THE CONVOLUTIONAL TSETLIN MACHINE*

A PREPRINT

Ole-Christoffer Granmo
CAIR, University of Agder

Sondre Glimsdal
CAIR, University of Agder

Lei Jiao
CAIR, University of Agder

Morten Goodwin
CAIR, University of Agder

Christian W. Omlin
CAIR, University of Agder

Geir Thore Berge
SSHF and CAIR

May 24, 2019

ABSTRACT

Deep neural networks have obtained astounding successes for important pattern recognition tasks, but they suffer from high computational complexity and the lack of interpretability. The recent Tsetlin Machine (TM) attempts to address this lack by using easy-to-interpret *conjunctive clauses* in propositional logic to solve complex pattern recognition problems. The TM provides competitive accuracy in several benchmarks, while keeping the important property of interpretability. It further facilitates hardware-near implementation since inputs, patterns, and outputs are expressed as bits, while recognition and learning rely on straightforward bit manipulation. In this paper, we exploit the TM paradigm by introducing the *Convolutional Tsetlin Machine* (CTM), as an interpretable alternative to convolutional neural networks (CNNs). Whereas the TM categorizes an image by employing each clause once to the whole image, the CTM uses each clause as a convolution filter. That is, a clause is evaluated multiple times, once per image patch taking part in the convolution. To make the clauses location-aware, each patch is further augmented with its coordinates within the image. The output of a convolution clause is obtained simply by ORing the outcome of evaluating the clause on each patch. In the learning phase of the TM, clauses that evaluate to 1 are contrasted against the input. For the CTM, we instead contrast against one of the patches, *randomly* selected among the patches that made the clause evaluate to 1. Accordingly, the standard Type I and Type II feedback of the classic TM can be employed directly, without further modification. The CTM obtains a peak test accuracy of 99.51% on MNIST, 96.21% on Kuzushiji-MNIST, 89.56% on Fashion-MNIST, and 100.0% on the 2D Noisy XOR Problem, which is competitive with results reported for simple 4-layer CNNs, BinaryConnect, and a recent FPGA-accelerated Binary CNN.

Keywords Tsetlin Machine · Interpretable Machine Learning · Convolutional Methods · Rule Based Learning · Image Analysis · Pattern Recognition

1 Introduction

The Tsetlin Machine (TM) [9] is a novel machine learning paradigm introduced in 2018. It is based on the Tsetlin Automaton (TA) [22], one of the pioneering solutions to the well-known multi-armed bandit problem [19, 8] and the first Finite State Learning Automaton (FSLA) [16]. The TM has the following main properties that make it attractive as a building block for machine learning [9]: (a) it solves complex pattern recognition problems with interpretable propositional formulae. (b) It learns on-line, persistently increasing both training- and test accuracy, before converging to a Nash equilibrium. The Nash equilibrium balances false positive against false negative classifications, while combating overfitting with frequent itemset mining principles. (c) Resource allocation dynamics are leveraged to optimize usage of limited pattern representation resources. By allocating resources uniformly across sub-patterns, local optima are avoided. (d) Inputs, patterns, and outputs are expressed as bits, while recognition and learning rely

*Source code and datasets for this paper can be found at https://github.com/cair/convolutional-tsetlin-machine

on straightforward bit manipulation. This facilitates low-energy consuming hardware design. (e) Finally, the TM has provided competitive accuracy in comparison with classical and neural network based techniques, while keeping the important property of interpretability. This is crucial for high stakes decisions [20].

The TM currently is state-of-the-art in FSLA-based pattern recognition. However, when compared with CNNs, it struggles with attaining competitive accuracy, providing e.g. 98.2% mean accuracy on MNIST (without augmenting the data) [9]. To address this deficiency, we here introduce the Convolutional Tsetlin Machine (CTM), a new kind of TM designed for image classification.

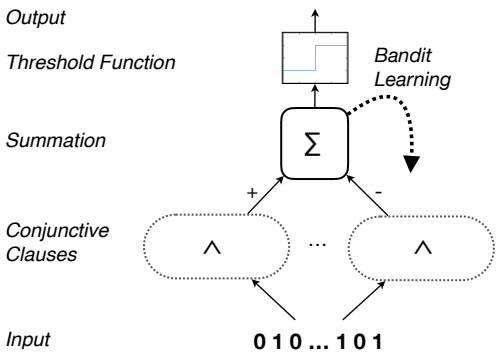
FSLA. The simple Tsetlin Automaton approach has formed the core of more advanced FSLA designs that solve a wide range of problems. This includes decentralized control [23], equi-partitioning [17], streaming sampling for social activity networks [7], faulty dichotomous search [27], and learning in deceptive environments [29], to list a few examples. The unique strength of all of these FSLA designs is that they provide state-of-the-art performance when problem properties are unknown and stochastic, and the problem must be solved as quickly as possible through trial and error.

