output by seeking to alter clauses that output 1 so that they instead output 0. This is achieved simply by penalizing exclusion of literals of value 0. Thus, when the clause output is 1 and the corresponding literal value of an MVF-LA is 0, the exclude action of the MVF-LA is penalized. Type II feedback is strong, with step size $s$. Recall that in all of the above MVF-LA update steps, the parameter $d$ decides the determinism of the updates.

5 | EMPIRICAL EVALUATION

We now study the performance of ADTM empirically using Bankruptcy, Balance Scale, Breast Cancer, Liver Disorders and Heart Disease datasets. Note that since we seek to achieve a trade-off between TM accuracy and interpretability, we have selected datasets that facilitate interpretation.

The ADTM is compared against regular TMs to assess to what degree learning accuracy suffers from increased determinism. The ADTM is also compared against seven other state-of-the-art ML approaches: Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Decision Trees (DTs), K-Nearest Neighbour (KNN), Random Forest (RF), Gradient Boosted Trees (XGBoost; Chen & Guestrin, 2016) and Explainable Boosting Machines (EBMs; Nori et al., 2019). For comprehensiveness, three ANN architectures are used: ANN-1—with one hidden layer of five neurons; ANN-2—with two hidden layers of 20 and 50 neurons each and ANN-3—with three hidden layers and 20, 150 and 100 neurons. Performance of these predictive models is summarized in Table 6. We compute both F1-score (F1) and accuracy (Acc.) as performance measures. However, due to class imbalance, we emphasize F1-score when comparing the performance of the different predictive models.

5.1 | Bankruptcy

The Bankruptcy dataset contains historical records of 250 companies. The outcome, Bankruptcy or Non-bankruptcy, is characterized by six categorical features. We thus binarize the features using thresholding (Abeyrathna, Granmo, Zhang, Jiao, & Goodwin, 2019) before we feed them into the ADTM. We first tune the hyper-parameters of the TM and the best performance is reported in Table 1, for $m = 100$ (number of clauses), $s = 3$ (step size for MVF-LA) and $T = 10$ (summation target). Each MVF-LA contains 100 states per action. The impact of determinism is reported in Table 1, for varying levels of determinism. As seen, performance is indistinguishable for $d$-values 1, 10 and 100, and the ADTM achieves its highest classification accuracy. However, notice the slight decrease of F1-score and accuracy when determinism is further increased to 500, 1000 and 5000.

Figure 9 shows how training and testing accuracy evolve over the training epochs. Only high determinism seems to influence learning speed and accuracy significantly. The performance of the other considered ML models is compiled in Table 6. The best performance in terms of F1-score for the other models is obtained by ANN-3. However, ANN-3 is outperformed by the ADTM for all $d$-values except when $d = 5000$.

| Table 1 | Performance of TM and ADTM with different $d$ on Bankruptcy |
|---------|-------------------------------------------------------------|
|         | TM | ADTM |
|         | $d = 1$ | $d = 10$ | $d = 100$ | $d = 500$ | $d = 1000$ | $d = 5000$ |
| F1      | 0.998 | 1.000 | 1.000 | 1.000 | 0.999 | 0.999 | 0.988 |
| Acc.    | 0.998 | 1.000 | 1.000 | 1.000 | 0.999 | 0.999 | 0.987 |

Figure 9 Training and testing accuracy per epoch on Bankruptcy
5.2 | Balance scale

The Balance Scale dataset contains three classes: balance scale tip to the right, tip to the left or in balance. The class is decided by the size of the weight on both sides of the scale and the distance to each weight from the centre. Hence the classes are characterized by four features. However, to make the output binary, we remove the ‘balanced’ class ending up with 576 data samples. The ADTM is equipped with 100 clauses. Each MVF-LA is given 100 states per action. The remaining two parameters, that is, s value and T are fixed at 3 and 10, respectively. Table 2 contains the results obtained with TM and ADTM. Even though ADTM uses the same number of clauses as the TM, the performance with regards to F1-score and accuracy is better with ADTM when all updates on MVF-LAs are stochastic. The performance of the ADTM remains the same until the determinism parameter surpasses 100. After that, performance degrades gradually.

