A Novel Tsetlin Machine with Enhanced Generalization

Usman Anjum * and Justin Zhan

Posted Date: 23 September 2024

doi: 10.20944/preprints202409.1767.v1

Keywords: Tsetlin Machines; regularization; classification; propositional logic; explainable AI; few-shot learning

 doi:10.20944/preprints202409.1767.v

Article

A Novel Tsetlin Machine with Enhanced Generalization

Usman Anjum * and Justin Zhan

University of Cincinnati; zhanjt@ucmail.uc.edu
* Correspondence: anjumun@ucmail.uc.edu

Abstract: The Tsetlin Machine (TM) is a novel machine learning approach that implements propositional logic to perform various tasks such as classification and regression. The TM not only achieves competitive accuracy in these tasks but also provides results that are explainable and easy to implement using simple hardware. The TM learns using clauses based on the features of the data, and final classification is done using a combination of these clauses. In this paper, we propose the novel idea of adding regularizers to the TM, referred to as Regularized TM (RegTM), to improve generalization. Regularizers have been widely used in machine learning to enhance accuracy. We explore different regularization strategies and their influence on performance. We demonstrate the feasibility of our methodology through various experiments on benchmark datasets.

Keywords: Tsetlin Machines; regularization; classification; propositional logic; explainable AI; few-shot learning

1. Introduction

The Tsetlin Machine (TM) is a novel machine learning algorithm first proposed in 2018 by Granmo et al. [1]. It is based on the Tsetlin Automaton (TA). The TM is able to find complex patterns using propositional logic. It learns by performing an action that can trigger a reward or penalty. The TM can have applications in not only pattern recognition but also in other domains and tasks, such as regression, healthcare, and natural language processing. TMs have the advantage of low memory usage and increased learning speed while achieving competitive predictive performance in several benchmark datasets [2,3]. The TM can easily be implemented using simple low-energy consumption hardware [4,5].

Using propositional logic in the TM has many added advantages over traditional machine learning algorithms. Propositional logic-based learning is more interpretable, and the results are easy for humans to understand. Hence, the TM has important applications in the field of explainable AI, where interpretability and transparency are important for machine learning algorithms [6,7]. Propositional logic-based algorithms are also much faster in execution time than traditional neural networks that are based on multiple layers using vector multiplication. In fact, some traditional machine learning algorithms have proposed using propositional logic based algorithms for learning [8,9].

In the TM, propositional logic is used to create clauses. The clauses are made from literals which are features of the data. The literals within the clauses are learned using Type I or Type II feedback based on the feedback from the classification result. The predicted class is based on the majority sum from the clauses which is then obtained by a step-function defined by the parameter T. The number of clauses and the parameter T are the main hyperparameters that increase the discriminative power of the TM and enable patterns to be learned more efficiently.

Increasing the number of clauses and having a high value of T, may lead to better predictive power of the TM but it may also lead to overfitting and an increase in resources used and a reduction in training speed.

In this paper, we propose a novel variant of the TM called Regularized TM (RegTM). RegTM is based on the idea of using regularizers and smoothness functions, just like in traditional machine learning algorithms [10]. We use two different variants of regularizers. The first regularizer, called the Moving Average Regularizer (MAR), uses the sum of clauses from previous iterations and samples and incorporates them into the current clause sum calculation. The second type of regularizer, called the Weighted Regularizer (WER), adds the sum of the weights to improve generalization. By using

 doi:10.20944/preprints202409.1767.v

these regularization techniques, the generalization capability of the TM can improve, increasing their predictive power, especially in cases where the data is low in count.

Furthermore, we also propose the use of the Sigmoid function [11,12]. The sigmoid function can make the TM differentiable, and it can be further optimized with standard methods like gradient descent.

In summary, the contributions of our paper are as follows:

- We propose a novel variant of the Tsetlin Machine called Regularized Tsetlin Machine (RegTM) that implements regularization based on past clause sums and weights to improve the accuracy of the TM by increasing its generalizability, making it highly suitable for few-shot learning and meta-learning optimization tasks.
- We propose to use the Sigmoid function as an alternative to the unit-step function.
- We perform different experiments using benchmark image datasets to show the efficiency of RegTM with the different regularizers and sigmoid function.

2. Related Works

Granmo et al. proposed the TM in 2018 as a way to provide solutions to the bandit algorithm [1,13,14]. Since then, there has been a lot of significant work in this domain to improve upon the existing idea.

The TM was recast as a contextual bandit algorithm in [15]. Two contextual bandit learning algorithms were proposed: TM with ϵ-greedy arm selection and TM with Thompson sampling.

TM was extended for image classification and was called the Convolution Tsetlin Machine (CTM) [16,17]. CTM employed clauses as a convolution filter on each image as opposed to one clause for each image.

In addition to image classification, TM has also been used for regression [18]. The Regression TM is able to output in a continuous form rather than as distinct categories. The Regression TM employs a new voting mechanism to analyze complex patterns.

For multi-output problems, a coalesced Tsetlin Machine (CoTM) was proposed that simultaneously learns both the weights and the composition of each clause by employing interacting Stochastic Searching on the Line (SSL) and Tsetlin Automata (TA) [19].

TM has also shown great potential for natural language processing, e.g., text encoding [20], text classification [21–23], sentiment analysis [24], question classification [25], and fake news detection [26].

TM has also been implemented in the domain of healthcare. For example, a TM architecture was developed for the identification of premature ventricular contraction (PVC) by analyzing long-term ECG signals [27]. The capability of TM using continuous data was proposed for forecasting disease outbreaks [18]. Intelligent prefiltering of healthcare measurement data using Principal Component Analysis and Partial Least Squares Regression was added to TM to improve performance [28].

A label-critical TM was proposed that used a twin of label-critical Tsetlin Automata (TA). Label-TA learned the correct labels of the individual samples guided by a logical self-corrected TM clause [29].

There has also been substantial work on improving the feasibility of TM. One of the ways of improving the TM was by focusing on the clauses. The accuracy of the TM was improved with the addition of a drop clause probability [30]. The drop clause probability dropped clauses with a specific probability to increase the robustness, accuracy, and training time of TM, similar to a dropout layer in neural networks.

