TAME: Task Agnostic Continual Learning using Multiple Experts

Haoran Zhu[1*] Maryam Majzoubi [2*] Arihant Jain [1*] Anna Choromanska [1]
[1]New York University [2]Google
{hz1922, aj2622, ac5455}@nyu.edu maryam.majzoubi@gmail.com

Abstract

*The goal of lifelong learning is to continuously learn from non-stationary distributions, where the non-stationarity is typically imposed by a sequence of distinct tasks. Prior works have mostly considered idealistic settings, where the identity of tasks is known at least at training. In this paper we focus on a fundamentally harder, so-called task-agnostic, setting where the task identities are not known and the learning machine needs to infer them from the observations. Our algorithm, which we call TAME (**T**ask-**A**gnostic continual learning using **M**ultiple **E**xperts), automatically detects the shift in data distributions and switches between task expert networks in an online manner. At training, the strategy for switching between tasks hinges on an extremely simple observation that for each new coming task there occurs a statistically-significant deviation in the value of the loss function that marks the onset of this new task. At inference, the switching between experts is governed by the selector network that forwards the test sample to its relevant expert network. The selector network is trained on a small subset of data drawn uniformly at random. We control the growth of the task expert networks as well as selector network by employing pruning. Our experimental results show the efficacy of our approach on benchmark continual learning data sets, outperforming the previous task-agnostic methods and even the techniques that admit task identities at both training and testing, while at the same time using a comparable model size.*

1. Introduction

Learning agents deployed in real world applications are exposed to a continuous stream of incrementally available information usually from non-stationary data distributions. The agent is required to adaptively learn over time by accommodating new experience while preserving previous learned knowledge. This is referred to as lifelong or continual learning, which has been a long-established challenge in artificial intelligence, including deep learning [8, 30, 39].

In the commonly considered scenario of lifelong learning, where the tasks come sequentially and each task is a sequence of events from the same distribution, one of the main challenges is to overcome catastrophic forgetting, where training the model on a new task interfere with the previously acquired knowledge and leads to the performance deterioration on the previously seen tasks. Deep neural networks generally perform well on classification tasks, but they heavily rely on having i.i.d. data samples drawn from stationary distribution during training time [7, 17, 33]. In the case of sequential tasks, their performance significantly deteriorates when learning new coming tasks [14, 24–26, 30].

A number of approaches have been suggested in the literature to deal with catastrophic forgetting. Some works [11, 40] provide systematic categorization of the continual learning frameworks and identify three different scenarios: incremental task learning, incremental domain learning, and incremental class learning, where their differences stem from the availability of task labels at testing and number of output heads. In incremental class and domain learning the task identity is not known during the testing. All of these scenarios however are based on the assumption that the task labels are known at the training phase. This assumption is limiting in practical real-world applications, where the agent needs to learn in a more challenging task-agnostic setting [18, 31, 32, 44]. In this learning setting the task identities are not available both at training and inference times. The literature started exploring this setting very recently and this setting is in the central focus of our paper.

In this work, we present an approach for handling task-agnostic continual learning inspired by the older approaches dedicated to learning non-stationary sequences based on experts advice [10, 27, 28], which explore and exploit the intermittent switches between distinct stationary processes. In these approaches the learner can make predictions on the basis of a fixed set of experts. Since the learner does not know the mechanisms by which the experts arrive at their predictions, it ought to exploit the information obtained by observing the losses of the experts. Based on the experts' losses it weights the experts to attenuate poor performers and emphasize the good ones, and forms the final predic-

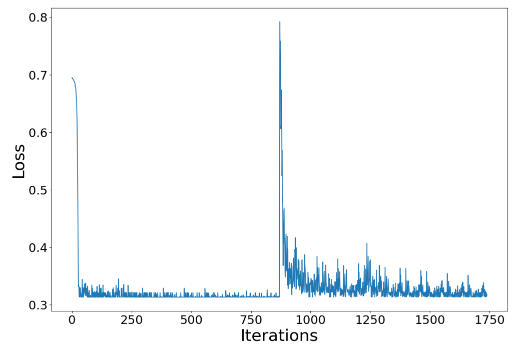


Figure 1. Deviation of the value of loss function of the expert when the task is switched