Rule-based Machine Learning. While the present paper focuses on extending the field of FSLA, we acknowledge the extensive work on rule-based interpretable pattern recognition from other fields of machine learning. Learning propositional formulae to represent patterns in data has a long history. One prominent example is frequent itemset mining for association rule learning [1], for instance applied to predicting sequential events [21, 15]. Other examples include the work of Feldman who investigated the hardness of learning formulae in disjunctive normal form (DNF) [6]. Furthermore, Probably Approximately Correct (PAC) learning has provided fundamental insight into machine learning, as well as a framework for learning formulae in DNF [24]. Approximate Bayesian approaches have recently been introduced to provide more robust learning of rules [25, 10]. However, in general, rule-based machine learning scales poorly and is prone to noise. Indeed, for data-rich problems, in particular those involving natural language and sensory inputs, state-of-the-art rule-based machine learning is inferior to deep learning. The CTM attempts to bridge the gap between the interpretability of rule-based machine learning and the accuracy of deep learning, by allowing the TM to more effectively deal with images.

CNNs. A myriad of image recognition techniques have been reported. However, after AlexNet won the ImageNet recognition challenge by a significant margin in 2012, the entire field of computer vision has been dominated by CNNs. The AlexNet architecture was built upon earlier work by LeCun et al. [13] who introduced CNNs in 1998. Countless CNN architectures, all following the same basic principles, have since been published, including the now state-of-the-art Squeeze-and-Excitation networks [11]. In this paper, we introduce convolution to the TM paradigm of machine learning. By doing so, we simultaneously propose a new kind of convolution filter: an interpretable filter expressed as a propositional formula. Our intent is to address a well-known disadvantage of CNNs, namely, that the CNN models in general are complex and non-transparent, making them hard to interpret. Consequently, the knowledge on why CNNs perform so well and what steps are need to improve the models is limited [28].

Binary CNNs. Relying on a large number of multiply-accumulate operations, training CNNs is computationally intensive. To mitigate this, there is an increasing interest in binarizing the CNNs. With only two possible values for the synapse weights, e.g., -1 and 1, many of the multiplication operations can potentially be replaced with simple accumulations. This could potentially open up for more specialized hardware and more compressed and efficient models. One recent approach is BinaryConnect [3], which reached a near state-of-the-art accuracy of 98.99% on MNIST, and 99.13% in combination with an SVM. Binary CNNs have been further improved with the introduction of XNOR-Nets, which replace the standard CNN filters with binary equivalents [18]. BNN+ [4] is the most recent binary CNN, extending the XNOR-nets with two additional regularization functions and an adaptive scaling factor. The CTM can be seen as an extreme Binary CNN in the sense that it is entirely based on fast summation and logical operators. Additionally, learning in the CTM is bandit based (online learning from reinforcement), while Binary CNNs are based on backpropagation.

Contributions and Paper Outline. Our contributions can be summarized as follows. First, in Section 2, we provide a brief introduction to the TM and a succinct definition of both recognition and learning. Then, in Section 3, we introduce the concept of convolution to TMs. Whereas the classic TM categorizes an image by employing each clause once on the whole image, the CTM uses each clause as a convolution filter. In all brevity, we propose a recognition and learning approach for clause-based convolution that produces interpretable filters. In Section 4, we evaluate the CTM on MNIST, Kuzushiji-MNIST, Fashion-MNIST, and the 2D Noisy XOR problem, and discuss the merits of the new scheme. Finally, we conclude and provide pointers to further work in Section 5.



(a)

0 0 * 1 * 0 0 0
0 0 * 1 * 0 0 0
0 * * 1 * * * 0
0 * * * * 0 0 *
0 0 0 * * 0 0 0
0 * 0 * * * 0 0
0 0 * 1 * * * 0
0 0 0 * 1 * * *

(b)

Figure 1: (a) The basic TM structure. (b) A bit pattern produced by the TM for categorizing 8×8 pixel handwritten digits. The symbol '*' indicates that the corresponding bit can take the value 0 or 1. The remaining bit values require strict matching. The pattern is relatively easy to interpret for humans. It is also efficient to evaluate for computers [9].

2 The Tsetlin Machine

The TM solves complex pattern recognition problems with *conjunctive clauses* in propositional logic, composed by a collective of TAs. It is rooted in game theory, the bandit problem, FSLA games, resource allocation, and frequent itemset mining. The rationale behind the TM can be found in full depth in the original paper on the TM (the paper also includes pseudo code) [9]. Despite the rather complex interplay of several research areas, both recognition and learning in the TM can be defined succinctly as follows.