Another improvement to TM was the addition of weights called the Weighted Tsetlin Machine (WTM) [31]. The Weighted Tsetlin Machine (WTM) adds weights to the clauses to reduce the computation time and memory usage.

Focused negative sampling (FNS) was introduced to discriminate between classes that are not easily distinguishable from each other to improve TM precision [32].

A method to more efficiently prune clauses using the Markov boundary was proposed [33]. The new scheme aimed to add an additional feedback scheme to the TM, called the Type III feedback.

 doi:10.20944/preprints202409.1767.v

Another proposed scheme added the absorbing automata to the Tsetlin Machine [34]. Another variant of TM focusing on improving the efficiency of clauses was called the Clause Size Constrained TM (CSC-TM) [35]. In CSC-TM, a soft constraint on the clause size is placed and as soon as the clause includes more literals, that is, more than the allowed constraint, the literals are removed. Another research work looked at finding the optimal clause count by proposing a run-time pruning system [36].

3. Preliminary: Vanilla Tsetline Macnhine

The TM is able to solve complex pattern recognition problems using proposition-based conjunctive logic statements [1]. A standard Tsetlin Automaton (TA) consists of a collection of TAs [37] to find solutions to the multi-armed bandit problem [38,39] and the Finite State Learning Algorithm (FSLA) [40]. The TM uses conjunctive clauses composed of a collection of TAs.

In this section, we provide an overview of the TM to help us understand the algorithm introduced in subsequent sections. The TM takes a vector X of n propositional variables as input within the domain 0,1 and is defined as:

$$\mathbf{X} = x_k \in 0,1^n \tag{1}$$

For example, if n=2, the variables are x_1 and x_2, and the domain is (0,0),(0,1),(1,0),(1,1).

For example, if $n = 2$, the variables are x_1 and x_2, then the domain is $\{(0,0),(0,1),(1,0),(1,1)\}$.

The TM determines patterns using conjunctive clauses. A conjunctive clause consists of literals and is meant to describe the dataset. A literal consists of the variables and their negations $\mathbf{L} = l_k \in 0,1^{2n}$ where each literal l_k is defined as:

$$l_k = \begin{cases} x_k & if \quad 1 \leq k \leq n, \\ \neg x_{k-n} & if \quad n+1 \leq k \leq 2n \end{cases} \tag{2}$$

The first n literals are the actual variables and the next n variables are the negated variables. Using the example of $n = 2$, $L = \{x_1, x_2, \neg x_1, \neg x_2\}$.

Typically, the number of clauses is a user set parameter. It is an important parameter that determines how well training is done [36]. Let the number of clauses be defined as m and each clause be denoted as C_j where $j \in \{1, ..., m\}$, then the clause is a conjunction of a subset of literals $L_j \subseteq L$:

$$C_j(\mathbf{x}) = \bigwedge_{l_k \in L_j} l_k = \prod_{l_k \in L_j} l_k \tag{3}$$

$C_j \in \{0,1\}$. Let $m = 2$ and the user defined clauses are $C_1(\mathbf{x}) = x_1 \wedge \neg x_2$ and $C_2(\mathbf{x}) = x_1 \wedge x_2$. Then, from the previous example, the literals have the index $L_1 = \{1,4\}$ and $L_2 = \{1,2\}$.

Each clause is also assigned a polarity $p \in \{-1,+1\}$. The polarity decides whether the clause output is negative or positive. Positive polarity clauses, denoted by $C_j^+ \in \{0,1\}^{\frac{m}{2}}$, are assigned to class $y = 1$ and negative polarity clauses, denoted by $C_j^- \in \{0,1\}^{\frac{m}{2}}$, are assigned to class $y = 0$. Typically, half of the clauses are positive and half are negative. The positive polarity clauses vote for classifying the input as the target class and the negative polarity clauses vote against the target class.

The final classification decision is done by summing the clause outputs:

$$s(\mathbf{x}) = \sum_{j=1}^{m/2} C_j^+ - \sum_{j=1}^{m/2} C_j^- \tag{4}$$
$$\hat{y} = u(s(\mathbf{x}))$$

 doi:10.20944/preprints202409.1767.v

where $u(v)$ is the unit step function defined as:

$$u(v) = \begin{cases} 1 & if \quad v \geq 0 \\ 0 & otherwise \end{cases} \tag{5}$$

In the end, the output of the sum of summation decides the class $\hat{y} \in \{0, 1\}$.

Equation (4) was modified in the by adding weights to the clauses Weighted Tsetlin Machine [31]:

$$s(\mathbf{x}) = \sum_{j=1}^{m/2} w_j^+ C_j^+ - \sum_{j=1}^{m/2} w_j^- C_j^- \tag{6}$$

For example, Figure 1 shows two positive polarity clauses, $C_1(x) = x_1 \wedge \neg x_2$ and $C_3(x) = \neg x_1 \wedge \neg x_2$ (negative polarity clauses are not shown). Both clauses evaluate to zero, and leading to a prediction of $\hat{y} = 1$.

For training, each clause is given feedback with a probability $\hat{p}$.

$$\hat{p} = \begin{cases} \frac{T + clip(s(x), -T, T)}{2T} & if \quad y = 0 \\ \frac{T - clip(s(x), -T, T)}{2T} & if \quad y = 1 \end{cases} \tag{7}$$

In the above equation, $s(x)$ is the sum of the clause of Equation (6). T is a user-defined parameter. The $clamp$ function scales the $s(x)$ between $-T$ and T.

The above random clause selection is crucial as it guides the clauses to distribute across the sub-patterns in a more efficient manner. The frequency of each clause becomes proportional to $\pm T$. The amount of feedback depends on this and more feedback is only generated when $s(x)$ is far from T. In this way, only a few clauses are activated, thus reducing the overhead during training.

The TA is depicted in the upper part of Figure 1. The TA determines the actions to take during learning. The TA can decide to include or exclude the literal in the clause based on the received feedback. The feedback can be categorized as Type I or Type II. Figure 2 depicts a two-action TA with 2N states. For states $1 \ldots N$, the TA performs exclude (Action 1) and for states $N + 1 \ldots 2N$, include (Action 2) actions are performed. Based on the feedback, a TA can receive a reward or a penalty. If a TA receives a reward, then the literal moves deeper into being included or excluded. If a TA receives a penalty, then the literal moves towards the middle until it moves from action 1 to action 2 and vice versa.