Progress of training and testing accuracy per epoch can be found in Figure 10. Each ADTM setup reaches its peak training and testing accuracy and becomes stable within a fewer number of training epochs. As can be seen, accuracy is maintained up to \( d = 100 \), thus reducing random number generation to 1% without accuracy loss. From the results listed in Table 6 for the other ML approaches, EBM achieves the highest F1-score and accuracy.

5.3 | Breast cancer

The Breast Cancer dataset contains 286 patients records related to the recurrence of breast cancer (201 with non-recurrence and 85 with recurrence). The recurrence of breast cancer is to be estimated using nine features: Age, Menopause, Tumour Size, Inv Nodes, Node Caps, Deg Malig, Side (left or right), the Position of the Breast and Irradiation. However, some of the patient samples miss some of the feature values. These samples are removed from the dataset in the present experiment. The ADTM is arranged with the following parameter setup: \( m = 100 \), \( s = 5 \), \( T = 10 \) and the number of states in MVF-LA per action is 100. The classification accuracy of the TM and ADTM are summarized in Table 3. The performance of both TM and ADTM is here considerably lower than for the previous two datasets, and further decreases with increasing determinism. However, the F1 measures obtained by all the other considered ML models are also low, that is, less than 0.500. The highest F1-score is obtained by ANN-1 and KNN.

| TABLE 2 | Performance of TM and ADTM with different \( d \) on balance scale |
| --- | --- |
| | TM | ADTM |
| | \( d = 1 \) | \( d = 10 \) | \( d = 100 \) | \( d = 500 \) | \( d = 1000 \) | \( d = 5000 \) |
| F1 | 0.945 | 0.982 | 0.983 | 0.982 | 0.968 | 0.951 | 0.911 |
| Acc. | 0.948 | 0.980 | 0.981 | 0.980 | 0.935 | 0.894 | 0.793 |

| FIGURE 10 | Training and testing accuracy per epoch on the Balance Scale |

| TABLE 3 | Performance of TM and ADTM with different \( d \) on Breast Cancer |
| --- | --- |
| | TM | ADTM |
| | \( d = 1 \) | \( d = 10 \) | \( d = 100 \) | \( d = 500 \) | \( d = 1000 \) | \( d = 5000 \) |
| F1 | 0.531 | 0.568 | 0.531 | 0.501 | 0.490 | 0.501 | 0.488 |
| Acc. | 0.703 | 0.702 | 0.698 | 0.691 | 0.690 | 0.690 | 0.693 |
The training and testing accuracy progress per epoch is reported in Figure 11, showing a clear degradation of performance with increasing determinism.

### 5.4 Liver disorders

The Liver Disorders dataset\(^5\) was created by BUPA Medical Research and Development Ltd. (hereafter ‘BMRDL’) during the 1980s as part of a larger health-screening database. The dataset consists of seven attributes. However, McDermott and Forsyth (2016) claim that many researchers have used the dataset incorrectly, considering the Selector attribute as the class label. Based on the recommendation of McDermott and Forsyth, we here instead use the Number of Half-Pint Equivalents of Alcoholic Beverages as the dependent variable, binarized using the threshold ≥ 3. The selector attribute is discarded. The remaining attributes represent the results of various blood tests, and we use them as features.

Here, ADTM is given 10 clauses per class, with \( s = 3 \) and \( T = 10 \). Each MVF-LA action possesses 100 states. The performance of ADTM for different levels of determinism is summarized in Table 4. For \( d = 1 \), the F1-score of ADTM is better than what is achieved with the standard TM. In contrast to the performance on previous datasets, the performance of ADTM on Liver Disorders dataset with respect to F1-score does not decrease significantly with \( d \). Instead, it fluctuates around 0.690.

As shown in Figure 12, unlike the other datasets, the ADTM with \( d = 1 \) requires more training rounds than with larger \( d \)-values, before it learns the final MVF-LA actions. It is also unable to reach the training accuracy obtained with higher \( d \)-values. Despite the diverse learning speed, testing accuracy becomes similar after roughly 50 training rounds. The other considered ML models obtain somewhat similar F1-scores, however, only DT, RF and EBM surpass an F1-score of 0.700.

![Figure 11](image1.png)  
**FIGURE 11** Training and testing accuracy per epoch on Breast Cancer

![Figure 12](image2.png)  
**FIGURE 12** Training and testing accuracy per epoch on Liver Disorders

The training and testing accuracy progress per epoch is reported in Figure 11, showing a clear degradation of performance with increasing determinism.