tion as the weighted sum of experts' predictions. Thus the learner needs to identify the best expert at each time and switch between the experts when the task switches occur. In the aforementioned works, the weights over the experts are the only carriers of the memory of previous experiences. Also, the discussed methods rely on the assumption that the number of experts/tasks are known in advance. Finally, these methods do not consider a separate train and test phase, but rather their optimization process is focused on minimizing the regret, which is the difference between the cumulative loss of the algorithm and the loss of the best method in the same class, chosen in hindsight (hindsight refers to full knowledge of the sequence to be predicted). Minimizing the regret however is not equivalent to counter-acting catastrophic forgetting since previous tasks that present little relevance to the currently learned ones are gradually being overwritten in memory. These methods thus are not directly applicable to the continual learning setting.

Motivated by having a set of experts representing a sequence of tasks, where each task is essentially a stationary segment of a longer non-stationary distribution,

we propose a learning system that initially starts with one expert and gradually adds or switches between experts when the tasks change. During the online training phase our algorithm automatically identifies when the task switches and either selects or creates the best expert for a new task, depending whether this task was seen before or not. The detection of task switches relies on the statistically significant deviation of the loss function value of the current expert, which marks the onset of the new task (see Figure 1). Similarly, the determination whether the task was seen before or not relies on the behavior of the per-expert loss functions (if the deviation of all per-expert loss values are high, new expert is created to represent the current task). Such simple detection mechanism is inspired by the classical experts advise literature discussed in the previous paragraph, where switching between experts is governed by the values of the loss functions of the experts. Moreover, we introduce a *selector* network which predicts the task identity of the samples at inference time. The selector network is trained on a small subset of training examples that were sampled uniformly at random from different tasks during the learning process. Despite the simplicity of our approach, it leads to a task-agnostic continual learning algorithm that compares favorably to existing methods and proves that a rich historical literature on online processing of non-stationary sequences can provide useful signal processing tools for addressing challenges in modern continual learning discipline.

The rest of the paper is organized as follows: Section 2 discusses the most relevant work. Section 3 introduces our algorithm which we call TAME: **T**ask **A**gnostic continual learning using **M**ultiple **E**xperts. Section 4 reports empirical results on benchmark continual learning data sets, and finally Section 5 concludes the paper.

2. Related Work

In recent years, there has been a plethora of techniques proposed for continual learning that mitigate the catastrophic forgetting problem in deep neural networks. The existing approaches can be divided into three categories: i) complementary learning systems and memory replay methods, ii) regularization-based methods, and iii) dynamic architecture methods. These techniques are not dedicated to the task-agnostic scenario since they assume the identity of the tasks are provided at least during the training phase. On the other hand, more challenging task-agnostic continual learning setting was addressed only recently in a handful of papers. We review them first since our paper considers the same setting. For completeness we also discuss the most relevant works from the broad continual learning literature and refer the reader to a survey paper [30] that provides a more comprehensive review of these approaches.