Type I feedback reduces false negatives and overfitting by frequently recognizing patterns. Type II feedback suppresses false positives by increasing the discrimination power of the learned patterns.

Type I feedback: In Type I feedback, clauses of positive polarity receive a positive feedback when $y = 1$ and negative feedback for clauses with negative polarity and when $y = 0$. There are two types of Type I feedback: Type 1a and Type1b. Type 1a feedback reinforces the TA include actions. Type 1b feedback is activated when either the clause output or the literal is 0. Type 1b causes the TA to exclude the literal.

Type I feedback is defined by a user-defined parameter s where $s \geq 1$. Let the TA state be defined by $a_{j,k}$ for action on the k^{th} literal and the j^{th} clause. The TA state $a_{j,k}$ receives Type 1a feedback with probability $\frac{s-1}{s}$ and increases $a_{j,k}$ by 1. Consequently, the TA receives Type 1b feedback with probability $\frac{1}{s}$ and reduces $a_{j,k}$ by 1. In this way, Type 1a pushes to include the literal and Type 1b pushes to exclude the literal.

Type II feedback: Type II feedback is given to clauses with positive polarity when $y = 0$ and to clauses with negative polarity when $y = 1$. When the clause output is 0, the TA states are not updated. When the clause output is 1, all 0-valued literals that make the output 0 in TA state $a_{j,k}$ are increased by 1. Thus, this feedback focuses on all literals that differentiate between $y = 0$ and $y = 1$.

 doi:10.20944/preprints202409.1767.v

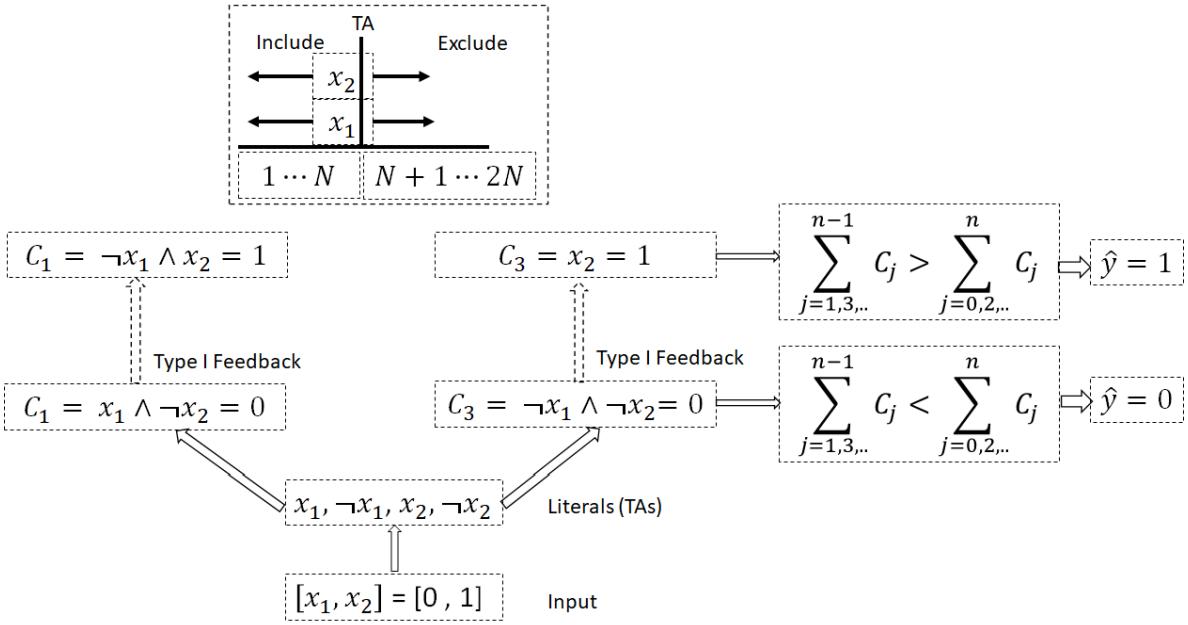

Figure 1. TM learning dynamics with input $(x_1 = 0, x_2 = 1)$ and output $y = 1$.

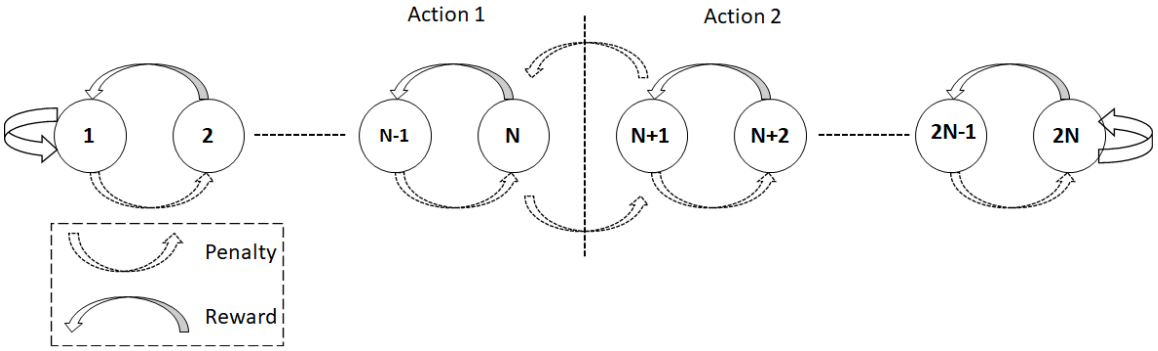

Figure 2. A two-action Tsetlin Automaton with 2N states.

4. Methodology

In this section, we introduce our novel TM variant titled the Regularized Tsetlin Machine (RegTM).

RegTM is based on the novel idea of using regularization and smoothing for Tsetline Machines. Regularization has been used extensively in machine learning algorithms. It is a way of infusing knowledge about data and parameters into the learning algorithm and provides a good balance between bias and variance. Regularization adds bias to the learning process and helps reduce the variance.