### Table 4 Performance of TM and ADTM with different \( d \) on Liver Disorders

|          | TM | ADTM |
|----------|----|------|
| \( d = 1 \) | 0.648 | 0.705 |
| \( d = 10 \) | 0.694 | 0.692 |
| \( d = 100 \) | 0.692 | 0.692 |
| \( d = 500 \) | 0.689 | 0.692 |
| \( d = 1000 \) | 0.610 | 0.610 |
| \( d = 5000 \) | 0.612 | 0.612 |

|          | TM | ADTM |
|----------|----|------|
| F1       | 0.533 | 0.610 |
| Acc.     | 0.533 | 0.610 |

![Table 4](image3.png)
5.5 | Heart disease

The Heart Disease dataset concerns prediction of heart disease. To this end, 13 features are available, selected among 75. Out of the 13 features, 6 are real-valued, 3 are binary, 3 are nominal and 1 is ordered.

In this case, the ADTM is built on 100 clauses. The number of state transitions when the feedback is strong, \( s \) is equal to 3 while the target, \( T \) is equal to 10. The number of states per MVF-LA action in the ADTM is 100.

As one can see in Table 5, the ADTM provides better performance than TM in terms of F1-score and accuracy when \( d = 1 \). F1-score then increases with \( d \) and peaks at \( d = 100 \). After some fluctuation, it drops to a value of 0.605 when \( d = 5000 \).

Figure 13 shows similar training and testing accuracy for all \( d \)-values, apart from the significantly lower accuracy of \( d = 5000 \).

Out of other ML algorithms, EBM provides the best F1-score, as summarized in Table 6. Even though ANN-1, ANN-2, DT, RF and XGBoost obtain better F1-scores than TM, the F1 scores of ADTM when \( d \) equals to 1, 10, 100, 500 and 1000 are higher.

6 | EFFECTS OF DETERMINISM ON ENERGY CONSUMPTION

Energy consumption of all TM implementations can be positively reduced by using ADTM, since random choice is a key mechanism in learning (see Section 4). This effect is especially notable in ASIC implementations aimed at low-energy on-chip learning applications, where energy overheads are low (e.g., compared to a personal computer).

### TABLE 5 Performance of TM and ADTM with different \( d \) on Heart Disease

|       | TM  | ADTM |
|-------|-----|------|
|       | \( d = 1 \) | \( d = 10 \) | \( d = 100 \) | \( d = 500 \) | \( d = 1000 \) | \( d = 5000 \) |
| F1    | 0.687 | 0.759 | 0.766 | 0.767 | 0.760 | 0.762 | 0.605 |
| Acc.  | 0.672 | 0.778 | 0.780 | 0.783 | 0.773 | 0.781 | 0.633 |

### FIGURE 13 Training and testing accuracy per epoch on Heart Disease

### TABLE 6 Classification accuracy of selected machine learning models

|            | Bankruptcy | Balance Scale | Breast Cancer | Liver Disorder | Heart Disease |
|------------|------------|---------------|---------------|----------------|--------------|
|            | F1         | Acc.          | F1            | Acc.           | F1           | Acc.         |
| ANN-1      | 0.995      | 0.994         | 0.990         | 0.990          | 0.458        | 0.719        |
|            |            |               |               |                | 0.671        | 0.612        |
|            | 0.738      | 0.772         |               |                |              |              |
| ANN-2      | 0.996      | 0.995         | 0.995         | 0.995          | 0.403        | 0.683        |
|            |            |               |               |                | 0.652        | 0.594        |
|            | 0.742      | 0.769         |               |                |              |              |
| ANN-3      | 0.997      | \textbf{0.997}| 0.995         | 0.995          | 0.422        | 0.685        |
|            |            |               |               |                | 0.656        | 0.602        |
|            | 0.650      | 0.734         |               |                |              |              |
| DT         | 0.993      | 0.993         | 0.986         | 0.986          | 0.276        | 0.706        |
|            |            |               |               |                | 0.728        | 0.596        |
|            | 0.729      | 0.781         |               |                |              |              |
| SVM        | 0.994      | 0.994         | 0.887         | 0.887          | 0.384        | 0.678        |
|            |            |               |               |                | 0.622        | 0.571        |
|            | 0.679      | 0.710         |               |                |              |              |
| KNN        | 0.995      | 0.994         | 0.953         | 0.953          | \textbf{0.458} | \textbf{0.755} |
|            |            |               |               |                | 0.638        | 0.566        |
|            | 0.641      | 0.714         |               |                |              |              |
| RF         | 0.949      | 0.942         | 0.859         | 0.860          | 0.370        | 0.747        |
|            |            |               |               |                | \textbf{0.729} | 0.607        |
|            | 0.713      | 0.774         |               |                |              |              |
| XGBoost    | 0.983      | 0.983         | 0.931         | 0.931          | 0.367        | 0.719        |
|            |            |               |               |                | 0.656        | \textbf{0.635} |
|            | 0.701      | 0.788         |               |                |              |              |
| EBM        | 0.993      | 0.992         | \textbf{1.000}| \textbf{1.000}| 0.389        | 0.745        |
|            |            |               |               |                | 0.710        | 0.629        |
|            | 0.783      | \textbf{0.824}|               |                |              |              |

