AdaTM: Logic Inspired Adaptive Tsetlin Machines for Efficient and Effective Continual Learning on the Edge

Chong Tang
University of Southampton
Southampton, United Kingdom
chong.tang@soton.ac.uk

Neelam Singh
University of Southampton
Southampton, United Kingdom
n.singh@soton.ac.uk

Jagmohan Chauhan
University of Southampton
Southampton, United Kingdom
j.chauhan@soton.ac.uk

ABSTRACT

Continuous learning, crucial for applications with sequentially arriving data, enables dynamically acquiring new skills without forgetting previous knowledge. The deep learning community has devised various strategies to address the challenge of 'catastrophic forgetting', successfully achieving long-term knowledge retention. Nonetheless, the intricate floating-point computations inherent in neural models result in considerable computational demands and energy usage, making these approaches unsustainable. This motivates us to explore the untapped potential of an emerging logic-inspired alternative, the Tsetlin Machine (TM), which has demonstrated state-of-the-art performance in various applications. We develop an Adaptive Tsetlin Machine (AdaTM) - The first end-to-end continual learning solution solely relying on propositional logic operations, suited for edge computing devices. ***AdaTM*** is constructed through dynamically expanding model architecture to accommodate new learning tasks. Furthermore, we implemented a class-balance memory buffer and optimal states selection techniques to combat knowledge fading and introduced a clause confidence score-based pruning strategy for scalability. Importantly, ***AdaTM***'s adaptability is accommodated without the need for computationally expensive recalibrations commonly associated with neural networks leading to high-efficiency gains. The adaptability and efficiency set the ***AdaTM*** apart, making it particularly well-suited for real-world applications where resources are constrained, such as edge devices and for on-device learning. Extensive empirical evaluations across multiple datasets and computational environments such as Raspberry Pi have been performed on key metrics such as average accuracy, forgetting measure, processing latency, and run-time memory. Specifically, ***AdaTM*** achieves up to **25%** higher accuracy while using approximately **35x** less memory and ensuring competitive run time latency. Overall, our ***AdaTM*** pioneers how to equip logic-based models with long-term knowledge preservation without relying on complex architectures and computations of conventional neural network-based continual learning.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning approaches**; • **Hardware** → *Power estimation and optimization.*

KEYWORDS

Continual Learning, Tsetlin Machine, Efficiency, On Device Learning, Edge Computing

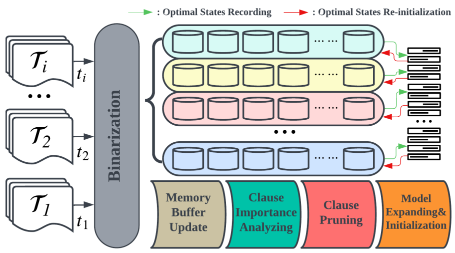

: Optimal States Recording
: Optimal States Re-initialization
$\mathcal{T}_i$
t_i
$\mathcal{T}_2$
t_2
$\mathcal{T}_1$
t_1
Binarization
Memory Buffer Update
Clause Importance Analyzing
Clause Pruning
Model Expanding& Initialization

Figure 1: Overview of the AdaTM architecture.

1 INTRODUCTION

Continual learning is essential for machine intelligence, striving for human-like adaptability by accumulating knowledge from diverse experiences. In this approach, models constantly adapt to new tasks and data, preserving prior knowledge. This adaptability is vital in applications ranging from robotics to natural language processing and is especially crucial in real-time response settings. Take a home assistive robot with on-edge continual learning as an example. Such a robot can personalize services through ongoing adaptation to user interactions and preferences, markedly enhancing user experience and operational efficiency. However, challenges like "catastrophic forgetting" arise, where models lose proficiency in old tasks when learning new ones[11, 19]. This problem is intensified by the class-incremental nature, leading to the potential need for retraining, further stressing computational and energy resources. These constraints become more pronounced in quick decision-making scenarios or resource-restricted settings like edge computing[2].

Numerous algorithms and techniques target effective continual learning. Methods like Elastic Weight Consolidation (EWC)[12] and Synaptic Intelligence (SI)[15] prevent forgetting through parameter regularization, while replay techniques like Experience Replay (ER)[4] and Deep Generative Replay (DGR)[25] use stored or generated examples to preserve past knowledge. iCaRL[23], a template-based method, utilizes class mean features to address forgetting. Despite these advances, each approach has its shortcomings. Many cannot rival the performance of joint training, and struggle with introducing new class categories. Neural network-based replay methods also heighten computational and energy demands. These challenges, although sometimes overshadowed by accuracy metrics, become pivotal in applications demanding on-device efficiency like edge AI. In this case, the Tsetlin Machine (TM) offers a unique avenue for exploration and is designed to bring a balanced approach to performance and computational efficiency. However, developing effective continual learning solutions for the Tsetlin Machine framework presents a notable challenge due to the distinct logic-based structure of TMs. This incompatibility means existing techniques for neural networks are not directly applicable, marking a clear divergence. This mismatch has not been extensively explored in prior research, representing a substantial challenge which is crucial for advancing TMs in computationally efficient applications.

Derived from automata theory, TM employs state-based decision-making and simple yet effective learning mechanisms[8]. It uses conjunctive clauses for feature representation and reinforcement learning for automata to refine decisions based on their quality. This design enables TM to tackle intricate pattern recognition tasks with fewer computational demands than neural networks. Furthermore, TM has shown competitive results on several benchmark inference tasks[3, 9, 22], underscoring its potential in the continual learning space. However, implementing continual learning with TM presents unique challenges. Key concerns include: **1).** How can we dynamically expand the architecture to accommodate the evolving nature of continual learning with progressively arriving tasks and classes? **2).** How might we balance the computational simplicity intrinsic to TM with consistent performance in a continual learning environment? **3).** Moreover, as the model grows, how do we safeguard the efficiency advantages that set TM apart? **4).** Most importantly, how can these concepts be effectively implemented in this novel and emerging model structure? Addressing these technical and implementation challenges, we propose ***AdaTM***. Unlike the intricate recalibrations in neural networks, ***AdaTM*** features dynamic model expansion: it spawns new TMs tailored for one-vs-all, budget-fixed, class-balanced replay as new classes appear. This adaptability is maintained without the computational and energy strains common to conventional methods, courtesy of efficient logic operations. We also developed a pruning method to optimize ***AdaTM***'s efficiency further.

In our rigorous tests, ***AdaTM*** consistently surpasses neural network-based continual learning algorithms in average accuracy, forgetting measure, processing speed, and run-time memory on both a laptop and a Raspberry Pi Model 4B. The main contributions of our study include– **(1) AdaTM Design for Continual Learning:** ***AdaTM*** is our pioneering exploration of TM architecture designed for continual learning. This design ensures dynamic adaptability to new data, efficiently learning incrementally and integrating knowledge. ***AdaTM*** stands out in resource-constrained scenarios due to its inherent simplicity and computational efficiency. A key feature is its ability to automatically generate new TMs for each new class, emphasizing seamless and recalibration-free learning progression; **(2) Efficient Pruning Mechanism within AdaTM:** Addressing potential inefficiencies in model growth, we have developed a clause confidence score-based pruning mechanism. This ensures the removal of non-essential clauses, thus maintaining a lean and efficient model. This approach further upholds the preservation of decision quality and operational efficiency, especially vital for applications in resource-limited settings; **(3) Robust Empirical Validation of AdaTM: *AdaTM***'s capabilities were rigorously tested across multiple datasets and environments. Our findings revealed its notable performance, with up to **25%** improved accuracy compared to leading neural network-based methods, while using significantly less memory. ***AdaTM***'s processing speed and consistent performance, even on platforms like the Raspberry Pi Model 4B, emphasize its potential as a front-runner in the continual learning domain.

2 RELATED WORK

Achieving flexible, human-like continual learning is still an open question. Van et al.[27] have classified continual learning into three primary scenarios: task-incremental, domain-incremental, and class-incremental learning. With this foundational understanding, we delve into an exploration of the prevailing methodologies in the subsequent sections. Specifically, given our paper's emphasis, our discussion will be centred on continual learning and Tsetlin Machines.

2.1 Continual Learning Strategies

For developing effective continual learning, various strategies have emerged to counteract catastrophic forgetting. They largely encompass regularization mechanisms, replay methods, and template-based solutions.[27].

2.1.1 Regularization. Elastic Weight Consolidation (EWC)[12], inspired by synaptic consolidation theories, employs a quadratic penalty based on the Fisher information matrix to preserve essential weight changes, thus retaining prior knowledge. Although EWC excels with MNIST and Atari games, it has constraints like depending on a Laplace approximation. Learning without Forgetting (LwF)[15] utilizes knowledge distillation to maintain prior task performance without old data, outperforming methods like Less Forgetting Learning[10]. Still, it occasionally shows minor past-task performance declines and requires task labels. LwF typically provides better new-task performance and adaptability than EWC. Synaptic Intelligence (SI)[29], an EWC derivative, optimizes computational efficiency by continuously assessing weight significance. Despite its success in tests like split MNIST, its reliance on certain SGD approximations and hyperparameters necessitates further comprehensive evaluations.