Task-Agnostic Continual Learning In the context of supervised learning setting, which is of central focus to this paper, one of the first methods addressing task-agnostic continual learning is the Bayesian Gradient Descent algorithm, popularly known as BGD [44]. This approach is based on an online version of variational Bayes and proposes a Bayesian learning update rule for the mean and variance of each parameter. As all Bayesian approaches, this method counter-acts catastrophic forgetting by using the posterior distribution of the parameters for the previous task as a prior for the new task. BGD obtains the most promising empirical results in the setting, where the method relies on the so-called "label trick" where the task identity is inferred from the class label. Label trick however breaks the task-agnostic assumption. Another approach called iTAML [31] proposes to use meta-learning to maintain a set of generalized parameters that represent all tasks. When presented with a continuum of data at inference, the model automatically identifies the task and quickly adapts to it with just a single update. However at training the inner loop of their algorithm, which generates task-specific models for each task that are then combined in the outer loop to form a more generic model, requires the knowledge of task label. At inference, the task is predicted using generalized model parameters. Specifically, for each sample in the continuum, the outcome of the general model is obtained and a maximum response per task is recorded. An average of the maximum responses per task is used as the task score. A task with a maximum score is finally predicted. iTAML counteracts catastrophic forgetting by keeping a memory buffer of samples from different tasks and using it to fine-tune generalized parameters representing all tasks to a currently seen one. This method is not task-agnostic, since it requires task labels at training, though the authors categorize their method as task-agnostic. CN-DPM [18] is an expansion-based method that eliminates catastrophic forgetting by allocating new resources to learn new data. They formulate the task-agnostic continual learning problem as an online variational inference of Dirichlet process mixture models consisting of a set of neural experts. Each expert is in charge of a subset of the data. Each expert is associated with a discriminative model (classifier) and a generative model (density estimator). For a new sample, they first decide whether the sample should be assigned to an existing expert or a new expert should be created for it. This is done by computing the responsibility scores of the experts for the considered sample and is supported by a short-term memory (STM) collecting sufficient data. Specifically, when a data point is classified as new, they store it to the STM. Once the STM reaches its maximum capacity, they train a new expert with the data in the STM. Another technique for task-agnosic continual learning, known as HCL [15], models the distribution of each task and each class with a normalizing flow model. For task identification, they use the state-of-the-art anomaly detection techniques based on measuring the typicality of the model's statistics. For avoiding catastrophic forgetting they use a combination of generative replay and a functional regularization technique.

In the context of unsupervised learning setting, VASE method [1] addresses representation learning from piecewise stationary visual data based on a variational autoencoder with shared embeddings. The emphasis of this work is put on learning shared representations across domains. The method automatically detects shifts in the training data distribution and uses this information to allocate spare latent capacity to novel data set-specific disentangled representations, while reusing previously acquired representations of latent dimensions where applicable. Authors represent data sets using a set of data generative factors, where two data sets may use the same generative factors but render them differently, or they may use a different subset of factors altogether. They next determine whether the average reconstruction error of the relevant generative factors for the current data matches the previous data sets by a threshold or not using Minimum Description Length principle. Allocating spare representational capacity to new knowledge protects previously learnt representations from catastrophic forgetting. Another technique called CURL [32] learns a task-specific representation on top of a larger set of shared parameters while dynamically expanding model capacity to capture new tasks. The method represents tasks using a mixture of Gaussians and expands the model as needed, by maintaining a small set of poorly-modelled samples and then initialising and fitting a new mixture component to this set when it reaches a critical size. The method also relies on replay generative models to alleviate catastrophic forgetting.

Non Task-Agnostic Continual Learning First family of non task-agnostic continual learning techniques consists of complementary learning systems and memory replay methods. They rely on replaying selected samples from the prior tasks. These samples are incorporated into the current learning process so that at each step the model is trained on a mixture of samples from a new task as well as a small subset of samples from the previously seen tasks. Some techniques focus on efficiently selecting and storing prior experiences through different selection strategies [4, 13]. Other approaches, e.g. GEM [21], A-GEM [6], and MER [33] focus on favoring positive backward transfer to previous tasks. Finally, there are deep generative replay approaches [34, 36] that substitute the replay memory buffer with a generative model to learn data distribution from previous tasks and generate samples accordingly when learning a new task. Another family of techniques, known as regularization-based methods, enforce a constraint on the parameter update of the neural network, usually by adding a regularization term to the objective function. This term penalizes the change in the model parameters when the new task is observed and assures they stay close to the parameters learned on the previous tasks. Among these techniques, we identify a few famous algorithms such as EWC [16], SI [43], MAS [3], and RWALK [5] that introduce different notions of the importance of synapses or parameters and penalizes changes to high importance parameters, as well as the LwF [20] method that can be seen as a combination of knowledge distillation and fine-tuning. Finally, the last family of techniques are the dynamic architecture methods that expand the architecture of the network by allocating additional resources, i.e., neurons or layers, to new tasks which is usually accompanied by additional parameter pruning and masking. This family consists of such techniques as expert-gate method [2], progressive networks [35], dynam-

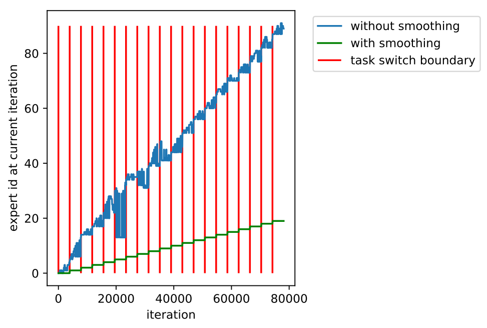


Figure 2. The effect of smoothing

ically expandable network [42], learn-to-grow method [19], Packnet [22], Piggyback [23], and hybrid schemes [12]. The last three techniques rely on network quantization and pruning to better control the complexity and size of the model.