Structure. The structure of the TM is shown in Figure 1a. The TM takes a vector X of o propositional variables as input: $\mathbf{X} = (x_k) \in \{0,1\}^o$, where o is size of the vector. The input is fed to multiple conjunctive clauses, each of which has the form: $(\bigwedge_{k \in \mathbf{I}^I} x_k) \wedge (\bigwedge_{k \in \bar{\mathbf{I}}^I} \neg x_k)$. Here, $\mathbf{I}^I$ and $\bar{\mathbf{I}}^I$ are non-overlapping subsets of the input variable indexes, $\mathbf{I}^I, \bar{\mathbf{I}}^I \subseteq \{1, \ldots, o\}$, $\mathbf{I}^I \cap \bar{\mathbf{I}}^I = \varnothing$. The subsets decide which portion of the input variables are *Included* in the clause (hence the upper index I), and whether they are negated or not. Conversely, we will in the following denote the variables that are *Excluded* with $\mathbf{I}^E$. To refer to a specific clause j, we will add a lower index j. The input variables from $\mathbf{I}^I$ are included as is, while the input variables from $\bar{\mathbf{I}}^I$ are negated. Jointly, the non-negated and negated variables are referred to as *literals*. The output $c \in \{0,1\}$ of a clause is accordingly fully specified by the input $\mathbf{X}$ and the selection of variables in $\mathbf{I}^I$ and $\bar{\mathbf{I}}^I$. In the TM, each clause is further assigned a fixed polarity $\{-1, +1\}$, which decides whether the clause output is negative or positive. Positive clauses are used to recognize class $y = 1$, while negative clauses are used to recognize class $y = 0$. Finally, a summation operator aggregates the output of the clauses, while a threshold function decides the class $\hat{y} \in \{0,1\}$ predicted for the input $\mathbf{X}$.[2]

Recognition. To effectively deal with negated variables, the first step of inference consists of constructing an augmented input vector $\mathbf{X}' = (x'_k) \in \{0,1\}^{2o}$ by concatenating $\mathbf{X}$ with $\mathbf{X}$ negated:

$$x'_k = \begin{cases} x_k & \text{if } 1 \leqslant k \leqslant o, \\ \neg x_{k-o} & \text{otherwise (i.e., } o+1 \leqslant k \leqslant 2o). \end{cases} \tag{1}$$

The next step is to introduce the collective of TA, which decides the composition of each clause. Assuming a structure with m clauses, half of the m clauses are given negative polarity, while the other half is given positive polarity. We here only define the operations on clauses with positive polarity. The definition for the negative ones is equivalent.

First of all, there are $2o$ TAs per clause j, one for each input $(x'_k) \in \{0,1\}^{2o}$. Each individual TA has a state $a^+_{j,k} \in \{1, \ldots, 2N\}$, and the state of all of the TAs are organized in an $\frac{m}{2} \times 2o$ matrix (m is divided by two since we only consider positive clauses): $\mathbf{A}^+ = (a^+_{j,k}) \in \{1, \ldots, 2N\}^{\frac{m}{2} \times 2o}$. The states decide the actions taken by the TAs. That is, for each clause j, some variables are *Excluded*: $\mathbf{I}^E_j = \left\{k | a^+_{j,k} \leqslant N, 1 \leqslant k \leqslant 2o\right\}$, and some are *Included*: $\mathbf{I}^I_j = \left\{k | a^+_{j,k} > N, 1 \leqslant k \leqslant 2o\right\}$.

[2] Multi-class pattern recognition problems are modelled by employing multiple instances of this structure, replacing the threshold operator with an *argmax* operator [9].

To calculate the output of the clauses, we also need the indexes of the input variables of value 1: $\mathbf{I}^1_{X'} = \{k|x_k = 1, 1 \leqslant k \leqslant 2o\}$. Here, the lower index $\mathbf{X}'$ refers to the set of input variables and the upper index 1 refers to the variable value used for selection. The output c^+_j of each positive clause, organized as a vector $\mathbf{C}^+ = (c^+_j) \in \{0,1\}^{\frac{m}{2}}$, can then be calculated as follows:

$$c^+_j = \begin{cases} 1 & \text{if } \mathbf{I}^I_j \subseteq \mathbf{I}^1_{X'}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The prediction $\hat{y}$ output by the TM is decided by first summing the output of both the positive $\mathbf{C}^+$ and negative $\mathbf{C}^-$ clauses: $v = \sum_j c^+_j - \sum_j c^-_j$. Finally, a threshold decides the output: $\hat{y} = 0 \leqslant v$.

Learning. Learning in the TM is based on coordinating the collective of TAs using a novel FSLA game that leverages resource-allocation and frequent itemset mining principles [9]. The TM employs two kinds of feedback to provide reinforcement to the TAs: Type I and Type II. Type I feedback combats false negatives, while Type II feedback suppresses false positives. In all brevity, feedback is handed out based on training examples $(\mathbf{X}, y)$, consisting of an input $\mathbf{X}$ and an output y. As illustrated by the feedback loop in Figure 1a (Bandit Learning), feedback is controlled by the sum v and a target value $T \in \mathbb{Z}^>$ set for v by the user. A larger T (with a corresponding increase in the number of clauses) makes the learning more robust. This is because more clauses are involved in learning each specific pattern, introducing an ensemble effect.