2.1.2 Template-based. Although regularization techniques mitigate catastrophic forgetting, they often fall short in class-incremental contexts[27], giving rise to template-based methods. Van et al.[26] transitioned from discriminative to generative classifiers, converting the class-incremental issue into a task-incremental one. Using distinct Variational Autoencoders (VAE) for each class, they sidestep data storage or replay, outdoing other rehearsal-free approaches. Yet, the model's scalability is questionable due to separate generative models per class, and occasionally, VAEs produce inferior samples with resource-heavy inference. Earlier, Rebuffi et al.[23] presented iCaRL, which conserves data efficiently using fewer exemplars. Its blend of distillation loss and nearest-mean classifiers bolsters its strength. Still, it sees escalating computational demands with more classes and doesn't quite match full batch training performance.

2.1.3 Replay. While template-based methods introduce unique approaches to continual learning, they grapple with computational and scalability issues. Replay methods offer an alternative, using 'replayed' memory samples during new task learning. A standout is Chaudhry et al.'s experience replay[5], which notably outperforms methods like GEM[16] and A-GEM[4]. Impressively, significant gains result even from a single example-per-class, thanks to implicit regularization. Shin et al.'s DGR[25] sidesteps data storage but hinges on its generative model quality, making it less suitable for quick sample generation. GDumb[20] adopts a straightforward two-step approach, utilizing a balanced class distribution and retraining networks from the beginning. Surprisingly, its performance surpasses intricate methods across benchmarks, highlighting the strength of balanced memory replay and questioning traditional continual learning perspectives.

While the above efforts address catastrophic forgetting, literature often points to the simplicity of replay methods as equally or more effective than complex techniques. This simplicity is not just appealing but also pragmatic. As we move towards resource-conscious environments, harnessing this simplicity grows in importance.

2.2 Efficient Continual Learning

Continual learning on edge devices requires a delicate balance of adaptability, performance, and efficiency. With the rise of on-device applications, model efficiency is paramount. Miro et al.[17] proposed a dynamic approach for edge devices using a hierarchical episodic memory, targeting both energy efficiency and accuracy. Yet, deep learning models remain energy-intensive. Kwon et al.[13] empirically assessed continual learning methods for mobile sensing on platforms like Jetson Nano GPU and smartphones. iCaRL stood out in performance, but its deep learning foundation brings inherent computational challenges, especially in real-world efficiency contexts. Wang et al.[28] also highlighted iCaRL's strengths and concerns in mobile sensing.

In essence, despite promising research in continual learning, the computational strain of deep learning cannot be ignored. The ideal approach would focus on efficient continual learning, blending algorithmic innovation with structural enhancements to minimize computational demands.

2.3 Tsetlin Machines

Rooted in automata theory, TMs use propositional logic for computation, offering both simplicity and interpretability[8]. Their utility spans text classification, audio keyword spotting, and sentiment analysis[14, 24], with specialized versions expanding their reach[6, 7, 9]. Recent tools, such as REDRESS, highlight TM's suitability for edge devices, boasting impressive speed and compression against binary neural networks[18]. The Lite-TM framework then propelled TMs to a new efficiency level, employing memory and latency-reducing techniques and adapting to energy resources[1]. This optimization outperformed certain binarized neural networks in energy efficiency and accuracy, especially in energy-harvesting scenarios.

While these advancements have broadened the horizons for TMs in various computational contexts, our research narrows the focus to a domain where TMs have yet to be explored. Specifically, our work represents a pioneering effort to harness the capabilities of TMs in continual learning. By addressing challenges such as memory footprint, latency, and dynamic adaptation to energy resources, we demonstrate that TM offers substantial advantages over deep neural networks, particularly in the context of continual learning across diverse computing platforms.

3 PRELIMINARIES OF TSETLIN MACHINES

Having highlighted the potential of TMs as a promising solution for continual learning, it is crucial to understand the machinery that powers this innovative paradigm. This section aims to elucidate the origins, mechanics, and advanced variants of TMs, setting the groundwork for our exploration of its application in continual learning.

3.1 Foundations in Automata Theory

Automata theory studies the behaviours of abstract computing devices, emphasizing their state transitions based on inputs. The Tsetlin Automaton, pivotal to this theory, underpins TM. This learning automaton, with its finite states, defines specific actions and refines them based on environmental feedback, either rewards or penalties. By making small adjustments, the Tsetlin Automaton excels at identifying the best actions in diverse contexts, enabling TM to process data and progressively learn. For simplification, TM is essentially a system that selects and combines features to efficiently recognize and classify various patterns.

3.2 Basic Mechanics

TMs offer an alternative paradigm to traditional neural networks. Contrary to employing continuous weights, TMs rely on conjunctive clauses constructed from binary input variables $x_1, \dots, x_n$ or their negations $\neg x_1, \dots, \neg x_n$. These clauses, utilizing Tsetlin Automata (TA) Teams, capturing unique data patterns, are represented as:

$$c_j(X) = \bigwedge_{lk \in L_j} lk \tag{1}$$

where $c_j(X)$ evaluates to 1 if every literal in its set is true. The literals in c_j are determined by the state of the Tsetlin Automaton, which adapts in response to environmental feedback. This ensures the formation of diverse clauses that encapsulate intricate data patterns. In stark contrast to many neural networks, TMs offer an advantage in operational transparency. For classification purposes, TMs compute the sum of outputs from both positive and negative clause groups, subsequently applying a threshold:

$$sum(X) = \sum_{j=1}^{m/2} o_j^+ - \sum_{j=1}^{m/2} o_j^- \tag{2}$$

The resultant classification is then ascertained by:

$$\hat{y} = \begin{cases} 1 & \text{if } sum(X) \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

By leveraging this logical mechanism, TMs adeptly recognize and differentiate complex data patterns.

3.3 Learning Dynamics

TM refines its clauses through a sophisticated feedback mechanism, as presented in [8], to achieve enhanced pattern recognition accuracy. This learning process is governed by two distinct feedback types:

Type I Feedback:

- Activated when the output $\hat{y}$ aligns with the polarity ω for a clause c_j^{ω}.

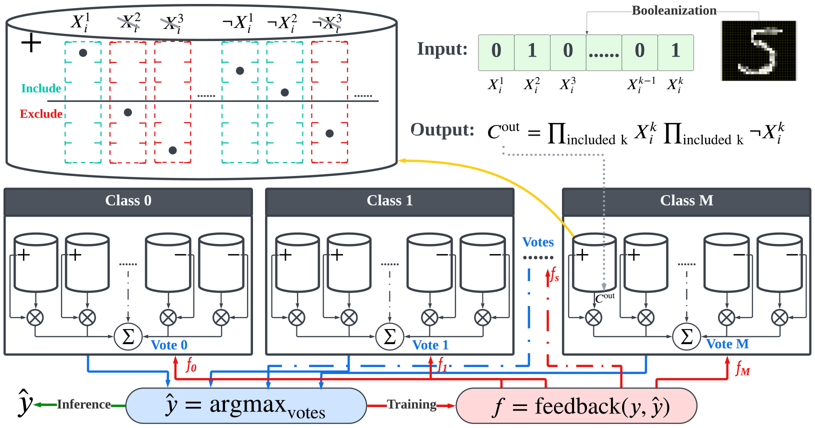

Booleanization
Input:
0
1
0
......
0
1
X_i^1 X_i^2 X_i^3 X_i^{k-1} X_i^k
Output: $C^{\text{out}} = \prod_{\text{included k}} X_i^k \prod_{\text{included k}} \neg X_i^k$
X_i^1 $\cancel{X_i^2}$ $\cancel{X_i^3}$ $\neg X_i^1$ $\neg X_i^2$ $\neg \cancel{X_i^3}$
+
Include
Exclude
Class 0
Class 1
Class M
Votes
C^{out}
Vote 0
Vote 1
Vote M
f_0
f_1
f_s
f_M
$\hat{y}$
Inference
$\hat{y} = \text{argmax}_{\text{votes}}$
Training
$f = \text{feedback}(y, \hat{y})$

Figure 2: Visualization of the key components of TMs and their roles in the decision-making process.

- Rewards the "Include" action and penalizes the "Exclude" action when the clause output is 1, aiming to boost the number of clauses that evaluate correctly to 1.
- Utilizes probabilities $p(\text{Reward}) = \frac{s-1}{s}$, $p(\text{Penalty}) = \frac{s-1}{s}$, and $p(\text{Inaction}) = \frac{1}{s}$ dependent on the clause output, literal value, and automaton action. Herein, s is the user-defined hyperparameter.