3. TAME Algorithm

In this section we describe the proposed algorithm `TAME`. Let $\mathcal{T}$ denote the set of all tasks. Each example is drawn i.i.d. from an unknown distribution P_t of the corresponding task, i.e. $(x_i^t, y_i^t) \sim P_t$. The tasks come in a sequential manner. We consider a scenario where the task identity as well as the number of tasks is not known. The goal is to learn these tasks sequentially without catastrophic forgetting by automatically identifying the task identities both at the training as well as the testing time.

`TAME` is based on using multiple *task expert networks*, where each expert network is associated with one task. At training, the algorithm automatically detects the shift in the data distribution in an online manner and switches between the existing experts or adds more experts if necessary. The strategy for the task switch detection relies on the statistically-significant deviation in the values of the loss function. At testing, we have an additional *selector* network that automatically forwards each sample to its relevant expert. This selector network is trained on a small subset of samples that are drawn uniformly at random from the sequence of samples from all tasks.

The pseudo-code of our algorithm is captured in Algorithm 1. We initially start with one expert network and gradually add more networks as needed. At each step of time we only have one active expert that is being trained on incoming data. We observe the value of the loss function of the current active expert. In order to smooth-out short term variations and highlight long-term patterns, a *smoothed* version of the loss is calculated, through an exponentially weighted moving average (EWMA) [41]. EWMA is a first-order infinite impulse response filter that applies weighting factors which decrease exponentially (never reaching zero). This is used to filter out higher frequency components that has no specific connection to shift in data distribution. The coefficient α is a constant smoothing factor between $[0, 1]$ that governs the amount of smoothing. Higher α diminishes previous observations faster. Figure 2 justifies the need for loss smoothing by comparing the performance of the proposed algorithm with and without smoothing. Smoothed loss helps avoiding false positives in detecting task switches, and thus prevents creating unnecessary experts, while at the same time it enables to maintain high detection accuracy.

Furthermore, we calculate a loss threshold value for each expert network. We assume a normal distribution for the value of the loss function. We set the significance threshold at three standard deviations above the mean. As shown in procedure `Get_threshold` the mean and standard deviation are calculated over a moving window of size W_{th} of the previously observed data.

In lines $30 - 43$ we compare the smoothed loss with the threshold value of the current active expert and if it is above the threshold we search over all other existing experts and choose the one that meets the threshold requirement. If no such expert network is found, it means that no expert well-represents the currently seen data and thus a new expert is added to the model and gets activated (see procedure `Add_expert`). Next, the selected network is trained on the input data.

We need to also train the selector network to switch between experts at inference. For this purpose, we have a buffer in the form of a priority queue with a fixed capacity C_s that is much smaller than the total number of samples. In lines $46 - 50$, we randomly sub-sample data to keep it in the buffer in an online fashion. The label for each sample is the current expert id which corresponds to the task identity that we inferred from the data. The selector network is trained on the samples from this buffer and later used at inference time to automatically distinguish the task label and sends the test data to the corresponding expert network.

To reduce the size of the experts and selector network, we perform network pruning. Typically, after pruning, the model needs to be retrained to prevent drastic drop in performance. To enable retraining of experts we introduce set of buffers $buffers_{prune}$ to store samples for each expert (task). Each buffer is implemented as a priority queue with a fixed capacity C_p. When new expert is created, a buffer for that expert is added to the set. In lines $46 - 50$, we randomly sub-sample data and fill in the buffer in an online fashion. Thus, for each task we only keep a fixed amount of randomly selected samples. After training for all tasks is done, in line 52, we prune and retrain the selector network using buffer $buffer_{selector}$. In lines $53 - 55$, we prune and retrain each expert using the corresponding buffer from $buffers_{prune}$.

