Drop Clause: Enhancing Performance, Robustness and Pattern Recognition Capabilities of the Tsetlin Machine

Jivitesh Sharma, Rohan Yadav, Ole-Christoffer Granmo, Lei Jiao

Center for Artificial Intelligence Research (CAIR)
University of Agder, Norway
{jivitesh.sharma, rohan.k.yadav, ole.granmo, lei.jiao}@uia.no

Abstract

Logic-based machine learning has the crucial advantage of transparency. However, despite significant recent progress, further research is needed to close the accuracy gap between logic-based architectures and deep neural network ones. This paper introduces a novel variant of the Tsetlin machine (TM) that randomly drops clauses, the logical learning element of TMs. In effect, TM with Drop Clause ignores a random selection of the clauses in each epoch, selected according to a predefined probability. In this way, the TM learning phase becomes more diverse. To explore the effects that Drop Clause has on accuracy, training time and robustness, we conduct extensive experiments on nine benchmark datasets in natural language processing (IMDb, R8, R52, MR, and TREC) and image classification (MNIST, Fashion MNIST, CIFAR-10, and CIFAR-100). Our proposed model outperforms baseline machine learning algorithms by a wide margin and achieves competitive performance compared with recent deep learning models, such as BERT-Large and AlexNet-DFA. In brief, we observe up to 10% increase in accuracy and $2\times$ to $4\times$ faster in learning than those of the standard TM. We visualize the patterns learnt by Drop Clause TM in the form of heatmaps and show evidence of the ability of drop clause to learn more unique and discriminative patterns. We finally evaluate how Drop Clause affects learning robustness by introducing corruptions and alterations in the image/language test data, which exposes increased learning robustness.

Introduction

Researchers across various fields are increasingly paying attention to the interpretability of AI techniques. While interpretability previously was inherent in most machine learning approaches, the state-of-the-art methods now increasingly rely on black-box deep neural network-based models. Natively, these can neither be interpreted during the learning stage nor while producing outputs. For this reason, a surge of techniques attempts to open the black box by visual explanations and gradient-based interpretability (Simonyan, Vedaldi, and Zisserman 2014; Zhang, Wu, and Zhu 2018; Radhakrishnan et al. 2018; Bau et al. 2017; Selvaraju et al. 2017; Ribeiro, Singh, and Guestrin 2016).

The Tsetlin Machine (TM) is an interpretable rule-based machine learning algorithm that produces logical rules (Granmo 2018). In brief, it employs an ensemble of Tsetlin Automata (TA) that learns propositional logic expressions from Boolean input features, and TM achieves interpretability by leveraging sparse disjunctive normal form. Indeed, humans are particularly good at understanding flat and short logical AND-rules, reflecting human reasoning (Noveck et al. 1991). In addition, due to its Boolean representations and finite-state learning mechanisms, it possesses a minimalistic memory footprint.

Similar to neural networks, the vanilla TM suffers overfitting in the sense that its learning elements, i.e., clauses, are prone to capture noisy and redundant patterns. Initially, we believed that this could be simply because of the large number of clauses. However, by decreasing the number of clauses, there was a proportionate decrease in performance. In order to mitigate overfitting, inspired by Dropout (Srivastava et al. 2014), we propose a method named *Drop Clause* (DC) in this paper[1]. More specifically, DC method randomly *drops* or switches off a set of clauses during training, which is similar to what dropout does in neural networks. Unlike dropout, DC induces stochasticity in TM learning, boosting performance in terms of accuracy and learning speed. It further improves the patterns captured by the clauses, making them more robust towards input perturbations. As a direct consequence of DC during training, the training time is reduced proportionally and the heatmaps produced becomes more distinct.

Paper Contributions:

- We propose to drop clauses randomly during each training iteration, which introduces additional stochasticity.
- We demonstrate that DC makes the TM capture more unique patterns, improving its generalization performance.
- DC TM leads to competitive results to large deep neural network models on nine benchmark datasets for NLP and image classification. Since TM is more akin to a standard machine learning algorithm, like a decision tree, than a deep learning model, we compare DC TM with well-known machine learning algorithms, documenting superior performance across all the datasets.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]The code is available online on: https://github.com/Anonymous-2491/Drop-Clause-Interpretable-TM.

- Via heatmaps, we visualize that DC TM becomes more clear-cut and robust on image classification and language sentiment analysis.