Type II Feedback:

- Engaged when the output $\hat{y}$ does not match the polarity ω for a clause c_j^{ω}.
- Penalizes "Exclude" actions for literals that evaluate to 0 when the clause output is 1, driving the inclusion of literals to augment discrimination power.
- Applies solely "Inaction" feedback when the clause output is 0 to prevent entrapment in local minima.
- Uses $p(\text{Penalty}) = 1$ when excluding a 0 literal and the clause output is 1. In other instances, only "Inaction" feedback is applied.

Together, Type I feedback amplifies the prevalence of correct 1 outputs, while Type II feedback incorporates literals to steer clauses to evaluate to 0 when required. Collectively, these feedback mechanisms aim to minimize expected output discrepancies and guide the TA towards optimal clause configurations.

3.4 Tsetlin Machine Variants

Beyond the standard TM, various TM variants address distinct challenges and improve its efficacy. The most notable are the Convolutional TM (CTM) and the Weighted TM (WTM).

3.4.1 Convolutional Tsetlin Machine. Inspired by convolutional neural networks (CNNs), the CTM combines TM with convolutional strategies. Instead of processing whole images, clauses in CTM serve as convolutional filters for specific image segments, enhancing the detection of localized patterns. For an image of size $X \times Y$ with a $W \times W$ filter, the CTM yields N outputs, each representing a distinct image patch. These outputs converge via an OR operation, giving an overall clause response. Image patches are further enriched with coordinate information, ensuring spatial sensitivity. The CTM's learning leverages feedback from random patches among the N outputs, emphasizing localized learning.

3.4.2 Weighted Tsetlin Machine. The WTM evolves the traditional TM by assigning a weight, denoted by w_j, to every clause c_j. Its hallmark is the weighted sum:

$$sum'(X) = \Sigma(w_j^+ \cdot o_j^+) - \Sigma(w_j^- \cdot o_j^-) \tag{4}$$

This ensures each clause's impact on the final decision mirrors its weight. It thus emulates the effects of multiple same clauses through weight enhancement, refining the model's design. Fractional weights enable nuanced clause influence adjustments. The WTM learning algorithm alters weights based on specialized feedback, amplifying weights for true positives and reducing for false positives, reinforcing consistently accurate clauses.

These variants introduced serve to expand the applicability and enhance the performance of TM. In the subsequent sections of this paper, we will evaluate the effectiveness of regular TM and the variants within our proposed ***AdaTM*** framework.

4 AdaTM

In this section, we formally introduce ***AdaTM***, a TM crafted for continual learning. ***AdaTM*** stands out for its innate capacity to assimilate new tasks, preserve historical knowledge, and maintain computational efficiency. Its architecture is underpinned by a combination of mechanisms addressing class integration, knowledge rejuvenation, and model scalability. We will explore these facets in greater depth in the subsequent sections.

4.1 Problem Setup

Continual learning encompasses various approaches, with our focus on class-based continual learning. In this approach, each new task $\mathcal{T}_i$ introduces separate classes, aiming to integrate new knowledge while preserving the understanding of previous classes from tasks $\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}$. Conventional neural networks, with their dense connections and shared weight matrix W. risk blending class boundaries. Thus, when W is adjusted for a new class or task $\mathcal{T}_i$, information on earlier classes might be lost, causing catastrophic forgetting.

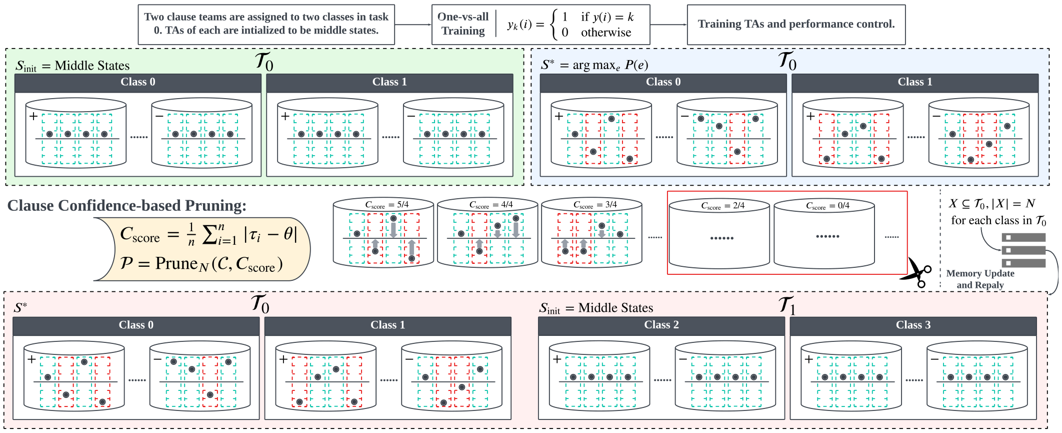

Two clause teams are assigned to two classes in task 0. TAs of each are intialized to be middle states.
One-vs-all Training
$y_k(i) = \begin{cases} 1 & \text{if } y(i) = k \\ 0 & \text{otherwise} \end{cases}$
Training TAs and performance control.
S_{init} = Middle States
$\mathcal{T}_0$
Class 0
Class 1
+
−
$S^* = \arg\max_e P(e)$
$\mathcal{T}_0$
Class 0
Class 1
Clause Confidence-based Pruning:
$C_{\text{score}} = \frac{1}{n}\sum_{i=1}^{n} |\tau_i - \theta|$
$\mathcal{P} = \text{Prune}_N(\mathcal{C}, C_{\text{score}})$
C_{score} = 5/4
C_{score} = 4/4
C_{score} = 3/4
C_{score} = 2/4
C_{score} = 0/4
$X \subseteq \mathcal{T}_0, |X| = N$
for each class in $\mathcal{T}_0$
Memory Update and Repaly
S^*
$\mathcal{T}_0$
Class 0
Class 1
S_{init} = Middle States
$\mathcal{T}_1$
Class 2
Class 3

Figure 3: Illustration of AdaTM's performance tracking and optimal state recording process between tasks T0 and T1. Depicts key steps of training, calculating performance metrics, identifying optimal states, memory updates, and clause pruning.

Algorithm 1 AdaTM with Pruning for Continual Learning

```
 1: procedure AdaptiveTsetlinContinual($\mathcal{T}_i, k, \alpha, \beta, N$)
 2:     Initialize memory buffer $\mathcal{B}$ of size $k$
 3:     Initialize clause teams $C$
 4:     while new task $\mathcal{T}_i$ arrives do
 5:         Create new clause teams $C_{new}$ for classes in $\mathcal{T}_i$
 6:         Append $C_{new}$ to $C$
 7:         Interleave training data of $\mathcal{T}_i$ with samples from $\mathcal{B}$
 8:         for each epoch $e$ do
 9:             Train using Adaptive Tsetlin Machine on merged data
10:             Calculate $P(e)$ with Eq. 7
11:             if $P(e)$ is maximum so far then
12:                 $S^*$ = current state
13:             end if
14:         end for
15:         Update $\mathcal{B}$ with samples from $\mathcal{T}_i$ ensuring class balance
16:         for each clause $c$ in $C$ do
17:             Calculate confidence score $C$ using $\tau$ and $\theta$
18:         end for
19:         Sort clauses in $C$ by confidence scores
20:         Remove the bottom $N$ clauses from $C$
21:     end while
22: end procedure
```

TM offers an alternative. Instead of dense connections, it employs **clause teams**, denoted as $C = \{c_1, c_2, \ldots, c_m\}$. ensuring dedicated representations for each class. This structure better retains knowledge. However, with TM's dependency on reinforcement learning, new data can influence TA states, potentially leading to a form of forgetting. While TM reduces interference, it's still vulnerable to challenges in class-based continual learning. This highlights the value of our ***AdaTM*** contribution.

4.2 Adaptive Tsetlin Machine

4.2.1 Adaptive Growth with Attention to Fading Knowledge. While TM's architecture naturally facilitates the integration of clause teams for emergent classes, as shown in steps 5 to 7 in **Algorithm1**, safeguarding the efficacy of established clause teams, denoted as $C_{\text{old}} \subseteq C$, remains a pivotal concern. In the absence of consistent engagement with relevant samples, these mature clause teams are susceptible to a gradual erosion in their representational accuracy.

To navigate this challenge, we introduce a memory buffer, $\mathcal{B}$. The size of the memory buffer, denoted as k, is empirically determined based on the training data size and the computational capacity available. Its primary aim is to retain a subset of past tasks in a way that offers a balanced representation of various classes. The buffer contains binarized samples, specifically input-output pairs. Formally, the buffer is represented as

$$\mathcal{B} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\} \tag{5}$$

where each (x_i, y_i) corresponds to input-output pairs culled from prior tasks. For a given class, there exists a corresponding subset $\mathcal{B}_j$ within $\mathcal{B}$, delineated by the relationship:

$$\mathcal{B}_j = \mathcal{B} \cap (X_j \times Y_j) \tag{6}$$

where X_j and Y_j represent the input and output spaces of the class, respectively. The input space X_j corresponds to the set of all possible input data instances that can belong to the class j, while the output space Y_j signifies the possible output labels or responses specific to that class.