Algorithm 1 `TAME`: Task Agnostic continual learning using Multiple Experts

```
Require: Data: ${(x, y)}$, Threshold window size: $W_{th}$, Smoothing factor: $\alpha$,
    Buffer capacity for training selector net: $C_s$, Buffer capacity for retraining after
    pruning: $C_p$

procedure: Add_expert ()
Input: $experts$, $N_e$
    Initialize a new $expert$ network.
    Initialize smoothed loss of the expert $expert.L_s \leftarrow None$
    Initialize $expert.deque$ with maximum capacity equal to $W_{th}$
    $expert.id = N_e$
    $experts$.add($expert$)
    $N_e$ += 1
Return $expert$

procedure: Get_threshold ()
Input: $expert.deque$
    $\mu$ = MEAN($expert.deque$)
    $\sigma$ = STD($expert.deque$)
Return $(\mu + 3 * \sigma)$


Initialize: buffer $buffer_{selector}$ with $C_s$ capacity; Buffers for pruning
    $buffers_{prune} \leftarrow$ [] with $C_p$ capacity for all incoming tasks; number of
    expert: $N_e \leftarrow 0$; $experts \leftarrow$ []; current task id $T_{id} \leftarrow 0$;
$expert_c$ = Add_expert($experts, N_e$)
$T_{id} = 1$
$buffers_{prune}[T_{id}] \leftarrow$ a priority queue with $C_p$ capacity (initialize buffer
    for the first task)
while Incoming Data do
    $L_c$ = loss of the current expert $expert_c$ on input ${(x, y)}$
    if $expert_c.L_s == None$ then
        $expert_c.L_s = L_c$
    else
        $expert_c.L_s = \alpha * L_c + (1 - \alpha) * expert_c.L_s$
    end if
    if $expert_c.L_s >$ Get_threshold($expert_c.deque$) then
        $expert_p = None$
        $T_{id} \leftarrow 0$
        for $e$ in $experts$ do
            $T_{id} = T_{id} + 1$
            if $\alpha * L_e + (1 - \alpha) * e.L_s <$ Get_threshold($e.deque$) then
                $expert_p = e$; break
            end if
        end for
        if $expert_p == None$ then
            $expert_c$ = Add_expert($experts, N_e$)
            $buffers_{prune}[T_{id} + 1] \leftarrow$ a priority with $C_p$ capacity (initialize
    buffer for the new task)
        end if
    end if
    Train $expert_c$ on batch of data ${(x, y)}$ and update its $deque$.

    for $(x_i, -)$ in ${(x, y)}$ do
        $priority = \mathcal{N}(0, 1)$
        $buffer_{selector}$.add(key:$priority$, value:$(x, expert_c.id)$)
        $buffers_{prune}[T_{id}]$.add(key:$priority$, value:$(x, y)$)
    end for
end while
Train and prune $selector$ network on samples in $buffer_{selector}$
for i in ${1, 2, \ldots, N_e}$ do
    Prune and retrain $experts[i]$ using buffer $buffers_{prune}[i]$ stored for
    $experts[i]$
end for
```

4. Experiments

In this section we evaluate `TAME` on benchmark continual learning data sets and compare with other state-of-the-art methods, namely previously proposed task-agnostic methods: BGD [44], iTAML [31], HCL [15], and CN-DPM [18], as well as techniques that are not task-agnostic but are dedicated to the continual learning setting, such as DEN [42], EWC [16], SI [43], A-GEM [6], and RWALK [5]. For evaluating the performance of the competitor algorithms we use open-source implementations, when available[1] [2] [3] [4] [5].