Drop Clause for Tsetlin Machines

Modern society needs reliable, unbiased, and trustworthy AI systems. Researchers have shown that neural networks are not fully mature for integration into society. For example, neural network interpretability is fragile towards adversarial examples (Ghorbani, Abid, and Zou 2019). Generating adversarial perturbations that produce visually indistinguishable images for humans leads to dramatically different neural network interpretations, without changing the label. While methods such as GradCAM (Selvaraju et al. 2017) and LIME (Ribeiro, Singh, and Guestrin 2016) can take a peek inside black-box models, the explanations offered can be arbitrarily inaccurate, as argued by Rudin et al. They pinpoint several reasons for choosing inherently interpretable models over attempting to explain black-box ones (Rudin 2019). They further raise three algorithmic challenges that the machine learning community faces to succeed with human-level interpretability. Although neural networks are inherently inadequate for these challenges, the TM addresses them natively:

- *Challenge 1: Rule-based logical models*. The TM learns a rule-based model using logical operations. Learning is game-theoretic with Nash equilibria that correspond to the optimal propositional logic expressions (Granmo 2018; Zhang et al. 2022; Jiao et al. 2022).
- *Challenge 2: Linear models with sparse scoring systems.* The TM is a linear model that produces sparse clauses and integer weighted scores (Abeyrathna, Granmo, and Goodwin 2021).
- *Challenge 3: Domain-specific interpretable AI.* The TM is inherently interpretable, used for producing interpretable models across several domains (Abeyrathna et al. 2020; Lei et al. 2021; Yadav et al. 2021, 2022).

In what follows, we first briefly present the basic TM and its convolutional version. Thereafter, we introduce the DC technique, which enhances the stochasticity of TM learning.

TM and Convolutional TM

A TM in its simplest form takes a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_o] \in \{0,1\}^o$ of o Boolean values as input, producing a Boolean output $\hat{y} \in \{0,1\}$. Patterns are expressed as conjunctive clauses C_j (AND-rules), built from literals $L = \{l_1, l_2, \ldots, l_{2o}\} = \{x_1, x_2, \ldots, x_o, \neg x_1, \neg x_2, \ldots, \neg x_o\}$:

$$C_j = \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k \right]. \tag{1}$$

In Eq. (1), a_k^j is the TA state that controls inclusion of literal l_k in clause j and $g(\cdot)$ maps the TA state to action 0 or 1. The imply operator, in turn, implements the action in the clause. That is, if $g(a_k^j)$ is 1, then the imply operator requires that the literal is 1 for the whole expression to be 1. If $g(a_k^j)$ is 0, on the other hand, the expression becomes 1 regardless the true value of l_k.

$$\hat{y} = \begin{cases} 1, & \sum_{j=1,3,\ldots}^{n-1} C_j - \sum_{j=2,4,\ldots}^{n} C_j \geq 0, \\ 0, & \text{Otherwise.} \end{cases} \tag{2}$$

Clauses are divided into even-indexed/odd-indexed clauses that vote for output $\hat{y} = \{0,1\}$. Odd-indexed clauses vote in favour of the class whereas even-indexed clauses vote against the class, as shown in Eq. (2). A positive vote count means majority of the clauses predict the input to belong to the class and a negative vote count implies majority of the clauses believe the input does not belong to the class. A detailed explanation can be found in (Granmo 2018).

The Convolutional TM (CTM) is as an interpretable alternative to 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. The output of a convolution clause is obtained simply by ORing the outcome of evaluating the clause on each patch:

$$\hat{y} = \begin{cases} 1, & \sum_{j=1,3,\ldots}^{n-1} \bigvee_{b=1}^{B} C_j - \sum_{j=2,4,\ldots}^{n} \bigvee_{b=1}^{B} C_j \geq 0, \\ 0, & \text{Otherwise.} \end{cases}$$

Here, b refers to one out of B available image patches. A detailed explanation can be found in (Granmo et al. 2019). Please see the supplementary material for further details on TM and CTM.

Drop Clause

Here we propose a novel regularization method for the TM, i.e., DC. This technique is inspired by the dropout method for neural networks. In DC, clauses are removed with a probability p in each training epoch:

$$\hat{y} = \begin{cases} 1, & \sum_{j=1,3,\ldots}^{n-1} \pi_j C_j - \sum_{j=2,4,\ldots}^{n} \pi_j C_j \geq 0, \\ 0, & \text{Otherwise.} \end{cases} \tag{3}$$

Above, $\pi_j \in \{0,1\}$ for clause j is zero with probability p for each complete epoch. Clearly, the vanilla TM and its DC variant are equivalent if $p = 0$.

DC TM is similar to dropout in neural networks, and the purpose is to reduce the chance of learning redundant patterns. A vanilla TM seeks to minimize prediction error on the training data. Achieving this, there may still be unused patterns available in the training data. These patterns can potentially be useful when facing new data, such as test data. By randomly dropping clauses for complete epochs, we mobilize other clauses to take over the role of the dropped clauses. DC can be thought of as having a pool of clauses and selecting a set of clauses with $1 - p$ probability for training every epoch. However, due to this stochastic learning, the mobilized clauses may solve the task in a different way each time. Hence, robustness increases overall when the complete set of clauses are turned on again. The resulting effects are evaluated experimentally in the next section.

The stochasticity induced by DC in the TM learning process

is similar to the stochasticity induced by stochastic gradient descent (SGD) (Robbins 2007). In SGD, a mini-batch of training samples is selected randomly from the training data. While, in DC, various sets of clauses are selected for training with probability $1-p$. The major difference is that, SGD is stochastic in terms of data sampling during the learning stage, whereas DC is stochastic in terms of sampling the elements of the model structure (clauses are the elements that form the TM model). Another difference is that the stochasticity of DC can be adjusted with the DC probability hyperparameter p. Note that the Drop Clause TM still retains its convergence properties. Please see supplementary material for details.