As the model undergoes training, data from $\mathcal{B}$ are integratively interleaved into the learning process. By periodically revisiting instances from this buffer, the system not only rejuvenates older clause teams but also consolidates pre-existing class delineations. As tasks unfold, while the buffer undergoes iterative updates, it remains anchored in its foundational principle of class-balanced representation (step 15 in **Algorithm1**). This multi-faceted strategy empowers TM to adeptly navigate the continual learning landscape, ensuring consistent performance even in the face of evolving data streams.

4.2.2 Rigorous Performance Metrics for Stochastic Trajectories. TM, by its very nature, is inherently susceptible to stochastic fluctuations. This, coupled with its reinforcement attributes, makes its trajectory during learning phases more intricate. To quantify its performance, we define P after each epoch e as:

$$P(e) = \alpha \times ACC_{\text{avg}} + \beta \times (1 - FM_{\text{avg}}) \tag{7}$$

where ACC_{avg} represent the continual learning average accuracy that has been defined in Eq. 11, FM_{avg} is the average forgetting measure that has been defined in Eq. 12, α and β are weighting coefficients that balance accuracy and retention, respectively.

For optimal continual learning, it becomes paramount to harness the best learning state. As such, we identify and archive the state S^* corresponding to the epoch exhibiting peak performance:

$$S^* = \arg\max_e P(e) \tag{8}$$

This state, a representation of the machine's optimal configuration at a given time, then serves as the foundational starting point for initializing clause teams during the subsequent task training phases, ensuring a harmonious blend of past insights with newfound knowledge. To gain a comprehensive understanding of the entire process as described, Fig. 3 depicts how the methodology operates between learning in $\mathcal{T}_0$ and when $\mathcal{T}_1$ emerges.

4.3 Clause Pruning Mechanism: Addressing Scalability Concerns

The ***AdaTM***'s architectural propensity to organically expand in response to emerging classes is both a strength and a potential source of inefficiency. While this expansion provides a robust framework for capturing new knowledge, it inadvertently leads to linear growth in the model size, which can, over time, hinder computational performance and real-time applicability. Particularly in scenarios involving a multitude of classes, the proliferation of clause teams can challenge the inherent efficiency for which TM is acclaimed.

To address this scalability challenge, we introduce a clause confidence score-based pruning strategy, aimed at judiciously reducing the incremental growth of clause teams without undermining the representational capacity of the model.

4.3.1 Clause Confidence Formulation. The intrinsic reinforcement learning paradigm of TM means that the states of TA inherently reflect their confidence levels in including or excluding specific input features. Let's denote the state of a particular TA as τ, where $\tau \in S$ and S is the set of all possible states. The boundary between the inclusion and exclusion states can be designated as θ. A clause's confidence score C_{score} can be mathematically quantified by considering the aggregate deviation of its associated TA states from this boundary B. Mathematically, for a clause with n TAs:

$$C_{\text{score}} = \frac{1}{n} \sum_{i=1}^{n} |\tau_i - \theta| \tag{9}$$

A greater distance of τ_i from θ implies higher confidence in the decision made by the respective TA, and consequently, the clause to which it belongs. Thus, the clauses with the highest aggregated scores, representing the most decisive patterns, are deemed paramount. This represents a novel and computationally efficient method for evaluating clause confidence, distinctively characteristic of the TM domain.

4.3.2 Pruning Strategy. Guided by the confidence scores, our pruning methodology can be succinctly captured as follows: At the end of the training phase for each task, clauses with the lowest confidence scores, which denote uncertainty or less importance in their representational capacities, are pruned. Specifically, the least N clauses, as per their confidence scores, are removed, thereby mitigating the overhead of unnecessary expansion.

$$\mathcal{P} = \text{Prune}_N(C, C_{\text{score}}) \tag{10}$$

Where $\mathcal{P}$ denotes the set of pruned clauses and Prune_N represents the function that sorts and prunes the N clauses with the lowest confidence scores from the set of all clauses C. The equivalent process visualization can be seen in Fig. 3 when a new task $\mathcal{T}_\infty$ arrives.

This strategy strikes a balance. While the adaptive growth allows for capturing novel class information, the pruning ensures the model remains computationally efficient. By grounding our pruning mechanism in the intrinsic properties of TA, we ensure that the pruned elements are those that contribute the least to the model's efficacy, thus preserving the integrity and performance of TM across tasks. To grasp the role and intricacies of the pruning mechanism within our framework, we direct the reader to steps 16 to 20 in **Algorithm1**.

5 EXPERIMENTAL SETUP

5.1 Datasets

Our selected datasets, displayed in Table 1, span a wide range of applications, showcasing ***AdaTM***'s versatility. Despite their limited complexity, these foundational datasets are pivotal for on-device implementations, particularly in the context of edge AI. The primary objective of using these datasets is to expediently iterate, optimize, and validate, thereby ensuring the model's efficiency and the feasibility of the proposed techniques in real-world scenarios.

Highlighting our class-incremental learning, each dataset was split into N tasks, introducing 2 new classes per task. This is in line with typical continual learning settings, like in split-MNIST, where classes are added incrementally to mirror real-world situations[21, 27], testing the system's adaptability and retention of prior knowledge.

Dataset Summary
MNIST: A dataset of handwritten digits with 10 classes. It has 60,000 training and 10,000 test samples, used for 5 tasks with 2 classes per task.
FashionMNIST: A dataset of clothing items with 10 classes. It has 60,000 training and 10,000 test samples, used for 5 tasks with 2 classes per task.
AudioMNIST: A dataset for audio recognition with 10 classes. It has 27,000 training and 3,000 test samples, used for 5 tasks with 2 classes per task.
TESS: A dataset for audio emotion recognition with 6 classes. It has 2,240 training and 560 test samples for 3 tasks with 2 classes per task.
PAMAP2: A dataset of human physical activity with 12 classes. We randomly selected 96,000 training and 24,000 test feature sets for 6 tasks with 2 classes per task.

Table 1: Summary of datasets used in the study.

Models	Type	Structure	Hyper-parameters
MLP	Full-precision model	2 Linear layers (256 neurons, ReLU) + Linear(256, num classes)	Learning rate: 0.01; Batch size: 500; Optimizer: SGD
LeNet5	Full-precision model	3 Conv layers (6, 16, 120 channels; 5x5 kernel, Tanh, BatchNorm) + 2 AdaptiveAvgPool (14x14, 5x5) + Linear(120, 84) + Tanh + Linear(84, num classes)	Learning rate: 0.1; Batch size: 500; Optimizer: SGD
ResNet18	Full-precision model	ResNet(4 layers, 2 BasicBlocks each) + Linear(2560, num classes)	Learning rate: 0.1; Batch size: 500; Optimizer: SGD
B-MLP	Binary model	2 Binarized Linear layers (256 neurons, HardTanh) + Binarized Linear(256, num classes)	Learning rate: 0.01; Batch size: 500; Optimizer: SGD
B-LeNet5	Binary model	3 Binarized Conv layers (6, 16, 120 channels; 5x5 kernel, HardTanh, BatchNorm) + 2 AdaptiveAvgPool (14x14, 5x5) + Binarized Linear(120, 84) + HardTanh + Binarized Linear(84, num classes)	Learning rate: 0.1; Batch size: 500; Optimizer: SGD
B-ResNet18	Binary model	ResNet(4 layers, 2 BasicBlocks each with Binarized Conv and HardTanh) + Binarized Linear(2560, num classes)	Learning rate: 0.1; Batch size: 500; Optimizer: SGD
TM	Non-weighted TM	1000 clauses per class, 8 TA states	T: 30; s: 15
CTM	Non-weighted CTM	1000 clauses per class, 8 TA states, (10, 10) patch size for MNIST and FashionMNIST, (10, 13) patch size for AudioMNIST and TESS	T: 30; s: 15
WTM	Weighted TM	500 clauses per class, 8 TA states	T: 100; s: 5
WCTM	Weighted CTM	500 clauses per class, 8 TA states, (10, 10) patch size for MNIST and FashionMNIST, (10, 13) patch size for AudioMNIST and TESS	T: 100; s: 5

Table 2: Specifications and hyperparameter settings for the various models compared in the study, categorized by model type and structure. Provides comprehensive details to understand the model configurations used for performance benchmarking.

5.2 Baselines and Models

To assess ***AdaTM***'s efficiency, the benchmarks selected range from classic but effective EWC to state-of-the-art GDumb. The baseline models and their respective settings are as follows: EWC with $\lambda = 1$; LwF with $\phi = [0, 0.5, 1.333, 2.25, 3.2]$ and $temperature = 2$; SI with $\lambda = 1$ and $\epsilon = 0.1$; DGR, ER, and iCaRL, each with a memory buffer of 1000 samples; and GDumb with a memory buffer of 4400 samples. ***AdaTM*** also had a 1000-sample buffer. In this context, "memory" denotes the buffer size or replay sample count used by models. Specifically for the TESS dataset, due to its limited size, the memory buffer for all strategies was adjusted to 100. For reference, the approximate buffer sizes for the datasets used were: MNIST and FashionMNIST at **3.14** MB each, AudioMNIST at **1.51** MB, TESS at **0.47** MB, and PAMAP2 at **0.20** MB.