4.1. Data sets

We use standard continual learning data sets: (1) **Permuted MNIST**, where a set of tasks is created by using a different random permutation of MNIST pixels. We generated a set of 20 data sets accordingly that correspond to 20 tasks. (2) **Split MNIST**, where a set of tasks is constructed by taking pairs of digits from the original MNIST data set, i.e. $\mathcal{T} = \{\{0, 1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}, \{8, 9\}\}$. (3) **Split CIFAR-100 (20)**, where the original CIFAR-100 is divided into 20 disjoint subsets, each containing 5 class labels, i.e. $\mathcal{T} = \{\{0 - 4\}, \{5 - 9\}, \ldots, \{95 - 99\}\}$. For comparison with HCL method, we use additional data sets: (4) **Split CIFAR-100 (10)**, where the original CIFAR-100 is divided into 10 disjoint subsets, each containing 10 class labels, i.e. $\mathcal{T} = \{\{0 - 9\}, \{10 - 9\}, \ldots, \{90 - 99\}\}$. (5) **Split CIFAR-10 (5)**, where the original CIFAR-10 is divided into 5 disjoint subsets each, containing 2 class labels, i.e. $\mathcal{T} = \{\{0 - 1\}, \{2 - 3\}, \ldots, \{8 - 9\}\}$. (6) **SVHN-MNIST**, that combines SVHN[29] and MNIST data sets in a way that SVNH is the first task and MNIST is the second one. (7) **MNIST-SVHN** where MNIST is the first task and SVNH is the second one.

We use the following size of the images in our data sets, i.e. $1 \times 28 \times 28$ for Split MNIST and Permuted MNIST, $3 \times 32 \times 32$ for Split CIFAR-100 (10), Split CIFAR-100 (20), and Split CIFAR-10 (5). For MNIST-SVHN and SVHN-MNIST, we upscale size of the images in MNIST to $3 \times 32 \times 32$ and use the original image size of $3 \times 32 \times 32$ for SVHN data set. We normalize Split MNIST and Permuted MNIST data sets by mean 0.1307 and standard deviation 0.3081, Split CIFAR-100 (10) and Split CIFAR-100 (20) by mean (0.5071, 0.4867, 0.4408) and standard deviation (0.2675, 0.2565, 0.2761), and Split CIFAR-10 (5) by mean (0.5, 0.5, 0.5) and standard deviation (0.5, 0.5, 0.5). For SVHN-MNIST and MNIST-SVHN data sets, we use mean mean 0.1307 and standard deviation 0.3081 for MNIST, and mean (0.5, 0.5, 0.5) and standard deviation (0.5, 0.5, 0.5) for SVHN. For Split CIFAR-100 (10) and Split CIFAR-100 (20), we use additional data augmentation techniques such as random crop, flip, and rotations.

[1] https://github.com/facebookresearch/agem(A-GEM, SI, RWALK, and EWC)
[2] https://github.com/jaehong31/DEN (DEN)
[3] https://github.com/igolan/bgd/ (BGD)
[4] https://github.com/brjathu/iTAML (iTAML)
[5] https://github.com/soochan-lee/CN-DPM (CN-DPM)

4.2. Networks Architecture

For all methods, except HCL, in case of Permuted-MNIST and Split MNIST data sets we use a network with 2 convolutional layers, which are followed by the usual ReLU activation function, the max pooling operation, and fully connected layers for the task expert networks and we use a 2-layer MLP for the selector network. For Split CIFAR-100 (20), we use a slightly modified version of VGG11 [37] architecture for the task expert networks and a pre-trained ResNet18 [9] for the selector network. The modification of VGG accommodates having 5 outputs. To compare with HCL, we use the above mentioned 2-layer convolutional network for SVHN-MNIST, MNIST-SVHN, and Split MNIST data sets. For Split CIFAR-10 (5) and Split CIFAR-100 (20) we use EfficientNet [38] model pretrained on ImageNet for the expert network and a pre-trained ResNet-18 for the selector network. Note that the choice of architectures we make is done on purpose to stay aligned with the architectures used by the competitor methods. In order to prevent the sudden jump of the loss function value during an initial stage of the training we add a sigmoid layer on the output of each model.

In competitor algorithms, for Permuted and Split MNIST data sets, we experimented with both the aforementioned 2-layer convolutional network as well as a 2-layer MLP and chose the best results. For Split CIFAR-100 (20), we used the VGG architecture except for BGD and CN-DPM. In the case of BGD, the VGG architecture led to the loss function divergence and unstable results, thus we used the architecture suggested by the authors in their paper for this data set. For CN-DPM the performance was worse when we used the VGG architecture so we report the results given in the CN-DPM's original paper. Furthermore, DEN implementation was not available for convolutional networks, so we were not able to test it for Split CIFAR-100 (20). For HCL, we report the results given in their paper on all listed data sets since their code is not publicly available.