Note: Bold values are the best F1-scores and Accuracies for each dataset.
While software implementations of the TM use centralized pseudorandom number generators (PRNGs) to facilitate the random choices (Figure 14a), the ASIC implementation uses many smaller PRNGs localized to individual TAs to maximize parallelism (Figure 14b). In the ASIC implementation of TM, linear feedback shift registers (LFSRs) are used as PRNGs due to their small size and simplicity Wheeldon et al. (2020).

Power is consumed by the PRNGs in the process of generating a new random number. This is referred to as switching power. In the TM, every TA update is randomized, and switching power is consumed by the PRNGs on every cycle. Additionally, power is also consumed by the PRNGs whilst idle. We term this leakage power. Leakage power is always consumed by the PRNGs whilst they are powered up, even when not generating new numbers.

In the ADTM with hybrid TA where the determinism parameter $d$ is introduced, $d = 1$ would be equivalent to a TM where every TA update is randomized. $d = \infty$ means the ADTM is fully deterministic, and no random numbers are required from the PRNG. If a TA update is randomized only on the $d$th cycle, the PRNGs need only be actively switched (and therefore consume switching power) for $\frac{1}{d}$ portion of the entire training procedure. The switching power consumed by the PRNGs accounts for 7% of the total system power when using a traditional TA (equivalent to $d = 1$). With $d = 100$ this is reduced to 0.07% of the system power, and with $d = 5000$ this is reduced further to 0.001% of the same. It can be seen that as $d$ increases in the ADTM, the switching power consumed by the PRNGs tends to zero.

In the special case of $d = \infty$, the PRNGs are no longer required for TA updates since the TAs are fully deterministic—we can omit these PRNGs from the design and prevent their leakage power from being consumed. The leakage power of the PRNGs accounts for 32% of the total system power. On top of the switching power savings this equates to 39% of system power, meaning large power and therefore energy savings can be made in the ADTM.

Figure 15 shows the number of randomization events for different $d$ values in the case of Heart Disease dataset. As expected, for lower $d$ values, the number of events reduces drastically. For example, in the first iteration this number reduces by $4219 \times$ from $d = 1$ to $d = 5000$. Notice, how the number of these events also reduces further for both cases as the number of iterations increase Granmo (2018). The reduced number of events can be positively leveraged towards power minimization.

Table 7 shows comparative training power consumption per datapoint (i.e., all TAs being updated concurrently) for two different $d$ values: $d = 1$ and $d = 5000$. Typically, the overall power is higher for bigger datasets as they require increased number of concurrent TAs as well as PRNGs. As can be seen, the increase in $d$ value reduces the power consumption by 11 mW in the case of Heart Disease dataset. This saving is made by reducing the switching activity in the PRNGs as explained above. More savings are made by larger $d$ values as the PRNG concurrent switching activities are reduced.