For training output $\mathbf{y} = \mathbf{1}$, TAs belonging to positive clauses are given Type I feedback to make v approach T. A matrix $\mathbf{P}^+_1 = (p^+_{j,1}) \in \{0,1\}^{\frac{m}{2}}$ picks out the positive clauses selected for feedback (the lower index of $\mathbf{P}^+_1$ refers to the output $y = 1$):

$$p^+_{j,1} = \begin{cases} 1 & \text{with probability } \frac{T - \max(-T, \min(T,v))}{2T}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here, the lower index of $p^+_{j,1}$ refers to the clause j and the output $y = 1$. As seen, clauses are randomly selected for feedback, with a larger chance of being selected for lower v. We now divide the Type I feedback into two parts, Type Ia and Type Ib. Type Ia reinforces *Include* actions to capture patterns within $\mathbf{X}'$. Only TAs taking part in clauses that output 1 and whose associated variable takes the value 1 are select for Type Ia feedback: $\mathbf{I}^{\text{Ia}}_{C^+} = \{(j,k)|x'_k = 1 \wedge c^+_j = 1 \wedge p^+_{j,1} = 1\}$. Type Ib feedback is designed to reinforce *Exclude* actions to combat over-fitting. Type Ib feedback is handed out to the TAs stochastically, using a user set parameter $s \geqslant 1.0$ (a larger s provides finer patterns). The stochastic part of the calculation is organized in a matrix $\mathbf{Q}^+ = (q^+_{j,k}) \in \{0,1\}^{\frac{m}{2} \times 2o}$ with entries:

$$q^+_{j,k} = \begin{cases} 1 & \text{with probability } \frac{1}{s}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Here, the lower index of $q^+_{j,k}$ refers to the clause j and the TA k, considering the positive clauses. TAs are selected for Type Ib feedback as follows: $\mathbf{I}^{\text{Ib}}_{C^+} = \left\{(j,k)|(x'_k = 0 \vee c^+_j = 0) \wedge p^+_{j,1} = 1 \wedge q^+_{j,k} = 1\right\}$. That is, TAs taking part in clauses that output 0, or whose associated variable takes the value 0, are select for Type Ib feedback, however, only if stochastically selected by $p^+_{j,1} = 1$ and $q^+_{j,k} = 1$. Finally, the states of the TAs are updated by increasing/decreasing the states of the selected TAs with two state update operators $\oplus$ and $\ominus$: $\mathbf{A}^+ \leftarrow \left(\mathbf{A}^+ \oplus \mathbf{I}^{\text{Ia}}_{C^+}\right) \ominus \mathbf{I}^{\text{Ib}}_{C^+}$ These operators add/subtract 1 from the states of the singled out TAs, however, not beyond the given state space.

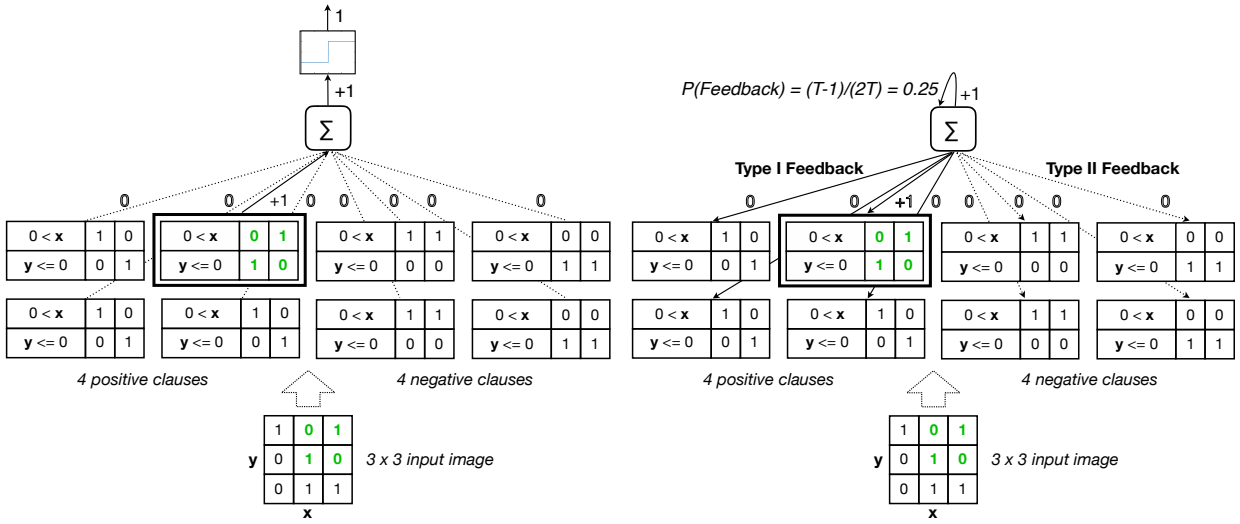
For training output $\mathbf{y} = \mathbf{0}$, TAs belonging to positive clauses are given Type II feedback to suppress clause output 1 (combats false positives). This, together with the negative clauses, make v approach $-T$. A matrix $\mathbf{P}^+_0 = (p^+_{j,0}) \in \{0,1\}^{\frac{m}{2}}$ picks out the positive clauses selected for feedback (the lower index of $\mathbf{P}^+_0$ refers to the output $y = 0$):