Enhanced Performance

We here investigate the effects that DC has on the performance of TMs. To assess the generality of our approach, we test DC both on NLP and image classification. For NLP, we use TM with weighted clauses, and for image classification, we use the CTM. For evaluation, we compare our approach with the comparable state-of-the-art deep learning models. Since TM is a rule-based machine learning method, we also cover more standard machine learning algorithms. The DC implementation is written in PyCUDA for parallel GPU computation. We train our models on 16 NVIDIA Tesla V100 TensorCore GPUs for the image classification task and on one NVIDIA RTX 3070 8GB GPU for the NLP tasks[2].

Natural Language Processing

We explore the performance of DC on NLP tasks first. The first experiments assess how varying the DC probability affects classification accuracy and then compare the best found configuration with similar machine learning algorithms and state-of-the-art techniques. We adopt 5 popular standard datasets, summarized below:

- ***IMDb*** consists of $50,000$ movie reviews for sentiment analysis, split in half for training and testing. Here, we use $10,000$ clauses and set TM threshold T to $8,000$ and specificity s to 2.0.
- ***Reuters-21578*** contains the text categorization datasets ***R8*** and ***R52***. R8 is divided into 8 categories, with $5,485$ training and $2,189$ testing samples, whereas R52 consists of 52 categories, divided into $6,532$ training and $2,568$ testing samples. We here employ $3,000$ clauses with $T = 2,000$ and $s = 7.0$.
- ***MR*** is another movie review dataset for binary sentiment classification, consisting of $10,662$ samples. We use the train-test split as in (Tang, Qu, and Mei 2015) and use $5,000$ clauses with $T = 4,000$ and $s = 6.0$.
- ***TREC*** is a question classification dataset that encompasses 6 categories. There are a total of $6,000$ samples, $5,500$ for training and 500 for testing. Here, we employ $5,000$ clauses with $T = 4,000$ and $s = 2.0$.

[2]Note that these are not hardware requirements. We use these GPUs just to train faster.

TM	$p = 0$	$p = .1$	$p = .25$	$p = .5$	$p = .75$
IMDB	87.2	88.3	89.6	90.4	**91.27**
R8	96.16	97.6	98.1	98.5	**98.94**
R52	89.14	89.5	90.8	91.5	**92.75**
MR	75.14	77.25	77.9	78.2	**78.67**
TREC	88.05	89.8	90.1	**90.5**	89.9

Table 1: Effect of DC on NLP Datasets.

CTM	$p = 0$	$p = .1$	$p = .25$	$p = .5$	$p = .75$
MNIST	99.3	99.3	**99.45**	99.35	98.2
F-MNIST	91.5	91.75	**92.5**	92.25	91.25
CIFAR-10	69.3	70.5	73.2	**75.1**	72.6
CIFAR-100	35.5	39.5	42.6	**45.2**	40.8

Table 2: Effect of DC on Image Classification.

Our first step is to compare the difference in performance with respect to changing DC probability p, as shown in Table 1. The selected DC probabilities are $p \in \{0.1, 0.25, 0.5, 0.75\}$. For IMDb, R8, R52 and MR, the best performance is achieved with $p = 0.75$, which is equivalent of dropping 75% of the total clauses per class, per epoch. As for TREC, $p = 0.5$ works best. As seen, DC has significant effect on accuracy, with average accuracy going up by 4.07% for IMDb, 2.78% for R8, 3.61% for R52, 3.53% for MR and 2.45% for TREC. Additionally, we observe a substantial reduction in training time proportional to the DC ratio. Inference times on these datasets are less than $2.3s$ per 1000 samples on average on an NVIDIA GTX 1080 GPU.

The TM is a rule-based machine learning algorithm, so we also compare our proposed TM model with a few traditional machine learning techniques. Specifically, we compare our proposed TM with DC against Support Vector Machine (SVM), Random Forests (RF), K-Nearest Neighbours (K-NN) and XGBoost (XGB). Table 2 displays the comparison results[3]. From Table 2, it is clear that DC TM outperforms other machine learning algorithms by a wide margin.

We also compare our model with deep learning methods, some of which representing the state of the art. Tables 4 and 5 show the results of the comparisons. Enhanced with DC, the accuracy of TM is comparable not only to CNNs and LSTMs but also to computationally complex and parametrically large state-of-the-art models like BERT-Large. On all datasets in Table 4, inclusion of DC propels the performance of the TM to outperform CNN (Kim 2014) and bidirectional LSTM with pretrained word embeddings. In fact, on R8 and R52, DC TM is able to achieve better accuracy than BERT (Devlin et al. 2019) and comes close to its performance on MR. Also, comparing our model with the state-of-the-art graph convolutional neural network, S^2GC (Zhu and Koniusz 2020), DC TM achieves better performance on R8 and MR, and comparable performance

[3]Note that the machine learning methods used here for comparison have default parameters from sklearn.