Regularization can reduce overfitting and improve the generalizability of the model [41].

One of the most common methods of regularization is to constrain the capacity of a model by adding some penalty function to the original objective function and making a new objective function. In TM we consider the objective function as the weighted clause sum equation represented in Equation (6). Hence, the new clause summation equation in RegTM can be written as:

$$s_{RegTM}(x) = \sum_{j=1}^{m/2} (w_j^+ C_j^+ + b_j^+) - \sum_{j=1}^{m/2} (w_j^- C_j^- - b_j^-) \tag{8}$$

where b_j^+ and b_j^- are the two regularization terms for the positive and negative classes.

 doi:10.20944/preprints202409.1767.v

Next we will describe the two regularization methods we propose for implementation in RegTM.

Moving Average Regularizer (MAR): The first regularization method we explore is inspired by the moving average term used in time series analysis [42]. In this case we assume that the the short-term fluctuations between clause sums can cause the model to be less generalizable. To remove the fluctuation, we propose a bias function that is dependent on the past values of the clause sums. We call this *moving average regularizer (MAR)*.

Consequently, b_j^+ and b_j^- from Equation (8) can be written as:

$$b_j^+ = p(\frac{1}{k}\sum_{n=1}^{k} \mathtt{C}_{j-n}^+) \tag{9}$$

$$b_j^- = p(\frac{1}{k}\sum_{n=1}^{k} \mathtt{C}_{j-n}^-) \tag{10}$$

where p is a weighting parameter to define how much weight we want to put on the regularizer terms, k is a parameter that determines how far into the past we want to consider the clause sums, and $\mathtt{C}_{j-k}^+$ and $\mathtt{C}_{j-k}^-$ are the positive and negative clause sums of the sample $(j-k)^{th}$:

$$\mathtt{C}_{j-k}^+ = \sum_{j=1}^{m/2} w_j^+ C_j^+ \tag{11}$$

$$\mathtt{C}_{j-k}^- = \sum_{j=1}^{m/2} w_j^- C_j^- \tag{12}$$

Weighted Regularizer (WER): The second method of regularization is based on weight decay [41], which we call *Weighted Regularizer (WER)*. For WER, the bias terms from Equation (8) are expressed as:

$$b_j^+ = p\sum_{j=1}^{m/2} (w_j^+)^k \tag{13}$$

$$b_j^- = p\sum_{j=1}^{m/2} (w_j^-)^k \tag{14}$$

where p is a parameter to define how much weight we want to put on the regularizer, k is the power of the weights, and $(w_j^+)^k$ and $(w_j^-)^k$ are weights from Equation (8) for the k^{th} sample.

Sigmoid: In this paper, we also introduce an alternative to Equation (7) for the feedback mechanism of the classes. One of the drawbacks of Equation (7) is that it is non-differentiable and cannot be optimized via standard methods such as gradient descent. This might be problematic if the TM is to be integrated with traditional neural networks for hyperparameter optimization and for providing explainability to neural networks.

Hence, we propose the use of the Sigmoid function. The Sigmoid function is a popular mathematical function that has an S-shaped curve and works in a similar way to the clamp function [9]. The sigmoid function maps values between $0, 1$. The sigmoid function is defined as:

$$\sigma(s(x)) = \frac{1}{1 + exp^{-s(x)}} \tag{15}$$

 doi:10.20944/preprints202409.1767.v

5. Experiments

In this section, we introduce the experimental parameters, data used for the experiments, and the experimental results.

5.1. Experimental Design

All of our experiments are performed using the Github repository of the TM [1].

We use the two standard benchmark datasets (MNIST and CIFAR10) for vision classification. The MNIST database contains 60,000 training images and 10,000 testing images. The dataset is a set of 28x28 grayscale images of 10 digits. The CIFAR10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The benchmark methods for comparison are the weighted Tsetlin Machine with literal and drop clauses [30,31]. The RegTM implementation includes using MAR and WER with and without the sigmoid function.

5.2. Experiment Results

The accuracy of the test dataset after 60 epochs is summarized in Tables 1 and 2. The column **TM** reports the results with vanilla weighted TM. The column **with drop probability** reports the results with drop clause and drop literal probabilities included in the algorithm. The results are reported for different values of clause counts (C).

The drop probabilities are obtained by using the literal and clause drop probability that results in the best accuracy. For MNIST, the best accuracy is obtained when both clause drop and literal drop probabilities are 0.1. For CIFAR10, the best accuracy is obtained when the clause drop probability and the literal drop probability are 0.4 and 0.0, respectively.

Table 1. Summary of results for MNIST Data (literal drop=0.1, clause drop=0.1).

Method (C=100, T=5000)	**TM**	**with drop probability**
No RegTM	96.43	96.54
Sigmoid	96.45	96.15
MAR only (k=0, p=0.03)	**96.75**	95.48
MAR + sigmoid (k=1, p=0.1)	96.68	96.42
WER (k=2, p=0.4)	96.59	96.23
WER + sigmoid (k=3, p=0.01)	96.63	96.23
Method (C=200, T=5000)	**TM**	**with drop probability**
No RegTM	97.28	97.08
Sigmoid	97.09	97.24
MAR only (k=0, p=0.03)	97.19	97.06
MAR + sigmoid (k=1, p=0.1)	**97.29**	97.21
WER (k=2, p=0.4)	97.10	97.17
WER + sigmoid (k=3, p=0.01)	97.24	97.04

[1] https://github.com/cair/tmu/tree/main

 doi:10.20944/preprints202409.1767.v

Table 2. Summary of results for CIFAR10 Data (literal drop=0.4, clause drop = 0.0).