**FIGURE 14** PRNG strategies for (a) software TM and (b) hardware TM

**FIGURE 15** Number of randomization events per epoch for the Heart Disease dataset
CONCLUSION

In this paper, we proposed a novel finite-state learning automaton (MFV-LA) that can replace the Tsetlin Automaton in TM learning, for increased determinism, and thus reduced energy usage. The new automaton uses multi-step deterministic state jumps to reinforce sub-patterns. Simultaneously, flipping a coin to skip every $d^{th}$ state update ensures diversification by randomization. The new $d$-parameter thus allows the degree of randomization to be finely controlled. For example, $d = 1$ makes every update random and $d = \infty$ makes the automaton fully deterministic. First, theoretically, using Markov chain properties, we showed that MVF-LA is able to select the action which has the lowest penalty probability almost surely when both the number of training iterations and memory states are set to infinity. Then, we simulated the MVF-LA and analysed its convergence empirically to support our theoretical inferences. Further, used together with TM, empirical results on five real-world datasets show that overall, only substantial degrees of determinism reduces accuracy. Energy-wise, the PRNG contributes to switching energy consumption within the TM, which can be completely eliminated with $d = \infty$. We can thus use the new $d$-parameter to trade-off accuracy against energy consumption, to facilitate low-energy ML. The proposed scheme can be used as an alternative to any of the low-power ML algorithms. Low-power ML algorithms are mainly used on edge devices of internet of things (IoT) networks. Our approach is particularly useful for robust on-chip learning (Shafik et al., 2020; Wheeldon et al., 2020).

ACKNOWLEDGEMENT

The authors gratefully acknowledge the contributions from Jonathan Edwards at Temporal Computing on strategies for deterministic Tsetlin Machine learning.

CONFLICT OF INTEREST

None.

ENDNOTES

1 An implementation of ADTM can be found at https://github.com/cair/Deterministic-Tsetlin-Machine.
2 Available from https://archive.ics.uci.edu/ml/datasets/qualitative_bankruptcy.
3 Available from http://archive.ics.uci.edu/ml/datasets/balance+scale.
4 Available from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer.
5 Available from https://archive.ics.uci.edu/ml/datasets/Liver+Disorders.
6 Available from https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID

Kuruge Darshana Abeyrathna https://orcid.org/0000-0003-4816-2597

REFERENCES

Abeyrathna, K. D., Granmo, O.-C., & Goodwin, M. (2021). Extending the tsetlin machine with integer-weighted clauses for increased interpretability. IEEE Access, 9, 8233–8248.
Abeyrathna, K. D., Granmo, O.-C., Zhang, X., & Goodwin, M. (2019). A scheme for continuous input to the Tsetlin machine with applications to forecasting disease outbreaks. In International conference on industrial, engineering and other applications of applied intelligent systems (pp. 564–578).
Abeyrathna, K. D., Granmo, O.-C., Zhang, X., Jiao, L., & Goodwin, M. (2019). The regression Tsetlin machine—A novel approach to interpretable non-linear regression. Philosophical Transactions of the Royal Society A, 378, 20190165.
Berge, G. T., Granmo, O.-C., Tveit, T. O., Goodwin, M., Jiao, L., & Matheussen, B. V. (2019). Using the Tsetlin Machine to learn human-interpretable rules for high-accuracy text categorization with medical applications. IEEE Access, 7, 115134–115146. https://doi.org/10.1109/ACCESS.2019.2935416
Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. Proceedings of the IEEE, 107(8), 1655–1674.
Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785–794).

García-Martín, E., Rodrigues, C. F., Riley, G., & Grahn, H. (2019). Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing, 134, 75–88. https://doi.org/10.1016/j.jpdc.2019.07.007

Gorji, S., Granmo, O.-C., Glimslsd, S., Edwards, J., Goodwin, M. (2020). Increasing the inference and learning speed of Tsetlin Machines with clause indexing. In International conference on industrial, engineering and other applications of applied intelligent systems.

Gorji, S. R., Granmo, O.-C., Pohulady, A., & Goodwin, M. (2019). A Tsetlin Machine with multigranular clauses. In Lecture notes in computer science: Proceedings of the thirty-ninth international conference on innovative techniques and applications of artificial intelligence (sgai-2019) (Vol. 11927). Springer.

Granmo, O.-C. (2018). The Tsetlin Machine—A game theoretic bandit driven approach to optimal pattern recognition with propositional logic. arXiv: 1804.01508.

Granmo, O.-C., Glimslsd, S., Jiao, L., Goodwin, M., Omlin, C. W., & Berge, G. T. (2019). The convolutional Tsetlin Machine. arXiv preprint arXiv:1905.09688.

Jiao, L. (2020). Markov chain and stationary distribution. In Channel aggregation and fragmentation for traffic flows (pp. 17–28). Springer.