	TM	TM (DC)	SVM	RF	K-NN	XGB
IMDB	87.2	91.27	83.2	78.5	72.5	84.4
R8	96.16	98.94	84.7	82.4	77.3	85.1
R52	89.14	92.75	71.0	72.5	58.2	75.4
MR	75.14	78.67	52.2	51.7	39.1	56.3
TREC	88.05	90.5	67.5	65.4	56.1	77.6

Table 3: Comparison with ML methods.

	TM (DC)	CNN	BiLSTM	BERT LARGE	S^2GC
IMDB	91.27	87.9	88.9	95.4	-
R8	98.94	95.71	96.31	96.02	97.4
R52	92.75	87.59	90.54	89.66	94.5
MR	78.67	77.75	77.68	79.24	76.7

Table 4: Comparison on IMDb, R8, R52 and MR.

on R52. Similar results are obtained on the TREC dataset shown in Table 5. DC TM outperforms a baseline LSTM model as well as a vanilla Transformer (TF) and Transformer with feature projection (FP) (Qin, Hu, and Liu 2020).

Even though BERT-Large achieves better performance on TREC dataset, TM has major advantages over these models in terms of computational complexity and transparency, which are further enhanced by DC (See Section "Enhanced Robustness"). While it is disputed whether attention is explainable (Jain and Wallace 2019), attention is much more complex (quadratic complexity with floating-point operations) than our proposed model (linear complexity with Boolean operations) that achieves comparable performance. Note that our model only relies on simple bag-of-words tokens in the target datasets, without considering any additional pretrained world knowledge (Yadav et al. 2021), like word2vec, Glove or BERT features. Yet, it achieves competitive performance compared to deep learning and some state-of-the-art models.

Image Classification

We now turn to image classification, again focusing on how DC affects performance of TM. To this end, we evaluate DC on four benchmark image classification datasets: MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100. We binarize the datasets using an adaptive Gaussian thresholding procedure as proposed in (Granmo et al. 2019). This binarization results in images with only 1 bit per pixel per channel, considerably reducing the memory overhead. For the task of image classification, we employ the CTM.

- ***MNIST*** encompasses $70,000$ 28×28 gray-scale images of hand written single digits, $60,000$ for training and $10,000$ for testing. Here, we use $8,000$ clauses with $T = 6,400$ and $s = 5.0$.

	TM (DC)	LSTM	TF	FP+TF	BERT-LARGE
TREC-6	90.5	87.19	87.33	89.5	95.6

Table 5: Comparison on TREC-6.

- ***Fashion-MNIST*** contains 28×28 gray-scale images from the Zalando catalogue. For this dataset, we use the same parameters as the MNIST dataset.
- ***CIFAR-10 and CIFAR-100*** consist of $50,000$ 32×32 color images of objects divided into 10 categories for CIFAR-10 and 100 for CIFAR-100. Another $10,000$ images are provided for testing. Here, we make use of $60,000$ clauses with $T = 48,000$ and $s = 10.0$.

Again, we explore the effects of four DC probability settings, $p \in \{0.1, 0.25, 0.5, 0.75\}$, shown in Table 2. The best performance is achieved with $p = 0.25$ for the MNIST and Fashion-MNIST, while $p = 0.5$ gives the best result for the CIFAR datasets, which is equivalent to dropping a quarter (or half) of the clauses per training iteration. Notice the considerable performance increase, especially on the CIFAR datasets, with DC of $p = 0.5$. Peak accuracy on MNIST is 99.45% $(\pm 0.25\%)$, and on Fashion-MNIST it is 92.5% $(\pm 0.25\%)$. CIFAR-10 and CIFAR-100 peak at 75.1% $(\pm 0.4\%)$ and 45.2% $(\pm 0.2\%)$ respectively, averaged over 100 runs. Apart from the accuracy gain, DC with $p = 0.5$ reduces training time by approximately 50%, for the CIFAR datasets. For instance, training time per epoch drops from $61.83s$ to $32.58s$ for CIFAR-10. The inference time for the DC CTM on CIFAR-10 is $1.5s$ per 1000 images on one NVIDIA Tesla V100 GPU, leveraging the computation benefits of binary operations for TM inference.

Table 6 shows the comparison of the DC CTM with other popular machine learning techniques on the image classification. As seen, CTM outperforms the traditional machine learning techniques by a wide margin, further widened by the introduction of DC[4].

Table 7 contains a comparison between the DC CTM with certain related and state-of-the-art techniques. Binary neural networks can be represented exactly as a propositional logic expression (Narodytska et al. 2018). As such, they are particularly comparable to CTM and we therefore also include results for EEV-BNN (Jia and Rinard 2020). EEV-BNN is a binary neural network that has been verified with Boolean satisfiability. Employing EEV-BNN, we use the MNIST-MLP and Conv-Large BNN architectures from (Jia and Rinard 2020) for comparisons. The DC CTM significantly outperforms EEV-BNN on MNIST and CIFAR-10, as shown in Table 7. We also compare our method with a transformer based model of reduced complexity to make it more comparable to our model. The Vision Nystromformer (ViN) (Jeevan and Sethi 2021) is a vision transformer (ViT) based model that reduces the quadratic computational complexity of ViT by using the Nystrom methods for approximat-

[4]Note that the accuracy on CIFAR-100 is not displayed in Table 6 for other machine learning algorithms as the accuracy is minuscule.