Method (C=100, T=5000)	**TM**	**with drop probability**
No RegTM	32.75	33.72
Sigmoid	32.92	33.84
MAR only (k=0, p=0.8)	33.82	32.27
MAR + sigmoid (k=3, p=0.5)	34.02	33.26
WER (k=3, p=1.0)	33.75	33.65
WER + sigmoid (k=2, p=0.7)	**33.86**	33.01
Method (C=200, T=5000)	**TM**	**with drop probability**
No RegTM	34.37	35.63
Sigmoid	35.89	36.31
MAR only (k=0, p=0.8)	34.84	36.45
MAR + sigmoid (k=3, p=0.5)	34.73	34.15
WER (k=3, p=1.0)	36.21	35.57
WER + sigmoid (k=2, p=0.7)	36.54	**36.57**

Similarly, we also use the values for k and p that give the highest accuracy. For the MNIST dataset, when considering MAR, the best accuracy is obtained when k=0 and p=0.03. For MAR with sigmoid, the best accuracy is obtained when k=1 and p=0.1. Similarly, for WER the best accuracy is obtained when k=2 and p=0.4 and for WER with sigmoid, by setting k=3 and p=0.01 gives the best accuracy.

For the CIFAR10 dataset, the best accuracies are given by $k = 0, p = 0.8$ for MAR, $k = 3, p = 0.5$, for MAR and sigmoid, $k = 3, p = 0.1$ for WER, and $k = 2, p = 0.7$ for WER and sigmoid.

In general, using RegTM regularizers improves the accuracy of the TM. In addition, the improvement in accuracy by using RegTM is better when compared to using drop literals and drop clauses. In fact, for the MNIST dataset, adding drop probability appears to have reduced the accuracy. Furthermore, combining RegTM with drop clauses improves accuracy even further, especially for the CIFAR10 dataset where the accuracy is already low.

For RegTM with only the sigmoid function (denoted by sigmoid), the accuracy is very close to the vanilla TM for both the MNIST and CIFAR10 datasets. This means that the sigmoid function can be used as a viable replacement for Equation (7), especially when there is a need to develop a differentiable model.

When comparing different datasets, the improvement in accuracy for MNIST is much lower compared to CIFAR10. This is because MNIST already has a high accuracy, so improvements in its accuracy are very small.

The results show that RegTM with WER has higher accuracy than MAR, and when it does have lower accuracy (e.g., in the case of the MNIST dataset), the difference is very low. Adding a sigmoid function to MAR and WER also appears to not affect the performance of MAR and WER and in some cases improves the accuracy.

5.3. Ablation Studies

Our ablation studies include observing the effect of the parameter k and p on the accuracy and comparing the runtime of RegTM with TM and TM with drop probabilities.

The effect of the parameters k and p on the accuracy for both the MNIST and CIFAR10 datasets is shown in Figures 3–6. It is observed that for a specific k value and any value of p, the accuracy drops sharply. In the case of the MAR algorithm, significantly lower accuracy is obtained when $k = 2$ and for the WER algorithm, when $k = 0$ much lower accuracy is observed. We will study this in more detail in future work. Across the rest of the k and p values, there is very little variation in the accuracy values. For MNIST the accuracy is almost the same. For CIFAR10, the variation is slightly more which may be due to the complexity of the dataset.

 doi:10.20944/preprints202409.1767.v

We also compare the runtime over 60 epochs for the different regularizers. The run-time of RegTM with MAR and WER is comparable to that of the vanilla TM and better than the use of dropping clauses and literals. Adding the sigmoid function does increase the run-time compared to vanilla TM and TM with drop clauses, but the increase is not so significant.

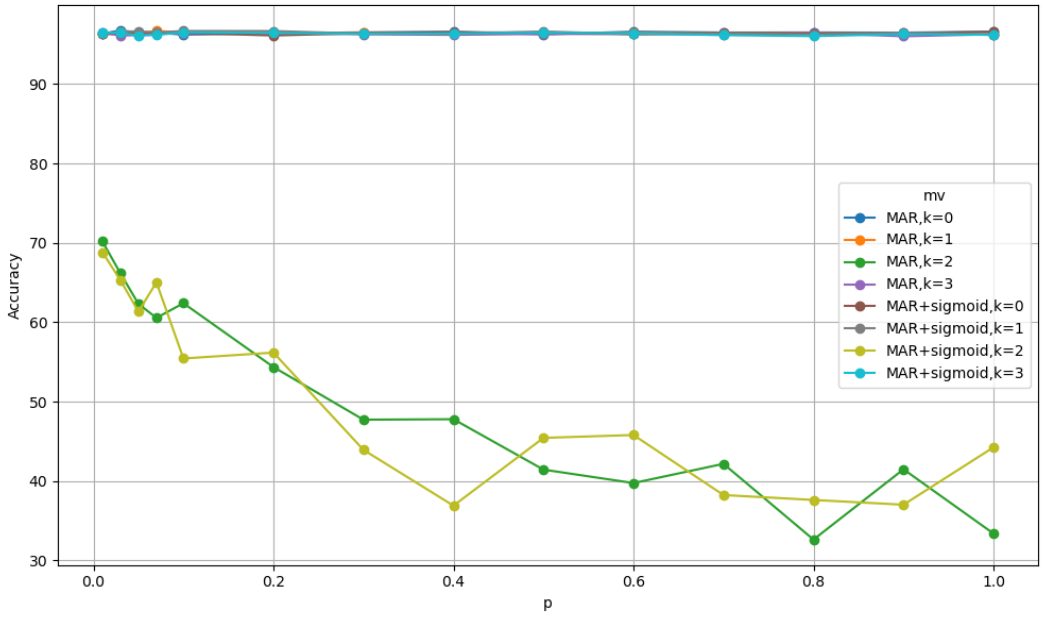

Figure 3. MNIST MAR algorithm (C=100, T=5000)