When determining the neural network architectures for our evaluations, we were guided by a series of deliberate decisions, with an emphasis on efficiency for edge computing applications:

1. **Multilayer Perceptron (MLP)**: This basic feedforward setup provides a foundational benchmark.
2. **LeNet5**: Included as a standard CNN, it facilitates comparison with AdaTM.
3. **Reduced ResNet18**: A streamlined ResNet18 caters to modern, deeper designs while fitting on-device learning, balancing complexity with efficiency for constrained settings.

To ensure a fair comparison in terms of efficiency, we tested both full-precision and binary versions of these networks with concise architectures or simplified versions appropriate for edge computing.

For TM models, both regular and convolutional types were evaluated, including weighted variants. Network and hyper-parameter specifics are in Table 2. All experiments were repeated five times.

5.3 Evalutation Metrics

To systematically assess the AdaTM in continual learning scenarios, we employ a set of metrics, each expressed with their respective symbols:

5.3.1 *Average Accuracy (ACC_{avg}).*

$$\text{Acc}_{\text{avg}} = \frac{\sum_{i=1}^{N} \text{Acc}_{\text{ovr},i}}{N} \tag{11}$$

Where N is the total number of tasks, and $\text{Acc}_{\text{ovr},i}$ is task-wise overall accuracy that averages performance up to the i^{th} task. This metric encapsulates the model's consistent performance over all tasks.

5.3.2 Forgetting Measure (FM). Given the n^{th} task:

$$\text{FM}_n = \frac{\text{Acc}_{\text{prev},n} - \text{Acc}_{\text{ovr},n-1}}{\text{Acc}_{\text{prev},n}}$$

Where $\text{Acc}_{\text{prev},n}$ is the task-wise previous accuracy that measures the knowledge retention after introducing the n^{th}. Then averaging across all tasks:

$$\text{FM}_{\text{avg}} = \frac{\sum_{i=2}^{N} \text{FM}_i}{N-1} \tag{12}$$

This metric discerns the model's performance decrement as new tasks emerge, providing a lens into its cross-task retention capabilities.

5.3.3 Processing Latency ($\mathcal{L}$). Captures the entire time taken across all tasks, including training and validation phases. It's a testament to the model's processing efficiency.

5.3.4 Maximum Run Memory ($\mathcal{M}$). A direct measure signifying the peak memory consumption, spotlighting the model's deployability in resource-tight settings.

5.3.5 Energy Consumption. This metric quantifies the energy consumed by the model on a Raspberry Pi, vital for understanding energy efficiency in on-device AI deployments. We integrated the power (measured with a current and voltage meter) over the model's runtime to determine the total energy, denoted as $\mathcal{E}$, spanning all tasks. We also noted the peak power $\mathcal{P}_{\text{peak}}$ to gauge maximum power demand, offering a thorough energy profile of the model on the device.

5.4 Hardware Specifications

We ran experiments on two computational environments, both using the CPU for training, to verify its adaptability across platforms. The first was a laptop with a 13th Gen Intel Core i7-13850HX, having 20 cores (8 performance, 12 efficient) and a peak frequency of 5.30 GHz. The second was a Raspberry Pi Model 4B, featuring a Broadcom BCM2711 Quad-core Cortex-A72 SoC at 1.5 GHz and 8GB LPDDR4-3200 SDRAM. We used PyTorch, with memory tracked by Memory Profiler.

6 RESULTS AND DISCUSSION

In this section, we explore the empirical performance of ***AdaTM*** against traditional neural network continual learning methods. By testing across varied benchmarks and environments, we underline the distinct benefits of ***AdaTM*** and its promise as a strong alternative to current strategies.

6.1 Average Accuracy and Forgetting Measure

6.1.1 AdaTM's Dominance in Comparative Performance. In our systematic evaluation across datasets, as showcased in Table 3, ***AdaTM*** consistently exhibited remarkable performance. For the MNIST and FashionMNIST datasets, ***AdaTM*** displayed dominance, achieving an accuracy of **97.29%** and **82.67%**respectively, thereby outperforming both ER and GDumb. Notably, while ***AdaTM*** took the lead in accuracy, GDumb showcased slightly better FM_{avg} in the FashionMNIST context. In datasets like AudioMNIST and TESS, ***AdaTM***'s performance persisted impressively. While GDumb achieved the highest accuracy in AudioMNIST, ***AdaTM*** wasn't far behind, and indeed, outperformed ER. However, the advancement of GDumb in both FashionMNIST and AudioMNIST can be attributed to its use of ResNet, which is resource-intensive compared to TM-based models. Next, the TESS dataset further illuminated ***AdaTM***'s prowess, with an accuracy of **96.26%**, surpassing Replay and iCaRL. Finally, for the PAMAP2 dataset, ***AdaTM*** maintained its robust performance with an accuracy of**74.80%** and low forgetting measures. iCaRL obtained the best results in the forgetting measures but at the expense of far reduced accuracy. Note that in our experiments, without the continual learning capability of ***AdaTM***, all TM models exhibited a pronounced disparity between their performance on current and prior tasks. While the models consistently achieved over **95%** accuracy on current tasks, their recall of prior tasks was notably compromised, with accuracies ranging from **0.4%** to **9%**.

6.1.2 Insights into Disparate Strategy Performances. Delving deeper into the comparative analysis, EWC and DGR notably lagged behind. Here, due to its regularization-based nature and similar results, only EWC is listed as a representative in the table, while LwF and SI are omitted. These strategies' performance across datasets such as MNIST, FashionMNIST, and AudioMNIST were particularly concerning, often hovering around or below the **50%** mark. In particular, we found EWC, LwF, and SI to be ill-equipped to handle class-incremental learning scenarios in our experiments. These methods rely on imposing penalties to prevent significant changes to previously learned weights. However, in a dynamic learning landscape with continually added classes, these penalties might be too restrictive, thereby hindering the model's capacity to learn new information without drastic loss of old knowledge. DGR while conceptually promising, exhibited inconsistency in its results. The performance of DGR is closely tied to the quality of the generated samples. Given a constrained training budget, the generator may not produce high-fidelity samples, thus affecting the performance of the classifier trained on them.

6.1.3 Delving into the AdaTM Framework. Furthermore, within the ***AdaTM*** framework, different variants show strong performance across datasets. However, their distinct configurations and inherent characteristics can lead to varied efficiencies. Specifically, the CTM, due to its more complex structure, may require increased processing time. On the other hand, weighted models like WTM and WCTM might achieve good results with fewer clauses, suggesting a more streamlined efficiency. Additionally, when introducing the pruning mechanism paired with WTM, it shows marginal performance differences across datasets compared to non-pruned cases. As we delve into these models, it becomes evident that a balance between performance and efficiency is a recurring theme within the ***AdaTM*** framework. This is worth further discussion in Section 6.2.

6.1.4 Interesting Findings and Considerations. An intriguing observation is the presence of negative forgetting measures in some models, including, notably, iCaRL on the TESS dataset and TMs on PAMAP2. A negative forgetting measure implies that a model's performance on previous tasks improves as it learns new tasks. This phenomenon, while counterintuitive, could be attributed to fluctuations during learning different tasks. A more in-depth exploration of this occurrence can be found in Section 6.3. Another interesting observation is that iCaRL, despite its prominence in the literature, did not consistently outperform our evaluations. This inconsistency might stem from the neural network's simplicity used in the experiments. Given iCaRL's template-based methodology, the exemplar features' quality becomes paramount. Simplistic networks might not capture adequate latent information, leading to suboptimal classification performance.

Aggregating these findings, several observations can be drawn. Primarily, ***AdaTM*** exhibits consistently high performance in terms of accuracy across datasets. ER and GDumb, while competitive, generally follow ***AdaTM***'s lead. In terms of forgetting measures, there is variability across datasets, but ***AdaTM*** consistently ranks among the top models, indicating its robustness in retaining previously learned information. Lastly, we can also observe that the pruning mechanism works well although there might be slight performance deterioration for some benchmarks.