	CTM	CTM (DC)	SVM	RF	K-NN	XGB
MNIST	99.3	99.45	93.4	93.8	95.3	96.2
F-MNIST	91.5	92.5	84.6	87.5	85.4	88.4
CIFAR-10	69.3	75.1	37.5	48.7	33.9	47.8
CIFAR-100	35.5	45.2	-	-	-	-

Table 6: Image Classification with ML methods.

	CTM (DC)	EEV -BNN	HYBRID -VIN	ALEXNET -DFA	MAXOUT
MNIST	99.45	98.25	96.4	98.2	99.06
F-MNIST	92.5	-	-	91.66	-
CIFAR-10	75.1	63.45	75.26	64.7	86.8
CIFAR-100	45.2	-	-	52.62	59.52

Table 7: Image Classification with SOTA models.

ing self-attention. It achieves 65.06% accuracy on CIFAR-10 whereas the Hybrid-ViN, which incorporates rotary positional embedding, obtains slightly better accuracy as shown in Table 7, comparable to our model.

We also contrast our model against AlexNet with Direct Feedback Alignment (DFA) (Webster, Choi, and changwook Ahn 2021) for parallel backpropagation training, which makes it comparable to the fast training of TM. The results in Table 7 show that AlexNet-DFA achieves similar performance on MNIST and Fashion-MNIST, whereas it is outperformed by DC CTM on CIFAR-10. However, there is a considerable difference on the CIFAR-100 dataset due to the larger label space. Finally, as DC is inspired from dropout, we compare our model with the Maxout network (Goodfellow et al. 2013), which was a natural companion to dropout. Maxout achieves significantly better performance on the CIFAR datasets as shown in Table 7. Note that on color images, binarization can lead to loss of important information especially when we use 1-bit per channel, compared with 8-bits per channel for lossless color information. Also, as previously stated, TM possesses computational advantages over these methods along with transparency.

Another point to note is that, originally, dropout can increase the performance of basic CNNs by about 3% and 6% on CIFAR-10 and CIFAR-100, respectively (Srivastava et al. 2014). The DC method shows promising results by improving the performance of vanilla CTM by about 6% on CIFAR-10 and 10% on CIFAR-100.

Enhanced Pattern Recognition

In this section, we observe more distinct heatmaps produced by TM and CTM for NLP and image classification tasks as a result of DC, thereby visually exemplifying its superior pattern recognition capabilities. We show enhancements in the quality of heatmaps on the MR dataset at the word-level and on the CIFAR-10 dataset at the pixel-level. The enhancements encompass richer and more distinct word representations for NLP tasks and more clear-cut pixel representations for objects in images. More specifically, these enhancements indicate that DC is able to capture more unique, relevant and discriminative patterns. These enhancements further manifest the stronger pattern recognition capabilities of DC[5].

Natural Language Sentiment Analysis

We investigate local word-level patterns captured by TM and DC TM in NLP using a randomly selected test example from MR: "A waste of fearless purity in the acting craft". The example was selected among the ones that correctly classified with DC but incorrectly classified without it. The purpose is to contrast and exemplify how DC captures better patterns and enhances the heatmaps. Note that for NLP, the majority of the features appear in negated form. In order to have word-level understanding of the model, we use a frequency-based interpretation, i.e., we highlight the features based on how frequently they appear in clauses (Yadav et al. 2021). An arguably unique property of TMs is that the patterns they produce are both descriptive (frequent) and discriminative. In other words, each clause captures a full description of the target concept, not merely the discrimination boundary.

Figure 1a highlights the top 100 most frequent features in negated form (darker color means high frequency), presenting in the clauses triggered by the given sample using vanilla TM without DC. Three example clauses C_0, C_2, and C_4 are shown below for the color-coded features. In this case, the prediction is wrong. More importantly, the literals in the conjunctions do not make sense.