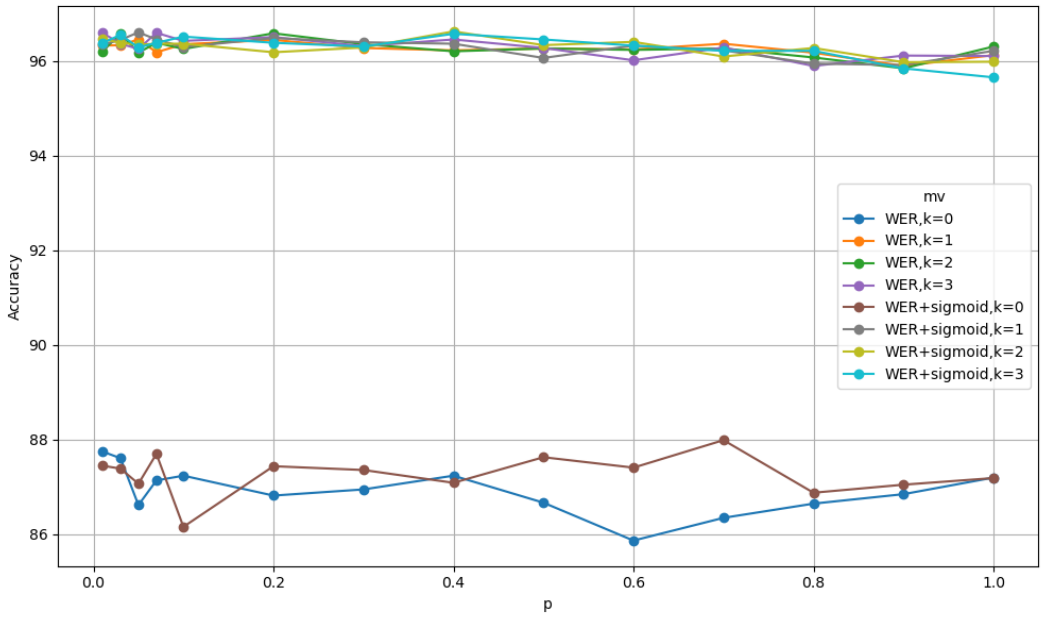

Figure 4. MNIST WER algorithm (C=100, T=5000)

 doi:10.20944/preprints202409.1767.v

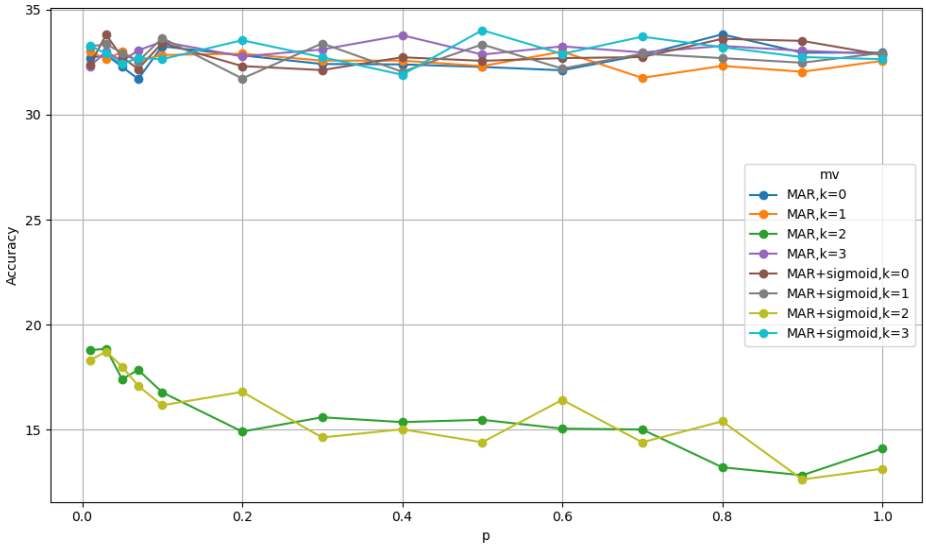

Figure 5. CIFAR10 MAR algorithm (C=100, T=5000)

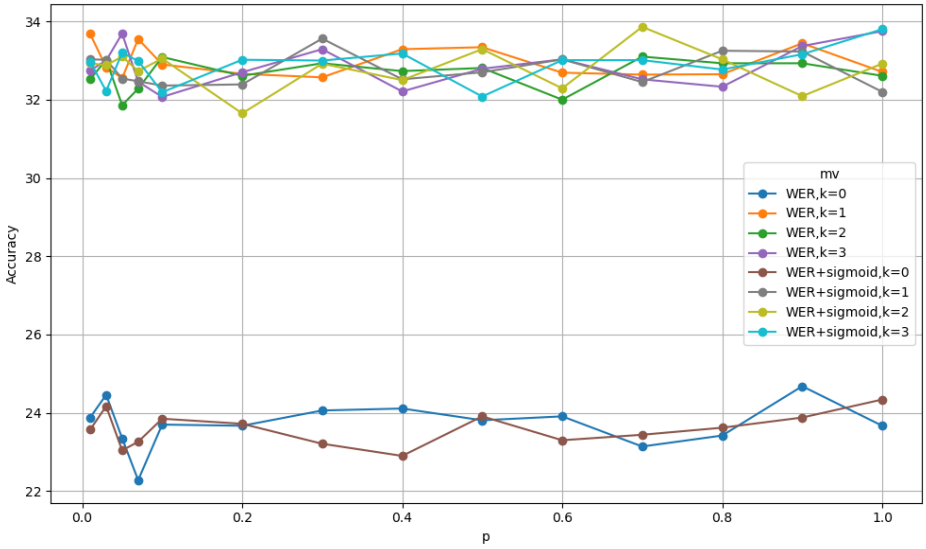

Figure 6. CIFAR10 WER algorithm (C=100, T=5000)

Table 3. Runtime Comparison (60-epochs, C=100, T=5000).

Method	MNIST	CIFAR10
Vanilla TM	00:08:10	00:21:10
Vanilla TM with drop clause	00:08:37	00:22:07
sigmoid only	00:08:18	00:22:12
MAR only	00:08:14	00:21:36
MAR + sigmoid	00:08:18	00:22:10
WER	00:08:10	00:21:25
WER + sigmoid	00:08:23	00:23:41

6. Conclusion

In this paper, we introduce a novel variant of the Tsetlin Machine called the Regularized Tsetlin Machine (RegTM) by proposing the idea of using regularizers to improve the performance and generalization of the TM. Even though the improvement in the results is marginal, it opens up new

 doi:10.20944/preprints202409.1767.v

potential for future research, where generalization is important, especially in the field of few-shot learning and meta-optimization [43,44].

Our work shows promise and improves upon the performance of TM. However, there are some limitations, epsecally in case of TM. In some cases, the explanatory advantages that TM provides are not intuitive. Understanding and making the interpretations provided by TMs are important. But this is an on-going research. In our paper, we do not specifically focus on the interpretability of TM but rather on increasing its generalizability through regularizers.