Jiao, L., Zhang, X., Granmo, O.-C., & Abeyrathana, K. D. (2021). On the convergence of Tsetlin Machines for the XOR operator. arXiv preprint arXiv: 2102.02547.

Lei, J., Wheeldon, A., Shafik, R., Yakovlev, A., & Granmo, O.-C. (2020). From arithmetic to logic based AI: A comparative analysis of neural networks and Tsetlin Machine. In 2020 27th IEEE international conference on electronics, circuits and systems (ICECS) (pp. 1–4).

McDermott, J., & Forsyth, R. S. (2016). Diagnosing a disorder in a classification benchmark. Pattern Recognition Letters, 73, 41–43.

Narendra, K. S., & Thathachar, M. A. (2012). Philosophical Transactions of the Royal Society A. 378. https://doi.org/10.1098/rsta.2019.0593.

Oommen, B. J. (1997). Stochastic searching on the line and its applications to parameter learning in nonlinear optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 27(4), 733–739.

Phoulady, A., Granmo, O.-C., Gorji, S. R., & Phoulady, H. A. (2020). The weighted Tsetlin machine: Compressed representations with clause weighting. In Ninth international workshop on statistical relational AI (STARAI 2020).

Shafik, R., Wheeldon, A., & Yakovlev, A. (2020). Explainability and dependability analysis of learning automata based AI hardware. In IEEE 26th international symposium on on-line testing and robust system design (IOLTS).

Shafik, R., Yakovlev, A., & Das, S. (2018). Real-power computing. IEEE Transactions on Computers, 67(10), 1445–1461.

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In ACL.

Thathachar, M. A. L., & Sastry, P. S. (2004). Networks of learning automata: Techniques for online stochastic optimization. Kluwer Academic Publishers.

Tsetlin, M. L. (1961). On behaviour of finite automata in random medium. Avtomat i Telemekh, 22(10), 1345–1354.

Wheeldon, A., Shafik, R., Rahman, T., Lei, J., Yakovlev, A., & Granmo, O.-C. (2020). Learning automata based energy-efficient AI hardware design for IoT. Philosophical Transactions of the Royal Society A. 378. https://doi.org/10.1098/rsta.2019.0593.

Yadav, R. K., Jiao, L., Granmo, O.-C., & Goodwin, M. (2021a). Human-level interpretable learning for aspect-based sentiment analysis. In Proceedings of AAAI, Vancouver, Canada.

Yadav, R. K., Jiao, L., Granmo, O.-C., & Goodwin, M. (2021b). Interpretability in word sense disambiguation using Tsetlin Machine. In Proceedings of ICAART, Vienna, Austria.

Zhang, X., Jiao, L., Granmo, O.-C., & Goodwin, M. (2020). On the convergence of Tsetlin Machines for the IDENTITY- and NOT operators. arXiv preprint arXiv:2007.14268.

AUTHOR BIOGRAPHIES

Kuruge Darshana Abeyrathna is a PhD research fellow at the University of Agder. He completed his BSc in Mechatronics Engineering at AIT University, Thailand in 2015. Then he directly joined to the Big Data research group at Thammasat University, Thailand for his MSc studies and graduated in 2017. His research interests are in Artificial Neural Networks, Data Mining, Optimization and Operations Research. Currently, at CAIR, he is working on a project which develops a global grid that facilitates real-time compilation, management and analysis of spatio-temporal data.

Ole-Christoffer Granmo is the Founding Director of Centre for Artificial Intelligence Research (CAIR), University of Agder, Norway. He obtained his master's degree in 1999 and the PhD degree in 2004, both from the University of Oslo, Norway. Dr. Granmo has authored in excess of 140 refereed papers with 6 best paper awards within machine learning, encompassing learning automata, bandit algorithms, Tsetlin machines, Bayesian reasoning, reinforcement learning and computational linguistics. He has further coordinated 7+ research projects and graduated 55+ master- and 7 PhD students. Dr. Granmo is also a co-founder of the Norwegian Artificial Intelligence Consortium (NORA). Apart from his academic endeavours, he co-founded the company Anzyz Technologies AS.