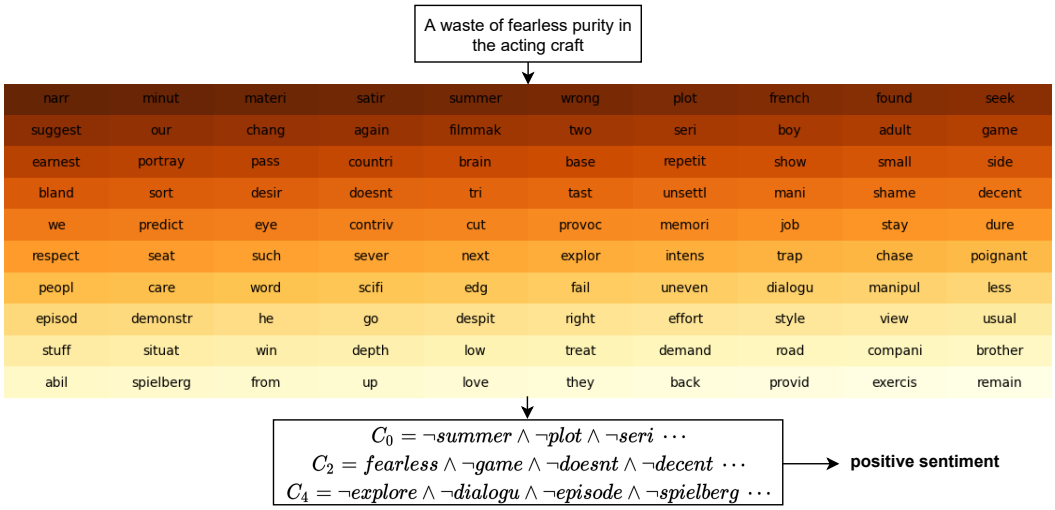
The corresponding result for DC TM is depicted in Figure 1b. Here, the 100 most frequent features in the negated form seem intuitive for predicting negative sentiment. Features such as "NOT witti (witty)", "NOT grace (graceful)", "NOT terrif (terrific)", "NOT honest", "NOT cool", or "NOT intellig (intelligent)" generally mean absence of positive sentiment, which in this case makes the model draw the correct conclusion (negative sentiment). Some of the specific patterns that are responsible for predicting the correct output are C_0, C_2, and C_4, as shown in the figure. Although randomly chosen, this example is representative for how TM with DC is able to capture a larger variety of correct patterns than what the vanilla TM is capable of.

Image Classification

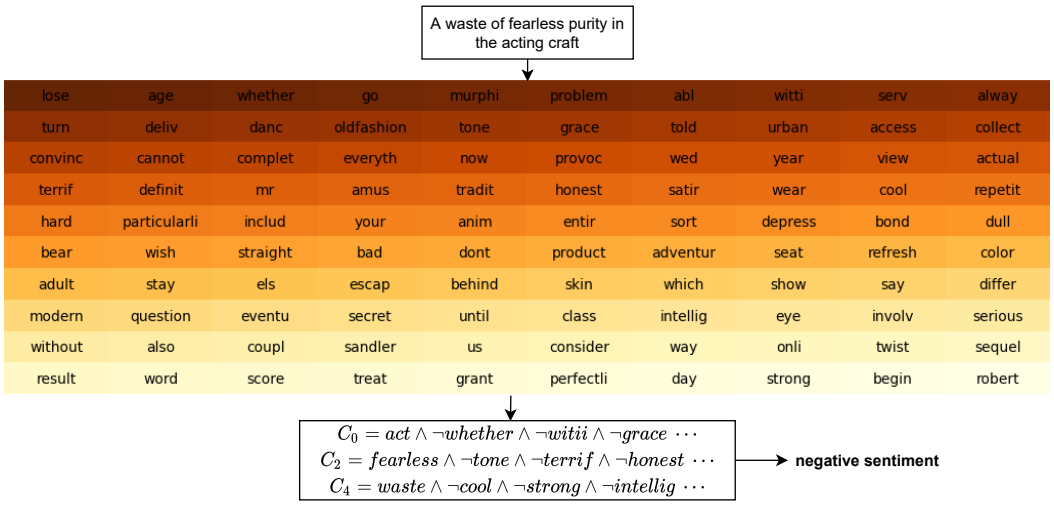
In image classification, TM clauses form self-contained patterns by joining pixels into multi-pixel structures. That is, the image pixels are sent directly to the clauses, following propositional AND-rules. Therefore, Boolean propositional expressions (clauses) capture patterns in an image that contains pixels as literals in the clauses, which are comparatively easy for humans to comprehend (Valiant 1984).

The CTM forms clauses from the pixels of the square image patches obtained in the convolution. Accordingly, we extract the top k weighted clauses per class in patch form. For visualization, we represent the non-negated pixels of a clause as 1, negated pixels as -1, and excluded pixels as 0. In additional to the image content, each clause also encodes positions in the image where it is valid. If a certain position is

[5]More examples of enhanced pattern recognition and interpretability can be found in the supplementary material.



(a) Without drop clause.

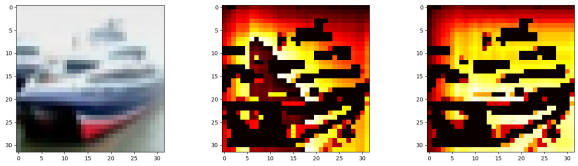


(b) With drop clause ($p = 0.75$).

Figure 1: Word-level patterns of a sample captured: (a) without and (b) with DC.

invalid, the clause literals corresponding to the pixel values are treated as 0 to indicate no activation at that location. The resulting clause masks are applied to the image and their activation maps are added up to produce a heatmap.

In Figure 2, we show the advantage of DC TM via an example. As can be seen from the figure, DC is able to capture the object with more precision by capturing more discriminative patterns and including pixels relevant to classification. We believe this is because the remaining clauses are forced to substitute the dropped clauses, learning to perform their tasks more independently. Due to the stochastic nature of learning, they will, however, learn to perform the tasks differently than the dropped clauses. As a result, DC reduces redundancy and induces diversity in the learning of the patterns. An example of how these heatmaps can be interpreted is shown in the supplementary material.