4.3. Training Details

We train the models using SGD optimizer with learning rate equal to 0.1, Nesterov momentum 0.9, and weight decay $5e-4$ and the batch size of to 128 for all data sets. We trained for 10 epochs on each task from Permuted MNIST and Split MNIST data sets and for 200 epochs for each task from Split CIFAR-100 (20). For Split CIFAR-100 (20) we drop the learning rate by the factor of 5 at the 60th, 120th, and 160th epochs. We also apply a warm-up training in the first epoch to prevent the network divergence early in the training. We train 90 epochs for SVHN-MNIST and MNIST-SVHN, 15 epochs for Split CIFAR-10 (5) and Split CIFAR-100 (20). Finally, we use L1 unstructured pruning to reduce the model size for each expert and the expect selector network. We retrain each expert after pruning with the sub-sampled data stored during training. We use SGD optimizer with learning rate equal to 0.1 and weight decay $1e-4$.

4.4. Hyperparameters

We next describe the values of the hyperparameters that are specific to our algorithm. The hyperparameters settings used in the experimets are summarized in Table 1. In all experiments we use the same window size W_{th} equal to 100 and loss smoothing factor α of 0.2.

For competitor algorithms we either used hyperparameter settings suggested by the authors or performed a parameter search. The scope of the hyperparameter search for the regularization-based methods and the training settings for A-GEM are shown in the supplementary materials.

4.5. Results

The metric we use for our evaluation is the average accuracy measured on the test set. The average accuracy is defined as $ACC = 1/T \sum_{i=1}^{T} R_{T,i}$, where $R_{T,i}$ is the classification accuracy of the model on task i, and T is the number of tasks.

In Table 2 we compare the average accuracy obtained by `TAME` and other algorithms. For BGD method, we do not use any "label trick" approaches and thus run it in a purely task-agnostic setting, same as `TAME`, HCL, and CN-DPM. For iTAML, the task identity is known during training, thus it is not a task-agnostic method under our standards, however since during inference they do not rely on the task identity we kept this method as our competitor. All the other algorithms have access to task descriptors both at training and testing. `TAME` achieves the highest accuracy and the smallest model size among all considered algorithms and data sets. Note that `TAME` also outperforms continual learning methods that have access to task labels at training and/or testing. In Table 3 we compare the performance of `TAME` with HCL. Our method outperforms this approach as well.

We compare the behaviour of the loss function (left) and its smoothed version (right) for each of the experts at training, where the experts are added sequentially as new tasks arrive. The results demonstrate that the smoothed loss more effectively reduces short-term variations and emphasizes long-term patterns. For more details, please see figure in the supplementary materials.

In Figures 3a-3c we illustrate the behaviour of average accuracy while model is trained on the sequence of tasks. The proposed algorithm, `TAME`, has the least drop in performance when adding more tasks among all considered methods and datasets.

In Figure 3d we demonstrate the effect of the buffer capacity on the accuracy of the selector network for Split CIFAR-100(20) data set. Note that the selector network accuracy depends on the similarity of tasks. For instance if

Table 1. Hyperparameter settings used for TAME

TAME	Permuted MNIST	Split MNIST SVHN-MNIST MNIST-SVHN	Split CIFAR-100 (20) Split CIFAR-100 (10) Split CIFAR-10 (5)
Threshold window W_{th}	100	100	100
Smoothing factor α	0.2	0.2	0.2
Buffer capacity C_s	5000	2500	7500
Buffer capacity C_p	6000	1000	200
Expert pruning rate (%)	98	98	98
Expert selector pruning rate (%)	50	50	50

Table 2. Average Accuracy (%) obtained by TAME and other algorithms for Permuted MNIST, Split MNIST, and Split CIFAR-100