Another limitation of TM is its inefficiency in handling large-scale data. This is because of the increase in the Tsetlin Automata, which can cause memory usage problems and reduce computational efficiency. One way of dealing with this issue is by using Sparse Tsetlin Machine [45] and parallelization [1]. However, it is still an active area of research that still requires further attention. In our paper, we do not focus on how to handle large-scale data. However, we believe regularizers could be used in combination with Sparse Tsetlin Machines to improve performance.

Another concern regarding TM is how to select the different hyperparameters. Currently, the way to select hyperparameters is using grid search, random search, cross-validation or using Bayesian optimization [1,46]. We have shown that using the sigmoid function does not reduce the performance of TM. This means that gradient descent could be used in combination with TM, allowing us to apply the same optimization techniques as those from traditional neural networks. Hence, we can implement other hyperparameter optimization techniques in TM that are also used in neural networks, which would be an interesting area for future research.

Another limitation of TM is that it can only handle binary features. This means that continuous features can only be handled by discretization, which may lead to information loss. This has been discussed in [47] but it still requires further research and investigation of techniques that can reduce information loss.

The main focus of TM has been in the field of classification and regression tasks, but its application in other tasks (clustering, segmentation, and sequence modeling) is still limited. Research on the application of TM in other tasks is ongoing. Our work on regularizers aims to be a step towards the application of TM in other domains and tasks, more specifically in few-shot learning. The applicability of TM for other tasks would be another potential future research area.

Another interesting area of study can involve investigating more complex regularizers, such as tangentprop [48]. However, implementing such regularizers and even the sigmoid function might be challenging considering that TMs are mainly designed for low-energy hardware. This is something that could also be further studied.

References

1. Granmo, O.C. The Tsetlin Machine–A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic. *arXiv preprint arXiv:1804.01508* **2018**.
2. Abeyrathna, K.D.; Bhattarai, B.; Goodwin, M.; Gorji, S.R.; Granmo, O.C.; Jiao, L.; Saha, R.; Yadav, R.K. Massively parallel and asynchronous tsetlin machine architecture supporting almost constant-time scaling. International Conference on Machine Learning. PMLR, 2021, pp. 10–20.
3. Lei, J.; Wheeldon, A.; Shafik, R.; Yakovlev, A.; Granmo, O.C. From arithmetic to logic based ai: A comparative analysis of neural networks and tsetlin machine. 2020 27th IEEE international conference on electronics, circuits and systems (ICECS). IEEE, 2020, pp. 1–4.
4. Cheng, R.; Vasudevan, D.; Kirst, C. Super-Tsetlin: Superconducting Tsetlin Machines. *IEEE Transactions on Applied Superconductivity* **2024**.
5. Morris, J.; Rafiev, A.; Xia, F.; Shafik, R.; Yakovlev, A.; Brown, A. An alternate feedback mechanism for tsetlin machines on parallel architectures. 2022 International Symposium on the Tsetlin Machine (ISTM). IEEE, 2022, pp. 53–56.
6. Xu, F.; Uszkoreit, H.; Du, Y.; Fan, W.; Zhao, D.; Zhu, J. Explainable AI: A brief survey on history, research areas, approaches and challenges. Natural Language Processing and Chinese Computing: 8th CCF International

Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8. Springer, 2019, pp. 563–574.
7. Došilović, F.K.; Brčić, M.; Hlupić, N. Explainable artificial intelligence: A survey. 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE, 2018, pp. 0210–0215.
8. Anjum, U.; Zadorozhny, V.; Krishnamurthy, P. Localization of Unidentified Events with Raw Microblogging Data. *Online Social Networks and Media* **2022**, *29*, 100209.
9. Petersen, F.; Borgelt, C.; Kuehne, H.; Deussen, O. Deep differentiable logic gate networks. *Advances in Neural Information Processing Systems* **2022**, *35*, 2006–2018.
10. Tian, Y.; Zhang, Y. A comprehensive survey on regularization strategies in machine learning. *Information Fusion* **2022**, *80*, 146–166.
11. Dombi, J.; Jónás, T. The generalized sigmoid function and its connection with logical operators. *International Journal of Approximate Reasoning* **2022**, *143*, 121–138.
12. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing* **2022**, *503*, 92–108.
13. Bouneffouf, D. Multi-armed Bandit Problem and Application **2023**.
14. Bouneffouf, D.; Rish, I.; Aggarwal, C. Survey on applications of multi-armed and contextual bandits. 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2020, pp. 1–8.
15. Seraj, R.; Sharma, J.; Granmo, O.C. Tsetlin Machine for Solving Contextual Bandit Problems. *Advances in Neural Information Processing Systems* **2022**, *35*, 30194–30205.
16. Tunheim, S.A.; Jiao, L.; Shafik, R.; Yakovlev, A.; Granmo, O.C. Convolutional Tsetlin Machine-based Training and Inference Accelerator for 2-D Pattern Classification. *Microprocessors and Microsystems* **2023**, *103*, 104949.
17. Granmo, O.C.; Glimsdal, S.; Jiao, L.; Goodwin, M.; Omlin, C.W.; Berge, G.T. The convolutional Tsetlin machine. *arXiv preprint arXiv:1905.09688* **2019**.
18. Darshana Abeyrathna, K.; Granmo, O.C.; Zhang, X.; Jiao, L.; Goodwin, M. The regression Tsetlin machine: a novel approach to interpretable nonlinear regression. *Philosophical Transactions of the Royal Society A* **2020**, *378*, 20190165.
19. Glimsdal, S.; Granmo, O.C. Coalesced multi-output tsetlin machines with clause sharing. *arXiv preprint arXiv:2108.07594* **2021**.
20. Bhattarai, B.; Granmo, O.C.; Jiao, L.; Yadav, R.; Sharma, J. Tsetlin Machine Embedding: Representing Words Using Logical Expressions. *arXiv preprint arXiv:2301.00709* **2023**.
21. Saha, R.; Granmo, O.C.; Zadorozhny, V.I.; Goodwin, M. A relational tsetlin machine with applications to natural language understanding. *Journal of Intelligent Information Systems* **2022**, pp. 1–28.
22. Saha, R.; Granmo, O.C.; Goodwin, M. Using Tsetlin machine to discover interpretable rules in natural language processing applications. *Expert Systems* **2023**, *40*, e12873.
23. Berge, G.T.; Granmo, O.C.; Tveit, T.O.; Goodwin, M.; Jiao, L.; Matheussen, B.V. Using the Tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications. *IEEE Access* **2019**, *7*, 115134–115146.
24. Yadav, R.K.; Jiao, L.; Granmo, O.C.; Goodwin, M. Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis. The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21). AAAI, 2021.
25. Nicolae, D.C. Question Classification using Interpretable Tsetlin Machine. The 1st International Workshop on Machine Reasoning (MRC 2021), 2021.
26. Bhattarai, B.; Granmo, O.C.; Jiao, L. Explainable Tsetlin Machine Framework for Fake News Detection with Credibility Score Assessment. Proceedings of the Language Resources and Evaluation Conference; European Language Resources Association: Marseille, France, 2022; pp. 4894–4903.
27. Zhang, J.; Zhang, X.; Jiao, L.; Granmo, O.C.; Qian, Y.; Pan, F. Interpretable Tsetlin Machine-based Premature Ventricular Contraction Identification. *arXiv preprint arXiv:2301.10181* **2023**.
28. Jenul, A.; Bhattarai, B.; Liland, K.H.; Jiao, L.; Schrunner, S.; Futsaether, C.; Granmo, O.C.; Tomic, O. Component Based Pre-filtering of Noisy Data for Improved Tsetlin Machine Modelling. 2022 International Symposium on the Tsetlin Machine (ISTM). IEEE, 2022, pp. 57–64.
29. Abouzeid, A.; Granmo, O.C.; Goodwin, M.; Webersik, C. Label-Critic Tsetlin Machine: A Novel Self-supervised Learning Scheme for Interpretable Clustering. 2022 International Symposium on the Tsetlin Machine (ISTM). IEEE, 2022, pp. 41–48.