(a) Image (b) TM ($p = 0$) (c) TM ($p = 0.5$)

Figure 2: CIFAR-10 patterns captured: (b) without and (c) with DC.

Enhanced Robustness

In this section, we show empirically that introduction of DC makes TM more robust towards input perturbations during testing. We evaluate robustness on both sentiment analysis

	TM	TM (DC)	BiLSTM
MR	75.14	78.67	77.68
MR (perturb)	73.12	77.12	75.85

Table 8: Robustness Test on MR.

	CTM	CTM (DC)	BNN
MNIST	99.3	99.45	98.25
MNIST-C	88.12	91.75	85.5

Table 9: Robustness Test on MNIST-C.

and image classification[6].

Natural Language Sentiment Analysis

We select the MR dataset to test the robustness of TM on sentiment analysis. We train our models on the standard training data and perturb the testing data to evaluate robustness towards out of distribution changes. We monitor the decrease in testing accuracy with and without perturbations. During testing, an input instance (sentence) is selected for perturbation with probability 0.5. Then, a word is chosen from this sentence randomly with uniform probability. The selected word is swapped with the most similar word according to the cosine similarity between their respective Glove embedding vectors. Swapping with the most similar word results in changing the binary feature representation by a hamming distance of 2, while keeping the meaning and sentiment of the sentence unchanged. The following is an MR example demonstrating this strategy: "*Magnificent* drama well worth watching" $\rightarrow$ "*Splendid* drama well worth watching". The word *Magnificent* is changed to its synonym *Splendid*. Table 8 shows the comparison between TM and DC TM on MR with and without test data perturbations. The difference between the test accuracy on the dataset with and without perturbations is 2.02% and 1.45% for the standard TM and DC TM respectively. In addition, DC makes TM more robust compared with BiLSTM, where the accuracy drops by 1.83%. The increase in TM robustness could be due to the fact that since clauses are stochastically dropped, they seem to be less prone to pick up irrelevant or noisy patterns.

Image Classification

We compare the robustness of CTM and DC CTM using the MNIST-C dataset (Mu and Gilmer 2019), which is a *corrupted* version of MNIST. It consists of 15 corruptions such as Gaussian blur, scale, rotate, shear, impulse noise, canny edges, fog etc. We train our models on the standard MNIST training data without any augmentations. During testing, we select whether to use non-corrupted or corrupted test image with probability 0.5. We then select one of the 15 corruptions to apply on the test image with uniform probability. DC enhances the robustness of CTM, as can be seen from Table 9. There is a drop of 11.18% for the standard CTM whereas the drop in accuracy is reduced to 7.7% with DC. Also, from Table 9, we observe that both the TM versions are more robust than binarized neural networks (BNN)[7], whose accuracy drops by 12.75%.

Discussion

To summarize our results, DC improves the performance of the TM, enhancing the advantages that TMs have over neural networks and traditional machine learning techniques when it comes to computational complexity, memory and energy consumption, training and inference time and robustness (Abeyrathna et al. 2021; Lei et al. 2020). The computational advantages make TMs suitable for federated learning and deployment on edge devices. On the other hand, the TM still achieves lower accuracy on the datasets we use here when compared with certain deep neural network models. However, TM is a machine learning method that achieves overall good performance while being inherently interpretable unlike deep neural networks that are black-box models. For certain NLP tasks, pretrained models, such as word2vec, GloVe, BERT, and GPT, have been dominating, leveraging unlabelled data. TMs require Boolean input, making it difficult to leverage existing high dimensional embeddings, impeding performance, with the exception of enhancing the input with GloVe-derived synonyms (Yadav et al. 2021). However, there is huge difference in model sizes, where SOTA deep neural networks have parameters upwards of 10 million floating-point parameters, easily going beyond 100 million (BERT-Large has 340 million parameters). On the other hand, TM employs a few thousand Boolean clauses. Also, the pixel-level interpretability on image classification can be difficult for humans to interpret as the Boolean expressions for clauses capturing patterns can be quite long, containing numerous pixels, especially for large images. Nevertheless, there are several $O(n)$ techniques that can easily simplify and reduce Boolean expressions (Meurer et al. 2017).

Conclusions

In this paper, we propose *drop clause* as a technique to improve the generalization ability of the TM. Drop clause enhances stochasticity during training, to capture more diverse patterns. As a result, the performance of TM is significantly boosted, which is empirically shown on a varied collection of datasets, both in terms of accuracy and training time. We further show how drop clause improves the pattern recognition capabilities of TM, simultaneously improving the clarity of the produced patterns. When comparing our model with the state-of-the-art deep learning models, we observe competitive accuracy levels. Since TM is more akin to traditional machine learning techniques, we also compare it with a selection of those, reporting superior performance of the TM. We finally establish that drop clause improves the robustness of the TM towards data corruptions and perturbations during testing.

[6]Note that we do not test for adversarial robustness. We only test the robustness of our models with corrupted input.

[7]The MNIST-MLP architecture from (Jia and Rinard 2020) is used here.