Strategies	Models	MNIST		FashionMNIST		AudioMNIST		TESS		PAMAP2	
		ACC_{avg}	FM_{avg}	ACC_{avg}	FM_{avg}	ACC_{avg}	FM_{avg}	ACC_{avg}	FM_{avg}	ACC_{avg}	FM_{avg}
EWC	MLP	19.43%	99.82%	19.93%	100.00%	20.10%	99.08%	25.00%	94.44%	13.07%	88.25%
	LeNet	22.77%	83.51%	21.89%	96.47%	21.33%	86.65%	55.31%	75.73%	-	-
	ResNet	19.72%	100.00%	19.96%	100.00%	19.88%	100.00%	48.85%	100.00%	-	-
	B-MLP	21.94%	84.81%	23.22%	93.29%	26.30%	78.41%	52.17%	73.84%	14.70%	81.64%
	B-LeNet	19.79%	95.59%	19.30%	99.87%	19.57%	99.22%	48.83%	92.46%	-	-
	B-ResNet	16.92%	100.00%	18.46%	100.00%	24.93%	91.44%	38.67%	95.91%	-	-
GDumb	MLP	85.97%	2.72%	72.12%	7.55%	77.07%	7.12%	57.26%	19.66%	43.42%	14.47%
	LeNet	85.52%	2.32%	71.30%	**7.16%**	87.42%	2.98%	56.43%	21.01%	-	-
	ResNet	93.92%	1.19%	78.79%	**5.50%**	**96.25%**	**0.71%**	26.56%	50.00%	-	-
	B-MLP	73.73%	4.18%	65.50%	9.10%	55.69%	10.25%	38.38%	25.69%	23.60%	16.97%
	B-LeNet	78.32%	5.01%	51.58%	15.25%	61.32%	8.66%	54.56%	25.53%	-	-
	B-ResNet	14.51%	**-0.46%**	25.99%	26.28%	66.45%	6.48%	43.36%	22.31%	-	-
DGR	MLP	53.61%	26.71%	47.46%	43.95%	24.95%	84.40%	35.48%	100.00%	23.18%	67.43%
	LeNet	36.99%	32.94%	20.09%	96.94%	29.10%	72.86%	34.65%	100.00%	-	-
	ResNet	31.00%	42.17%	19.81%	100.00%	19.78%	100.00%	24.90%	100.00%	-	-
	B-MLP	54.71%	12.38%	36.21%	51.87%	32.65%	66.08%	20.12%	100.00%	19.26%	51.07%
	B-LeNet	50.99%	24.49%	44.74%	31.71%	34.05%	57.00%	30.91%	100.00%	-	-
	B-ResNet	23.15%	57.28%	10.00%	85.16%	23.88%	85.50%	28.42%	100.00%	-	-
ER	MLP	90.57%	3.05%	80.55%	8.38%	71.14%	10.80%	50.21%	48.25%	38.96%	18.00%
	LeNet	**96.24%**	**1.07%**	**81.80%**	8.05%	**93.39%**	**1.62%**	82.57%	20.01%	-	-
	ResNet	95.92%	1.62%	79.03%	11.96%	**93.55%**	**2.02%**	37.76%	65.85%	-	-
	B-MLP	83.98%	6.25%	73.49%	13.40%	74.03%	6.26%	70.33%	27.27%	27.65%	27.34%
	B-LeNet	89.59%	2.93%	66.65%	9.88%	74.75%	4.83%	72.61%	21.25%	-	-
	B-ResNet	83.25%	3.14%	72.44%	11.99%	89.37%	3.26%	54.15%	38.53%	-	-
iCaRL	MLP	80.88%	2.62%	70.20%	7.44%	73.11%	5.90%	62.04%	18.41%	49.39%	**2.94%**
	LeNet	76.11%	1.91%	64.63%	7.78%	67.03%	8.63%	65.43%	6.16%	-	-
	ResNet	83.04%	1.44%	72.37%	**7.25%**	90.97%	2.34%	82.81%	**-7.50%**	-	-
	B-MLP	78.84%	1.96%	66.98%	13.06%	68.71%	8.23%	79.51%	**4.65%**	30.82%	**9.41%**
	B-LeNet	46.96%	13.94%	44.94%	24.06%	41.04%	15.66%	53.92%	31.08%	-	-
	B-ResNet	38.26%	10.23%	40.57%	15.69%	-	-	-	-	-	-
AdaTM	TM	93.88%	2.59%	80.55%	12.54%	90.45%	3.56%	**93.36%**	7.43%	**70.82%**	**15.64%**
	CTM	**97.03%**	**1.25%**	**82.67%**	8.16%	89.15%	4.08%	88.80%	10.88%	-	-
	WTM	93.87%	2.35%	80.79%	10.08%	91.21%	2.91%	**96.26%**	**3.62%**	**74.80%**	**-3.00%**
	WCTM	**97.29%**	**1.04%**	**82.33%**	9.35%	88.92%	3.52%	91.29%	8.53%	-	-
AdaTM+Prune	WTM	92.54%	2.88%	80.53%	11.28%	90.99%	2.82%	**98.55%**	**0.49%**	**73.02%**	**-8.66%**

Table 3: Comparative performance metrics of AdaTM and other models on diverse datasets. Highlights AdaTM's efficacy for continual learning across different benchmarks compared to SOTA techniques.

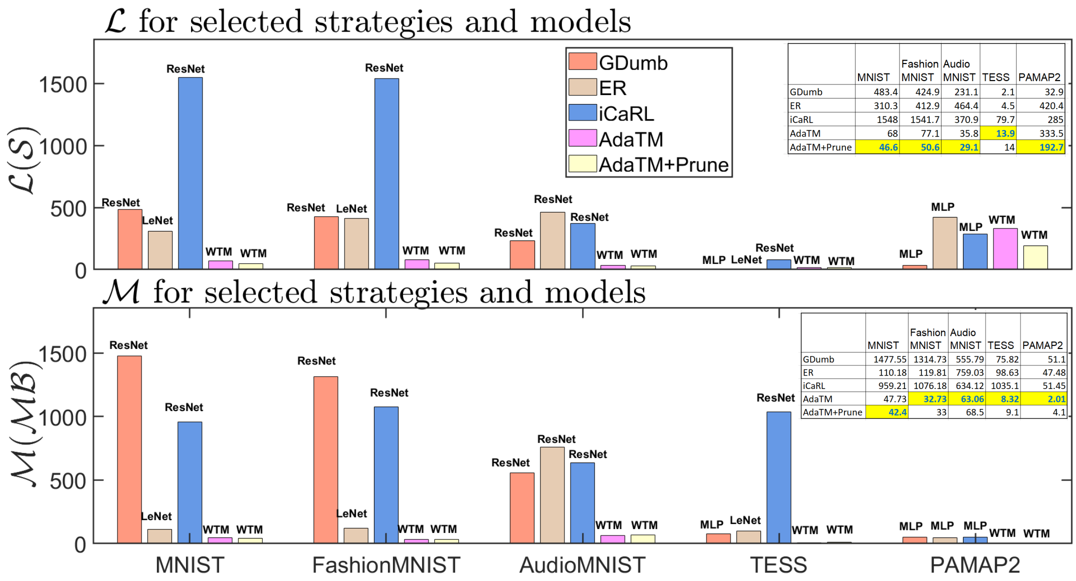
6.2 Efficiency Analysis

We will now discuss the resource and energy-related measures for various continual learning approaches, especially for those that showed the best performance.

6.2.1 Run Time Memory. In evaluating memory consumption metrics, ***AdaTM*** and ***AdaTM+Prune*** consistently demonstrate efficiency, as shown in the bottom plot in Fig. 4. For instance, on the MNIST, both strategies consume less memory than GDumb's ResNet, the most memory-intensive method, with ***AdaTM*** saving approximately **31x** and ***AdaTM+Prune*** saving about **35x** in comparison. They also significantly outperform ER's LeNet. On FashionMNIST, the trend persists as ***AdaTM*** and ***AdaTM+Prune*** require significantly less run memory than GDumb with ResNet, the peak consumer. Similarly, ***AdaTM*** and ***AdaTM*** with pruning maintain a lean profile on PAMAP2, using about **26x** and **13x** less run memory, respectively compared to iCaRL with MLP. Such comparisons further underscore ***AdaTM*** and ***AdaTM+Prune***'s efficiency, positioning them as preferable choices against a range of other strategies. In fact, memory constraints are a primary factor in determining if a particular machine learning design can be practically realized on an edge device or not. ***AdaTM***'s compact memory profile enables continual learning in such constrained environments, positioning it as a standout choice for devices where memory resources are limited.

6.2.2 Latency. Processing latency varies across models and strategies. Our analysis discerns between latency-focused strategies and those balancing latency with accuracy. It is worth mentioning that results for EWC, LwF, SI, and DGR are omitted due to their consistently poor accuracy and forgetting measures across datasets.

The top plot of Fig. 4, ***AdaTM*** showcases latency measured on the laptop. On the MNIST dataset, ***AdaTM*** surpasses iCaRL and ER by factors of **23** and **4.5**, respectively. The ***AdaTM+Prune*** strategy accentuates this efficiency. For example, on FashionMNIST, pruned ***AdaTM*** outstrips GDumb's ResNet, enhancing AdaTM's performance by **(52.37%)**. This trend is evident in AudioMNIST, where AdaTM+Prune improves latency over ***AdaTM*** by **(23.02%)**. On TESS and PAMAP2, although ***AdaTM*** and ***AdaTM+Prune*** may not achieve the least latencies, they excel in accuracy **(35%/27%)** improvement over GDumb for TESS/PAMAP2). Notably, their increased latency on PAMAP2 arises as TM processes samples sequentially, unlike batch processing in deep neural networks, prompting potential optimization in future versions of our work on Tsetlin Machines.