Rishad Shafik is a Senior Lecturer in Electronic Systems within the Microsystems Research Group. Prior to this role, he worked in various other research, consultancy and teaching roles at University of Southampton, University of Bristol, NMI and IUT, Bangladesh. Dr. Rishad received PhD and MSc (with distinction) degrees from the University of Southampton in 2010 and 2005, and BSc in Electronic Engineering degree (with distinction) from the Islamic University of Technology (IUT), Bangladesh in 2001. He is one of the editors of the recently published book—‘Energy-efficient Fault-Tolerant Systems’ by Springer USA. He is actively involved in editorial/review roles of a number of IEEE/ACM/IET journals and organisation committees of major international conferences and symposiums (see below); currently he is chairing the
UK Circuits and Systems Workshop 2020 on ‘AI @The Edge’; previously chaired 30th International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT2017) in Cambridge, UK. He has published in excess of 110 articles in major international journals and conference proceedings—several of them were nominated for best paper awards—one won the best paper award at SGAII2020 and the other won the best poster award at ICECS2020. He is a Senior Member of IEEE and IET.

Lei Jiao received the BE degree in Telecommunications Engineering from Hunan University, Changsha, China, in 2005, the ME degree in Communication and Information System from Shandong University, Jinan, China, in 2008, and the PhD degree in Information and Communication Technology from the University of Agder (UiA), Norway, in 2012. He is currently working as an Associate Professor with the Department of Information and Communication Technology, UiA. His research interests include machine learning, resource allocation and performance evaluation for communication and energy systems.

Adrian Wheeldon is a finishing PhD student at the Microsystems Group, Newcastle University, UK. He concurrently holds the position of Senior Research Assistant at the Centre for Electronics Frontiers, University of Southampton, UK. His main focus of research includes asynchronous circuits, AI hardware and AI architectures for energy frugal and pervasive applications. In 2016, he received the ME degree in Electronic Engineering from the University of Southampton, UK.

Alex Yakovlev is an international pioneer of low-power asynchronous circuit design and automation, for which he was elected to Fellow of IET in 2015, IEEE in 2016 and RAEng in 2017. He is Professor of Computing Systems Design at the School of Engineering, Newcastle University, where he has been working since 1991. He received DSc from Newcastle University in 2006, and PhD from St. Petersburg Electrical Engineering Institute (Russia) in 1982, both in the field of asynchronous systems. At Newcastle, since 2000 he is Head of the Microsystems Group and Founder of the Asynchronous Systems Lab, with over 60 PhD alumni. His team is well known for its contributions in designing asynchronous circuits, concurrent systems, Petri nets, metastability and synchronizers. He has published 8 monographs and over 500 papers in top international journals and conferences. He built his expertise in power-proportional computing and design for survivability through a prestigious EPSRC-funded Dream Fellowship in 2011–2012, from which he holds international patents. He has chaired several major conferences (e.g., ASYNC, DATE-WS, PATMOS), and has given many invited/keynote talks on related topics. His most recent work is in the fields of electromagnetic computing and circuit design for machine learning based on Tsetlin automata.

Jie Lei is currently a Research Assistant within the Microsystems Research Group, Newcastle University, UK. He received his MSc in Embedded Systems and IoT from Newcastle University, UK. He is proficient in hardware/software co-design, embedded systems programming and design of communications interfaces for IoT systems. His research interests include design space exploration of AI systems and application design such as voice recognition using the principles of learning automata.

Morten Goodwin received the BSc and MSc degrees from the University of Agder, Norway, in 2003 and 2005, respectively, and the PhD degree from Aalborg University Department of Computer Science, Denmark, in 2011, on applying machine learning algorithms on eGovernment indicators which are difficult to measure automatically. He is a Professor with the Department of ICT, the University of Agder, Deputy Director for Centre for Artificial Intelligence Research, a public speaker and an active researcher. His main research interests include machine learning, including swarm intelligence, deep learning and adaptive learning in medicine, games and chatbots. He has more than 100 peer reviews of scientific publications. He has supervised more than 110 student projects, including Master and PhD theses within these topics, and more than 90 popular science public speaking events, mostly in Artificial Intelligence.

How to cite this article: Abeyrathna, K. D., Granmo, O.-C., Shafik, R., Jiao, L., Wheeldon, A., Yakovlev, A., Lei, J., & Goodwin, M. (2023). A multi-step finite-state automaton for arbitrarily deterministic Tsetlin Machine learning. Expert Systems, 40(4), e12836. https://doi.org/10.1111/exsy.12836