References

Abeyrathna, K. D.; Bhattarai, B.; Goodwin, M.; Gorji, S. R.; Granmo, O.-C.; Jiao, L.; Saha, R.; and Yadav, R. K. 2021. Massively Parallel and Asynchronous Tsetlin Machine Architecture Supporting Almost Constant-Time Scaling. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 10–20. PMLR.

Abeyrathna, K. D.; Granmo, O.-C.; and Goodwin, M. 2021. Extending the Tsetlin Machine With Integer-Weighted Clauses for Increased Interpretability. *IEEE Access*, 9.

Abeyrathna, K. D.; Pussewalage, H. S. G.; Ranasinghea, S. N.; Oleshchuk, V. A.; and Granmo, O.-C. 2020. Intrusion Detection with Interpretable Rules Generated Using the Tsetlin Machine. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE.

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network Dissection: Quantifying Interpretability of Deep Visual Representations. *CoRR*, abs/1704.05796.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Ghorbani, A.; Abid, A.; and Zou, J. 2019. Interpretation of Neural Networks Is Fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 3681–3688.

Goodfellow, I.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013. Maxout Networks. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 1319–1327. Atlanta, Georgia, USA: PMLR.

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

Granmo, O.-C.; Glimsdal, S.; Jiao, L.; Goodwin, M.; Omlin, C. W.; and Berge, G. T. 2019. The Convolutional Tsetlin Machine. *arXiv preprint arXiv:1905.09688*.

Jain, S.; and Wallace, B. C. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 3543–3556. Minneapolis, Minnesota: Association for Computational Linguistics.

Jeevan, P.; and Sethi, A. 2021. Vision Xformers: Efficient Attention for Image Classification. arXiv:2107.02239.

Jia, K.; and Rinard, M. 2020. Efficient Exact Verification of Binarized Neural Networks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1782–1795. Curran Associates, Inc.

Jiao, L.; Zhang, X.; Granmo, O.-C.; and Abeyrathna, K. D. 2022. On the Convergence of Tsetlin Machines for the XOR Operator. *IEEE Trans. Pattern Anal. Mach. Intell.*

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Doha, Qatar: Association for Computational Linguistics.

Lei, J.; Rahman, T.; Shafik, R.; Wheeldon, A.; Yakovlev, A.; Granmo, O.-C.; Kawsar, F.; and Mathur, A. 2021. Low-Power Audio Keyword Spotting Using Tsetlin Machines. *Journal of Low Power Electronics and Applications*, 11.

Lei, J.; Wheeldon, A.; Shafik, R.; Yakovlev, A.; and 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)*, 1–4.

Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, v.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3: e103.

Mu, N.; and Gilmer, J. 2019. MNIST-C: A Robustness Benchmark for Computer Vision. *CoRR*, abs/1906.02337.

Narodytska, N.; Kasiviswanathan, S.; Ryzhyk, L.; Sagiv, M.; and Walsh, T. 2018. Verifying properties of binarized deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Noveck, I.; Lea, R. B.; Davidson, G. M.; and O'brien, D. 1991. HUMAN REASONING IS BOTH LOGICAL AND PRAGMATIC. *Intellectica*, 11: 81–109.

Qin, Q.; Hu, W.; and Liu, B. 2020. Feature Projection for Improved Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8161–8171. Online: Association for Computational Linguistics.

Radhakrishnan, A.; Durham, C.; Soylemezoglu, A.; and Uhler, C. 2018. Patchnet: Interpretable Neural Networks for Image Classification. arXiv:1705.08078.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 1135–1144. New York, NY, USA: Association for Computing Machinery. ISBN 9781450342322.

Robbins, H. 2007. A Stochastic Approximation Method. *Annals of Mathematical Statistics*, 22: 400–407.

Rudin, C. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.

In *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–626.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958.

Tang, J.; Qu, M.; and Mei, Q. 2015. PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks. *CoRR*, abs/1508.00200.

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

Webster, M. B.; Choi, J.; and changwook Ahn. 2021. Learning the Connections in Direct Feedback Alignment. In *International Conference on Learning Representations*.

Yadav, R. K.; Jiao, L.; Granmo, O.-C.; and Goodwin, M. 2021. Enhancing Interpretable Clauses Semantically using Pretrained Word Representation. In *BLACKBOXNLP*.

Yadav, R. K.; Jiao, L.; Granmo, O.-C.; and Goodwin, M. 2021. Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*. AAAI.

Yadav, R. K.; Jiao, L.; Granmo, O.-C.; and Goodwin, M. 2022. Robust Interpretable Text Classification against Spurious Correlations Using AND-rules with Negation. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Zhang, Q.; Wu, Y. N.; and Zhu, S.-C. 2018. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8827–8836.

Zhang, X.; Jiao, L.; Granmo, O.-C.; and Goodwin, M. 2022. On the Convergence of Tsetlin Machines for the IDENTITY- and NOT Operators. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10): 6345–6359.

Zhu, H.; and Koniusz, P. 2020. Simple spectral graph convolution. In *International Conference on Learning Representations*.