30. Sharma, J.; Yadav, R.; Granmo, O.C.; Jiao, L. Drop clause: enhancing performance, robustness and pattern recognition capabilities of the Tsetlin machine. Proceedings of the AAAI Conference on Artificial Intelligence, 2023, Vol. 37, pp. 13547–13555.
31. Phoulady, A.; Granmo, O.C.; Gorji, S.R.; Phoulady, H.A. The weighted tsetlin machine: compressed representations with weighted clauses. *arXiv preprint arXiv:1911.12607* **2019**.
32. Glimsdal, S.; Saha, R.; Bhattarai, B.; Giri, C.; Sharma, J.; Tunheim, S.A.; Yadav, R.K. Focused Negative Sampling for Increased Discriminative Power in Tsetlin Machines. 2022 International Symposium on the Tsetlin Machine (ISTM). IEEE, 2022, pp. 73–80.
33. Granmo, O.C.; Andersen, P.A.; Jiao, L.; Zhang, X.; Blakely, C.; Berge, G.T.; Tveit, T. Learning Minimalistic Tsetlin Machine Clauses with Markov Boundary-Guided Pruning. 2023 International Symposium on the Tsetlin Machine (ISTM). IEEE, 2023, pp. 1–8.
34. Bhattarai, B.; Granmo, O.C.; Jiao, L.; Andersen, P.A.; Tunheim, S.A.; Shafik, R.; Yakovlev, A. Contracting Tsetlin Machine with Absorbing Automata. *arXiv preprint arXiv:2310.11481* **2023**.
35. Abeyrathna, K.D.; Abouzeid, A.A.O.; Bhattarai, B.; Giri, C.; Glimsdal, S.; Granmo, O.C.; Jiao, L.; Saha, R.; Sharma, J.; Tunheim, S.A.; others. Building concise logical patterns by constraining tsetlin machine clause size. *arXiv preprint arXiv:2301.08190* **2023**.
36. Rahman, T.; Maheshwari, S.; Shafik, R.; Yakovlev, A.; Das, S. MILEAGE: An Automated Optimal Clause Search Paradigm for Tsetlin Machines. 2022 International Symposium on the Tsetlin Machine (ISTM), 2022, pp. 49–52. doi:10.1109/ISTM54910.2022.00017.
37. Tsetlin, M.L. On behaviour of finite automata in random medium. *Avtomat. i Telemekh* **1961**, *22*, 1345–1354.
38. Robbins, H. Some aspects of the sequential design of experiments **1952**.
39. Gittins, J.C. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **1979**, *41*, 148–164.
40. Narendra, K.S.; Thathachar, M.A. *Learning automata: an introduction*; Courier corporation, 2012.
41. Moradi, R.; Berangi, R.; Minaei, B. A survey of regularization strategies for deep models. *Artificial Intelligence Review* **2020**, *53*, 3947–3986.
42. Hamilton, J.D. *Time series analysis*; Princeton university press, 2020.
43. Wang, Y.; Yao, Q.; Kwok, J.T.; Ni, L.M. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* **2020**, *53*, 1–34.
44. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. International conference on machine learning. PMLR, 2017, pp. 1126–1135.
45. Østby, S.; Brambo, T.M.; Glimsdal, S. The Sparse Tsetlin Machine: Sparse Representation with Active Literals. *arXiv preprint arXiv:2405.02375* **2024**.
46. Abeyrathna, K.D.; Granmo, O.C.; Zhang, X.; Goodwin, M. A scheme for continuous input to the Tsetlin machine with applications to forecasting disease outbreaks. Advances and Trends in Artificial Intelligence. From Theory to Practice: 32nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2019, Graz, Austria, July 9–11, 2019, Proceedings 32. Springer, 2019, pp. 564–578.
47. Mathisen, E.; Smørvik, H.S. Analysis of binarization techniques and Tsetlin machine architectures targeting image classification. Master's thesis, University of Agder, 2020.
48. Simard, P.; Victorri, B.; LeCun, Y.; Denker, J. Tangent prop-a formalism for specifying selected invariances in an adaptive network. *Advances in neural information processing systems* **1991**, *4*.