	MNIST	Fashion MNIST	Audio MNIST	TESS	PAMAP2
GDumb	483.4	424.9	231.1	2.1	32.9
ER	310.3	412.9	464.4	4.5	420.4
iCaRL	1548	1541.7	370.9	79.7	285
AdaTM	68	77.1	35.8	**13.9**	333.5
AdaTM+Prune	**46.6**	**50.6**	**29.1**	14	**192.7**

	MNIST	Fashion MNIST	Audio MNIST	TESS	PAMAP2
GDumb	1477.55	1314.73	555.79	75.82	51.1
ER	110.18	119.81	759.03	98.63	47.48
iCaRL	959.21	1076.18	634.12	1035.1	51.45
AdaTM	47.73	**32.73**	**63.06**	**8.32**	**2.01**
AdaTM+Prune	**42.4**	33	68.5	9.1	4.1

Figure 4: Comparative Analysis of Processing Latency and Maximum Run Memory across Different Models and Strategies.

Transitioning from the laptop to a more resource-constrained platform, our investigations extended to the Raspberry Pi. The latency trends were consistent. Specifically, ***AdaTM*** on the Raspberry Pi demonstrated a latency advantage of **1.8x** over the iCaRL paired with LeNet for MNIST. Furthermore, when implementing the ***AdaTM+Prune*** strategy, latency is reduced by a striking **2.7x** compared to its non-pruned counterpart. To put it in perspective, this pruned variant was faster by approximately **5x** and **7.7x** than the LeNet-based iCaRL and ER methods on the MNIST, respectively. Also, it was faster by about **1.2x** than both iCaRL and ER paired with LeNet on the TESS.

Conclusively, regardless of the platform, the results underscore ***AdaTM***'s compelling balance of latency and accuracy. Particularly, ***AdaTM+Prune*** emerges as a robust strategy, delivering enhanced latency and maintaining premier accuracy, setting it apart from competing methodologies.

Model	Strategy	MNIST		TESS	
		$\mathcal{P}_{\text{peal}}$ (Watt/s)	$\mathcal{E}$ (J)	$\mathcal{P}_{\text{peal}}$ (Watt/s)	$\mathcal{E}$ (J)
LeNet	ER	5.94	18790.45	**5.62**	798.59
	GDumb	5.99	**3844.54**	5.87	673.44
	iCaRL	6.17	12784.28	5.92	832.48
MLP	ER	**4.38**	7644.01	5.79	**551.68**
	GDumb	**4.12**	**1403.37**	5.65	**524.44**
	iCaRL	**4.38**	5029.03	5.70	555.06
WTM	AdaTM	**5.31**	7075.95	**5.64**	801.26
	AdaTM+Prune	5.72	**2717.27**	**5.55**	**549.71**

Table 4: On-device performance metrics of AdaTM on Raspberry Pi across key metrics.

6.2.3 On-Device Energy Consumption. The energy metrics presented in Table 4 illuminate the energy efficiency of different model-strategy pairings on a Raspberry Pi deployment. We selected MNIST, a complex dataset, and TESS, a simpler one, to span the range of dataset intricacies. PAMAP2 was excluded due to ***AdaTM***'s pronounced performance edge. For FashionMNIST and AudioMNIST, the intrinsically high energy demands of LeNet and ResNet, evident in their latency and memory metrics, rendered their inclusion less pertinent. As a crucial benchmark, the ***AdaTM*** strategy paired with the WTM model requires a peak power of **(5.31)** Watt/s for MNIST, and **(5.64)** Watt/s for TESS, with total energy consumptions of **(7075.95)** Joules and **(801.26)** Joules, respectively. It can be seen that the latency result is more pronounced on devices with limited computational abilities compared to the previous laptop platform, highlighting this common real-world challenge.

On inspecting the MLP model combined with the GDumb strategy, it manifests the lowest energy consumption of **(1403.37)** Joules, its peak power ascends to **(4.12)** Watt/s for the MNIST dataset. For TESS, this pairing consumes **(524.44)** Joules, with a peak power of **(5.65)** Watt/s. Although MLP models can be energy-efficient, their accuracy is often compromised for more intricate tasks. Consequently, the elevated energy consumption of the ***AdaTM*** framework might be rationalized by its enhanced performance and accuracy. Conversely, the LeNet model, a standard CNN paired with the iCaRL strategy, expends **12784.28** Joules for MNIST, and **832.48** Joules for TESS, marking an approximate **1.8x** increment over the AdaTM and WTM pairing on MNIST. This finding implies that even foundational CNNs like LeNet can surpass a TM model in terms of energy demand and latency. Given this juxtaposition, it is plausible to infer that more elaborate architectures, such as ResNet, would be even more energy-hungry and time-consuming, accentuating the energy-saving merits of TM models in on-device contexts.

Moreover, incorporating pruning into the ***AdaTM*** strategy combined with the WTM model engenders a marked decline in energy expenditure. The ***AdaTM+Prune*** pairing demands merely **(2717.27)** Joules for MNIST, and **(549.71)** Joules for TESS, approximately **2.6x** and **1.5** less energy than the non-pruned ***AdaTM***. This underscores the pruning techniques' promise in augmenting the energy efficiency of sophisticated models, a vital factor for on-device implementations.

6.3 Evaluation of Pruning Mechanism

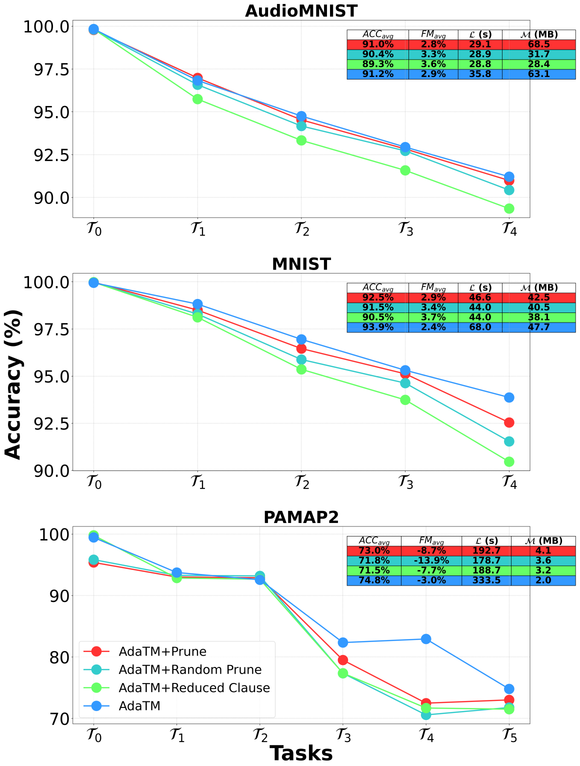


Figure 5: Task-wise comparative analysis of AdaTM pruning mechanisms on key performance metrics.

To evaluate ***AdaTM***'s pruning mechanism, we performed an ablation study on MNIST, AudioMNIST, and PAMAP2 datasets, highlighting its adaptability across different modalities and ensuring focused analysis. Our goal was to discern the strengths of ***AdaTM***'s pruning from various angles. Experimentally: The base ***AdaTM*** uses WTM with 500 clauses per class; ***AdaTM+Prune*** starts with 500 clauses per class, pruning to 200 after each task; AdaTM+Random Prune mimics AdaTM+Prune but uses random instead of deterministic pruning; AdaTM+Reduced Clause begins with 200 clauses per class, with no expansion or pruning. Task-specific and aggregate performances are detailed in Fig. 5.

6.3.1 Effectiveness and Efficiency: AdaTM+Prune vs. AdaTM. To evaluate the impact of the pruning mechanism, we compared AdaTM with pruning to the standard ***AdaTM***. Across the datasets, ***AdaTM*** generally achieved slightly higher accuracy than ***AdaTM+Prune*** **91.21%** vs. **90.99%** for AudioMNIST, **93.87%** vs. **92.54%** for MNIST, and **74.80%** vs. **73.02%** for PAMAP2. AdaTM also marginally outperformed ***AdaTM+Prune*** in knowledge retention on the MNIST dataset. However, when it comes to processing speed, AdaTM+Prune excelled: **29.1** seconds vs. **35.8** seconds for AudioMNIST, **46.6** seconds vs. **68.0** seconds for MNIST, and a notable **192.7** seconds vs. **333.5** seconds for PAMAP2. In terms of memory usage, results were mixed. For AudioMNIST, ***AdaTM*** consumed **63.1** MB compared to AdaTM+Prune's **68.5** MB, and for PAMAP2, ***AdaTM*** was more efficient with **2.01** MB against ***AdaTM+Prune***'s **4.1** MB. This suggests that TM's training memory efficiency might be influenced more by input feature size.

In summary, while the pruning mechanism might marginally affect accuracy in certain cases, it significantly boosts processing efficiency, emphasizing its potential in real-time or edge-computing scenarios.