$$p^+_{j,0} = \begin{cases} 1 & \text{with probability } \frac{T + \max(-T, \min(T,v))}{2T}, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The lower index of $p^+_{j,0}$ refers to the clause j and the output $y = 0$, respectively. Here too, clauses are randomly selected for feedback, with a larger chance of being selected for higher v. Next, the TAs selected are the ones that will turn clauses that output 1 into clauses that output 0: $\mathbf{I}^{\text{II}}_{C^+} = \{(j,k)|x'_k = 0 \wedge c^+_j = 1 \wedge p^+_{j,0} = 1\}$. Again, the states of the selected TAs are updated using the dedicated operators: $\mathbf{A}^+ \leftarrow \mathbf{A}^+ \oplus \mathbf{I}^{\text{II}}_{C^+}$. All of the above operations are for positive clauses. For negative clauses, Type I feedback is simply replaced with Type II feedback and vice versa!

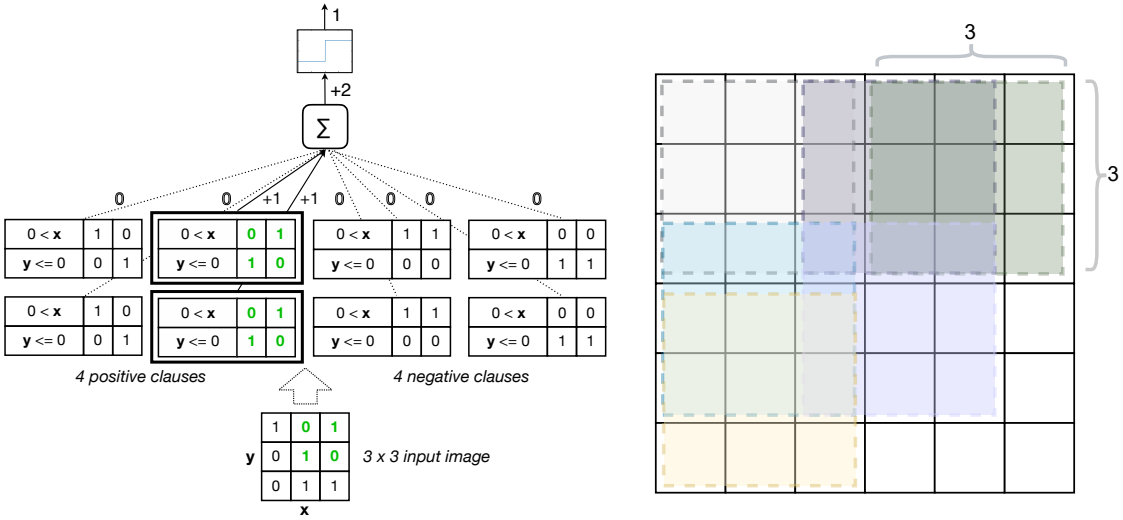
3 The Convolutional Tsetlin Machine

Consider a set of images $\mathcal{X} = \{\mathbf{X}_c|1 \leqslant c \leqslant C\}$, where c is the index of the images. Each image is of size $X \times Y$, and consists of Z binary layers (which together encode the pixel colors using thresholding [5], one-hot encoding, or any



(a) (b)

Figure 2: Example of inference (a) and learning (b) for the Noisy 2D XOR Problem.



(a) (b)

Figure 3: (a) Goal state for the Noisy 2D XOR problem. (b) Illustration of image, filter and patches.

other appropriate binary encoding). A classic TM models such an image with an input vector $\mathbf{X} = (x_k) \in \{0,1\}^{X \times Y \times Z}$ that contains $X \times Y \times Z$ propositional variables. Further, each clause is composed from $X \times Y \times Z \times 2$ literals. As illustrated in Figure 1b, a clause then quite simply decides which bits of a given image layer must take the value 0, which must take the value 1, and which can be ignored, to match the clause. Inspired by the impact convolution has had on deep neural networks, we here introduce the Convolutional Tsetlin Machine (CTM).

Interpretable Rule-based Filters. The CTM uses filters with spatial dimensions $W \times W$, again with Z binary layers. Further, the clauses of the CTM take the role of filters. Each clause is accordingly composed from $W \times W \times Z \times 2$ literals. Additionally, to make the clauses location-aware, we augment each clause with binary encoded coordinates. Location awareness may prove useful in applications where both patterns and their location are distinguishing features, e.g. recognition of facial features such as eyes, eyebrows, nose, mouth, etc. in facial expression recognition. In all brevity, when applying a filter of size $W \times W$ on an image $(x_k) \in \{0,1\}^{X \times Y \times Z}$, the filter will be evaluated on $B = B_X \times B_Y$ image patches. Here, $B_X = \left\lceil \frac{X-W}{d} \right\rceil + 1$ and $B_Y = \left\lceil \frac{Y-W}{d} \right\rceil + 1$, with d being the step size of the convolution. Each image patch thus has a certain location within the image, and we augment the input vector with the

coordinates of this location. We denote the resulting augmented input vector $\mathbf{X}^b$: $\mathbf{X}^b = (x_k) \in \{0,1\}^{W \times W \times Z + B_X + B_Y}$. As seen, the input vector is extended with one propositional variable per position along each dimension, with the position being encoded using thresholding [5] or one-hot encoding. Figure 3b illustrates an example of the image, patches, and a filter for $X = Y = 6$, $W = 3$, and $d = 2$. In this example, the 3×3 filter moves from left to right, from top to bottom, 2 pixels per step.