6.3.2 Deterministic Prune vs. Random Prune. To gauge the effectiveness of our pruning approach, we compared AdaTM+Prune with AdaTM+Random Prune, aiming to highlight the benefits of deterministic pruning over stochastic methods.

Deterministically, ***AdaTM*** with pruning consistently outperforms ***AdaTM*** with the random Pruning method in learning performance. For example, on MNIST, AdaTM+Prune achieved **92.54%** accuracy, while the stochastic method reached **91.54%**. The forgetting measure, crucial in continual learning, further emphasized ***AdaTM+Prune***'s edge, with lower values across datasets indicating better stability. Interestingly, both models showed negative forgetting on the PAMAP2 dataset, possibly from enhanced intermediary task performance. Yet, ***AdaTM+Prune***'s measure of **-8.66%** was less negative than AdaTM+Random Prune's **-13.91%**, suggesting better knowledge retention. Efficiency-wise, textbf*AdaTM+Prune* had slightly higher latencies and memory usage, but these differences were minimal. This indicates that the deterministic pruning's gains in accuracy and stability aren't at a significant computational expense. Overall, our results bolster the efficacy of the proposed pruning method, blending top-notch performance with reasonable efficiency metrics.

6.3.3 Fine-tuning After Prune vs. Direct Reduced Clauses. Comparing ***AdaTM+Prune*** and AdaTM+Reduced Clause, the dynamic adaptation of the former offers clear performance advantages. Across all datasets, ***AdaTM+Prune*** consistently surpasses AdaTM+Reduced Clause in average accuracy. For example, on MNIST, ***AdaTM+Prune*** achieves **92.54%** accuracy, contrasting with AdaTM+Reduced Clause **90.46%**. This superior performance is reflected in the forgetting measure, with ***AdaTM+Prune*** demonstrating greater stability, evident from its lower forgetting rates in AudioMNIST and MNIST. Though AdaTM+Reduced Clause marginally outperforms in computational efficiency, such as a latency of **192.7**s versus **188.7**s on PAMAP2, and **38.1** MB memory use against ***AdaTM+Prune***'s **42.5** MB on MNIST, these gains don't offset the accuracy loss.

The data suggests starting with a wider clause set, then pruning and fine-tuning is more effective. This approach, being more selective, hastens convergence to better outcomes. It emphasizes the value of an initial expansion followed by systematic reduction and calibration over starting with a limited clause set.

In the end, we can analyse the task-wise performance for all four configurations. As we can observe from line trends, ***AdaTM*** usually starts strong and often retains top performance. However, ***AdaTM+Prune*** closely follows, indicating the effectiveness of the pruning method. In most tasks, AdaTM+Random Prune and AdaTM +Reduced Clause fall behind the other two methods, reinforcing the superiority of the developed pruning technique over random pruning or starting with fewer clauses.

7 CONCLUSION

Our research highlights the promising capabilities of the proposed ***AdaTM*** framework. A key attribute of the ***AdaTM*** is we are the first to enable this logic-based learning model the dynamic architectural adaptability, designed to integrate new learning tasks without the need for the resource-intensive recalibrations found in standard neural networks. Other contributions include the integration of a unique pruning mechanism and empirical evidence supporting its robust performance. Our comprehensive assessments show that the ***AdaTM*** not only matches but often surpasses the performance benchmarks of existing neural network models, covering crucial metrics such as average accuracy, forgetting measure, processing latency, and runtime memory. While popular datasets like CIFAR-10 serve as standard benchmarks, the real significance of our research transcends specific datasets. It's about shaping a more efficient and scalable framework for continual learning in diverse, real-world scenarios. Note that making the Tsetlin Machine work for more complex datasets such as CIFAR-10 and CIFAR-100 is an active area of research which will require fundamental changes to the architecture and modelling processes of TM. This is an orthogonal area of research which we would like to do in the future, further paving the way for a sustainable continual learning paradigm.

Looking ahead, we recognize the need to refine the Tsetlin Machine's training approach to reduce latency on the PAMAP2 dataset. Implementing batch processing is on our near-future roadmap to enhance latency while maintaining the advantages of ***AdaTM***. Additionally, we plan to explore adaptive pruning mechanisms, understand interactions between continual learning tasks, and possibly integrate neural-symbolic approaches with ***AdaTM***. As AI and continual learning evolve, systems like ***AdaTM*** are set to play a crucial role in establishing performance standards in edge computing.

ACKNOWLEDGMENTS

This work was supported by EPSRC Grant EP/X01200X/1.

REFERENCES

[1] Abu Bakar, Tousif Rahman, Rishad Shafik, Fahim Kawsar, and Alessandro Montanari. 2022. Adaptive Intelligence for Batteryless Sensors Using Software-Accelerated Tsetlin Machines. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. 236–249.

[2] Chaitanya Baweja, Ben Glocker, and K Kamnitsas. 2018. Towards continual learning in medical imaging. *ArXiv* abs/1811.02496 (2018).

[3] Geir Thore Berge, Ole-Christoffer Granmo, Tor Oddbjørn Tveit, Morten Goodwin, Lei Jiao, and Bernt Viggo Matheussen. 2019. Using the Tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications. *IEEE Access* 7 (2019), 115134–115146.

[4] Arslan Chaudhry, MarcAurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. [n. d.]. On Efficient Lifelong Learning with A-GEM. ([n. d.]).

[5] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486* (2019).

[6] K Darshana Abeyrathna, Ole-Christoffer Granmo, Xuan Zhang, Lei Jiao, and Morten Goodwin. 2020. The regression Tsetlin machine: a novel approach to interpretable nonlinear regression. *Philosophical Transactions of the Royal Society A* 378, 2164 (2020), 20190165.

[7] Sondre Glimsdal and Ole-Christoffer Granmo. 2021. Coalesced multi-output tsetlin machines with clause sharing. *arXiv preprint arXiv:2108.07594* (2021).

[8] Ole-Christoffer Granmo. 2018. The Tsetlin Machine–A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic. *arXiv preprint arXiv:1804.01508* (2018).

[9] Ole-Christoffer Granmo, Sondre Glimsdal, Lei Jiao, Morten Goodwin, Christian W Omlin, and Geir Thore Berge. 2019. The convolutional Tsetlin machine. *arXiv preprint arXiv:1905.09688* (2019).

[10] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. 2016. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122* (2016).

[11] Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving Forgetting Prevention and Knowledge Transfer in Continual Learning. *ArXiv* abs/2112.02706 (2021).

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[13] Young D Kwon, Jagmohan Chauhan, Abhishek Kumar, Pan Hui HKUST, and Cecilia Mascolo. 2021. Exploring system performance of continual learning for mobile and embedded sensing applications. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 319–332.

[14] Jie Lei, Tousif Rahman, Rishad Shafik, Adrian Wheeldon, Alex Yakovlev, Ole-Christoffer Granmo, Fahim Kawsar, and Akhil Mathur. 2021. Low-power audio keyword spotting using Tsetlin machines. *Journal of Low Power Electronics and Applications* 11, 2 (2021), 18.

[15] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[16] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[17] Xinyue Ma, Suyeon Jeong, Minjia Zhang, Di Wang, Jonghyun Choi, and Myeongjae Jeon. 2023. Cost-effective On-device Continual Learning over Memory Hierarchy with Miro. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[18] Sidharth Maheshwari, Tousif Rahman, Alex Yakovlev, Ashur Rafiev, Lei Jiao, Ole-Christoffer Granmo, et al. 2023. REDRESS: Generating Compressed Models for Edge Inference Using Tsetlin Machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[19] Cuong V Nguyen, A Achille, Michael Lam, Tal Hassner, V Mahadevan, and Stefano Soatto. 2019. Toward Understanding Catastrophic Forgetting in Continual Learning. *ArXiv* abs/1908.01091 (2019).

[20] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 524–540.

[21] Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. 2021. Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369* (2021).

[22] Saeed Rahimi Gorji, Ole-Christoffer Granmo, Adrian Phoulady, and Morten Goodwin. 2019. A Tsetlin machine with multigranular clauses. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 146–151.

[23] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[24] Rupsa Saha, Ole-Christoffer Granmo, and Morten Goodwin. 2020. Mining interpretable rules for sentiment and semantic relation analysis using tsetlin machines. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 67–78.

[25] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems* 30 (2017).

[26] Gido M Van De Ven, Zhe Li, and Andreas S Tolias. 2021. Class-incremental learning with generative classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3611–3620.

[27] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three types of incremental learning. *Nature Machine Intelligence* 4, 12 (2022), 1185–1197.

[28] Zifeng Wang, Zheng Zhan, Yifan Gong, Geng Yuan, Wei Niu, Tong Jian, Bin Ren, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. 2022. SparCL: Sparse continual learning on the edge. *Advances in Neural Information Processing Systems* 35 (2022), 20366–20380.

[29] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*. PMLR, 3987–3995.