Recognition. The CTM uses the classic TM procedure for recognition (see Section 2). However, for the CTM each clause outputs B values per image (one value per patch), as opposed to a single output for the TM (Eq. 2). We denote the output of a positive clause j on patch b by $c_j^{b,+}$. To turn the multiple outputs $c_j^{1,+}, \ldots, c_j^{B,+}$ of clause j into a single output denoted by c_j^+, we simply OR the individual outputs:

$$c_j^+ = \bigvee_{b=1}^{B} c_j^{b,+}. \tag{6}$$

Learning. Learning in the CTM leverages the TM learning procedure. As seen in Section 2, Type Ia, Type Ib, and Type II feedback are influencing each clause j based on the content of the augmented input vector $\mathbf{X}'$. For the CTM, the input vector is an image patch, and there are B patches in an image. There is thus B augmented inputs $\mathbf{X}'^b$, $1 \leqslant b \leqslant B$, per clause. Therefore, to decide which patch to use when updating a clause, the CTM randomly selects a single patch among the patches that made the clause evaluate to 1. The clause is then updated according to this patch. That is, an augmented input $\mathbf{X}'^b$ is drawn from the set: $\{\mathbf{X}'^b | c_j^{b,+} = 1, 1 \leqslant b \leqslant B\}$. Observe that if the set is empty, only Type Ib feedback is applicable, and then the augmented input vector is not needed. For non-empty sets, the TAs to be updated are finally singled out using the randomly selected patch:

$$\mathbf{I}_{C+}^{\text{Ia}} = \left\{(j,k) | x_k^b = 1 \wedge c_j^+ = 1 \wedge p_{j,1}^+ = 1\right\}, \tag{7}$$

$$\mathbf{I}_{C+}^{\text{Ib}} = \left\{(j,k) | (x_k^b = 0 \vee c_j^+ = 0) \wedge p_{j,1}^+ = 1 \wedge q_{j,k}^+ = 1\right\}, \tag{8}$$

$$\mathbf{I}_{C+}^{\text{II}} = \left\{(j,k) | x_k^b = 0 \wedge c_j^+ = 1 \wedge p_{j,0}^+ = 1\right\}. \tag{9}$$

The reason for randomly selecting a patch is to have each clause extract a certain sub-pattern, and the randomness of the uniform distribution statistically spread the clauses for different sub-patterns in the target image. Finally, observe that the computational complexity of the CTM grows linearly with the number of clauses m, and with the number of patches B. However, the computations can be easily parallelized due to their decentralized nature.

Step-by-step Walk-through of Inference on Noisy 2D XOR. Rather than providing hand-crafted features which can be used for image classification, the CTM learns feature detectors. We will explain the workings of the CTM by an illustrative example of noisy 2D XOR recognition and learning (see Figure 3 and Section 4). Consider the CTM depicted in Figure 2a. It consists of four positive clauses which represent XOR patterns that must be present in a positive example image (positive features) and four negative clauses which represent patterns that will not trigger a positive image classification (negative features). The number of positive and negative clauses is a user-defined parameter. The bit patterns inside each clause are represented by the output of four TA, one for each bit in a 2×2 filter.

Consider the 3×3 image shown in Figure 2b. The filter represented by the second positive clause matches the patch in the top-right corner of the image and it is the only clause with output 1; similarly, none of the negative clauses respond since their patterns do not match the pattern found in the current patch (Figure 2b). Thus, the Tsetlin Machine's combined output is $v = 1$. Learning of feature detectors proceeds as follows: With the CTM's threshold value set to $T = 2$, the probability of feedback is $\frac{T-v}{2T} = \frac{1}{4}$, and thus learning taking place, which pushes the CTM's output v towards $T = 2$. Note that Type I feedback reinforces true positive output and reduces false negative output whereas Type II feedback reduces false positive output.

A subsequent state of the CTM is shown in Figure 3a. Note that there are now two positive clauses which detect their pattern in the top-right corner patch. The combined output of all clauses is 2; thus, no further learning is necessary for the detection of the XOR pattern in this patch. Also, the location of the occurrence of each pattern is included. The location information uses a bit representation as follows: Suppose an XOR pattern occurs at the three X-coordinates 1, 4, and 6. For the corresponding binary location representation, these coordinates are considered thresholds: If a coordinate is greater than a threshold, then the corresponding bit in the binary representation will be 0; otherwise, it is set to 1. Thus, the representation of the X-coordinates 1, 4, and 6 will be '111', '011' and '001', respectively. These representations of the location of 2×2 patterns are also learned by TAs.

4 Empirical Results

In this section, we evaluate the CTM on four different datasets :

2D Noisy XOR. The 2D Noisy XOR dataset contains 4×4 binary images, 2500 training examples and 10 000 test examples. The image bits have been set randomly, expect for the 2×2 patch in the upper right corner, which reveals the class of the image. A diagonal line is associated with class 1, while a horizontal or vertical line is associated with class 0. Thus the dataset models a 2D version of the XOR-relation. Furthermore, the dataset contains a large number of random non-informative features to measure susceptibility towards the curse of dimensionality. To examine robustness towards noise we have further randomly inverted 40% of the outputs in the training data.

MNIST. The MNIST dataset has been used extensively to benchmark machine learning algorithms, consisting of 28×28 grey scale images of written digits [13].

Kuzushiji-MNIST. This dataset contains 28×28 grayscale images of Kuzushiji characters, cursive Japanese. Kuzushiji-MNIST is more challenging than MNIST because there are multiple distinct ways to write some of the characters [2].

Binary Fashion-MNIST. This dataset contains 28×28 grayscale images of articles from the Zalando catalogue, such as t-shirts, sandals, and pullovers [26]. This dataset is quite challenging, with a human accuracy of 83.50%. We binarize these data by thresholding on 25. We selected a rather low grey value, to capture the complete shape of the articles.

The latter three datasets contain 60 000 training examples and 10 000 test examples. We augmented MNIST with 140 000 training images using InfiMNIST [14] and KMNIST with 120 000 training images using a random scaling factor in the range $[0.9 - 1.1]$. Further, we encoded the pixel values using four bits ($|Z| = 4$) based on uniformly distributed thresholds. Table 2 reports test accuracy for the CTM, while Table 1 contains the corresponding configurations. The

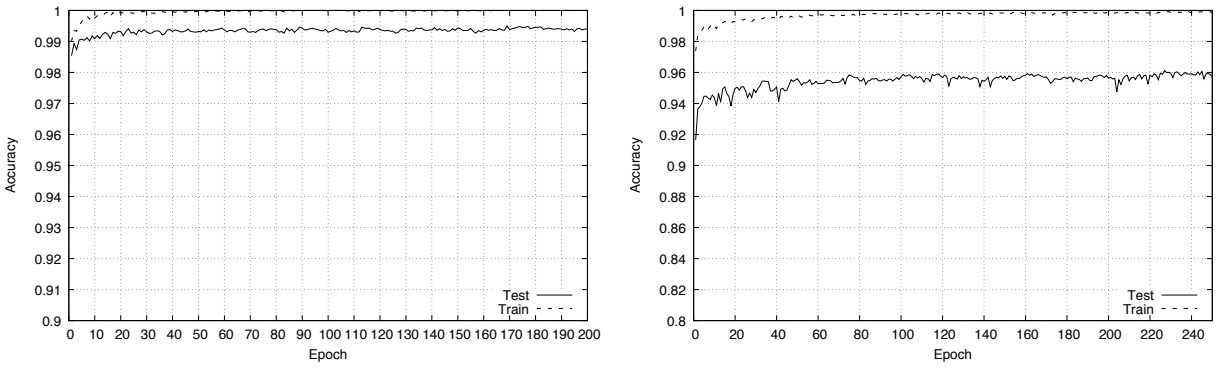
	Search Range	2D Noisy XOR	MNIST	K-MNIST	Fashion-MNIST
#Class Clauses	$1 - 20\,000$	40	8 000	8 000	16 000
T	$1 - 400$	60	200	100	80
s	$1.0 - 40.0$	3.9	10.0	10.0	30.0
W	$2 - 28$	2	10	10	11
$\|Z\|$	$1 - 10$	1	4	4	1

Table 1: CTM configurations.

Model	2D N-XOR	MNIST	K-MNIST	F-MNIST
4-Nearest Neighbour [2, 26]	61.62	*97.14*	*91.56*	85.90
SVM [2, 26]	94.63	*98.57*	*92.82*	87.08
Simple CNN [2]	99.4	*99.06*	*95.12*	90.00
BinaryConnect [3]	-	*98.99*	-	-
FPGA-accelerated BNN [12]	-	*98.70*	-	-
CTM (Mean)	99.99 ± 0.0	99.37 ± 0.0	95.88 ± 0.02	88.34 ± 0.06
CTM (95 %ile)	100.0	99.45	96.10	89.28
CTM (Peak)	100.0	99.51	96.21	89.56
PreActResNet-18 [2]	-	*99.56*	*97.82*	*92.00*
ResNet18 + VGG Ensemble [2]	-	*99.60*	*98.90*	-

Table 2: Empirical results - test accuracy in percent.

hyperparameters were found using a grid search for the given ranges [**?**]. The results are based on ensemble averages, obtained from the last 100 epochs of 400, with 5 replications of each experiment. The CTM performs rather robustly from run to run, providing tight 95% confidence intervals for the mean performance. While this evaluation focuses on the CTM, we have also included results for selected popular algorithms, as points of reference. Results listed in italic are reported in the corresponding papers. Results for BinaryConnect and FPGA-accelerated BNNs on K-MNIST and Fashion-MNIST were not available, so are not reported. Notice that the CTM outperforms the binary CNNs on MNIST, as well as a simple 4-layer CNN, an SVM and a 4-nearest neighbour configuration. However, it is outperformed by the more advanced deep learning network architectures PreActResNet-18 and ResNET18+VGG. Figure 4a depicts training and test accuracy for the CTM on MNIST, epoch-by-epoch, in a single run. Test accuracy peaks at 99.50% after 168 epochs and 99.51% after 327 epochs. Figure 4b contains corresponding results for Kuzushiji-MNIST. Here,



(a) (b)

Figure 4: Single-run accuracy per epoch for CTM on (a) MNIST and (b) K-MNIST.

test accuracy peaks at 96.06% after 244 epochs and 96.15% after 306 epochs. Further, test accuracy climbs quickly in the first epochs, passing 99% already in epoch 2 for MNIST. For both datasets, training accuracy approaches 100%.

5 Conclusion and Further Work

This paper introduced the Convolutional Tsetlin Machine (CTM), leveraging the learning mechanism of the Tsetlin Machine (TM). Whereas the TM categorizes images by employing each clause once per $X \times Y$ input, the CTM uses each clause as a $W \times W$ convolution filter. The filters learned by the CTM are interpretable, being formulated using propositional formulae (see Figure 1b). To make the clauses location-aware, each patch is further enhanced with its coordinates within the image. Location awareness may prove useful in applications where both patterns and their location are distinguishing features, e.g. recognition of facial features such as eyes, eyebrows, nose, mouth, etc. in facial expression recognition. By *randomly* selecting which patch to learn from, the standard Type I and Type II feedback of the classic TM can be employed directly. In this manner, the CTM obtains results on MNIST, Kuzushiji-MNIST, Fashion-MNIST, and the 2D Noisy XOR Problem that compares favorably with simple 4-layer CNNs as well as two binary neural network architectures.

In our further work, we intend to investigate more advanced binary encoding schemes, to go beyond grey-scale images (e.g., addressing CIFAR-10 and ImageNet). We further intend to develop schemes for deeper CTMs, with the first step being a two-layer CTM, to introduce more compact and expressive patterns with nested propositional formulae.

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[2] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep Learning for Classical Japanese Literature. *arXiv:1812.01718*, Dec 2018.

[3] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[4] S. Darabi, M. Belbahri, M. Courbariaux, and V. Partovi Nia. BNN+: Improved Binary Network Training. *arXiv:1812.11800*, Dec 2018.

[5] K. Darshana 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:1905.04199*, May 2019.

[6] V. Feldman. Hardness of Approximate Two-Level Logic Minimization and PAC Learning with Membership Queries. *Journal of Computer and System Sciences*, 75(1):13–26, 2009.

[7] M. Ghavipour and M. R. Meybodi. A streaming sampling algorithm for social activity networks using fixed structure learning automata. *Applied Intelligence*, 2018.

[8] J. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B (Methodological)*, 41(2):148–177, 1979.

[9] O.-C. Granmo. The Tsetlin Machine - A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic. *arXiv:1804.01508*, Apr 2018.

[10] J. R. Hauser, O. Toubia, T. Evgeniou, R. Befurt, and D. Dzyabura. Disjunctions of Conjunctions, Cognitive Simplicity, and Consideration Sets. *Journal of Marketing Research*, 47(3):485–496, 2010.

[11] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[12] C. Lammie, W. Xiang, and M. Rahimi Azghadi. Accelerating Deterministic and Stochastic Binarized Neural Networks on FPGAs Using OpenCL. *arXiv:1905.06105*, May 2019.

[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[15] T. McCormick, C. Rudin, and D. Madigan. A Hierarchical Model for Association Rule Mining of Sequential Events: An Approach to Automated Medical Symptom Prediction. *Annals of Applied Statistics*, 2011.

[16] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall, Inc., 1989.

[17] B. J. Oommen and D. C. Ma. Deterministic Learning Automata Solutions to The Equipartitioning Problem. *IEEE Transactions on Computers*, 37(1):2–13, 1988.

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[19] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 1952.

[20] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[21] C. Rudin, B. Letham, and D. Madigan. Learning theory analysis for association rules and sequential event prediction. *Journal of Machine Learning Research*, 14:3441–3492, 2013.

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

[23] B. Tung and L. Kleinrock. Using Finite State Automata to Produce Self-Optimization and Self-Control. *IEEE Transactions on Parallel and Distributed Systems*, 7(4):47–61, 1996.

[24] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[25] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. A Bayesian Framework for Learning Rule Sets for Interpretable Classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

[27] A. Yazidi and B. John Oommen. On the analysis of a random walk-jump chain with tree-based transitions and its applications to faulty dichotomous search. *Sequential Analysis*, 37:31–46, Jan 2018.

[28] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[29] J. Zhang, Y. Wang, C. Wang, and M. Zhou. Symmetrical Hierarchical Stochastic Searching on the Line in Informative and Deceptive Environments. *IEEE Transactions on Cybernetics*, 47(3):626 – 635, Jul 2016.