Data sets (#tasks)	Permuted MNIST (20)		Split MNIST (5)		Split CIFAR-100 (20)	
	Acc. (%)	Param.	Acc. (%)	Param.	Acc. (%)	Param.
task identity is known during training and testing						
EWC	54.81	61.7K	98.18	61.7K	32.78	9.23M
SI	81.31	61.7K	94.85	61.7K	30.28	9.23M
A-GEM	79.61	61.7K	97.72	61.7K	43.57	9.23M
RWALK	46.23	61.7K	96.84	61.7K	31.13	9.23M
DEN	83.61	120.2K	95.51	120.2K	NA	NA
task identity is known during training, but not during testing						
iTAML	NA	NA	97.95	61.7K	54.55	9.23M
task-agnostic (task id is not known during both training and testing)						
BGD (without label trick)	79.15	61.7K	19.00	61.7K	3.77	9.23M
CN-DPM	14.99	616.1K	94.19	746.8K	20.45	19.20M
HCL	NA	NA	90.89	NA	NA	NA
TAME	**87.32**	**55.53K**	**98.63**	**37.02K**	**62.39**	**9.02M**

Table 3. Average Accuracy (%) obtained by TAME and HCL

Data sets (#tasks)	SVHN-MNIST Acc. (%)	MNIST-SVHN Acc. (%)	Split MNIST (5) Acc. (%)	Split CIFAR-10 (5) Acc. (%)	Split CIFAR-100 (10) Acc. (%)
HCL-FR	96.38	95.62	90.89	89.44	59.66
HCL-GR	93.84	96.04	84.65	80.29	51.64
TAME	**97.45**	**97.63**	**98.63**	**91.32**	**61.06**

Table 4. Size C_p of buffer used for retraining experts after pruning versus average accuracy for Split CIFAR-100 (20), Split MNIST, and Permuted MNIST data sets

Data sets	50	100	200	500	1000	2000	3000	6000
Split CIFAR-100 (20)	56.12	57.86	62.39	63.47	64.41	/	/	/
Split MNIST	97.80	98.20	98.22	98.38	98.63	98.38	/	/
Permuted MNIST	/	62.55	69.95	76.83	79.93	83.41	84.94	87.32

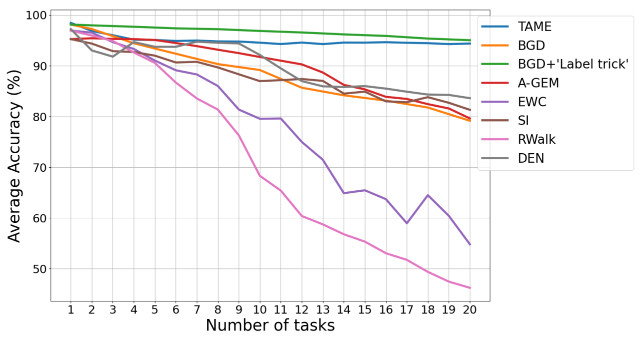
we use 20 super-classes from Split CIFAR-100(20), where similar labels are grouped together, the selector accuracy increases from $\sim 62\%$ to $\sim 79\%$.

We also show the effect of the buffer capacity for pruning in Table 4. For Permuted MNIST and Split MNIST, even a small size of buffer yield a good average accuracy.

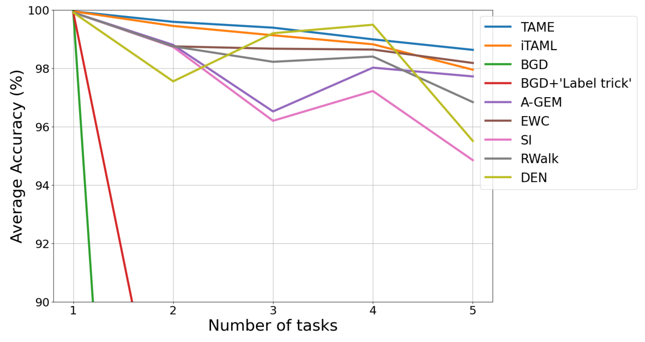
We also present in Figure 4 the ability of the model to use the existing experts when a previously seen task appears again in the sequence. For instance in Figure 4b when task 2 appears for a second time in the sequence of $\mathcal{T} = \{t_1, t_2, t_3, t_2, t_4\}$ the algorithm appropriately switches to the already existing expert 2 rather than instantiating a new expert.

5. Conclusion and Discussion

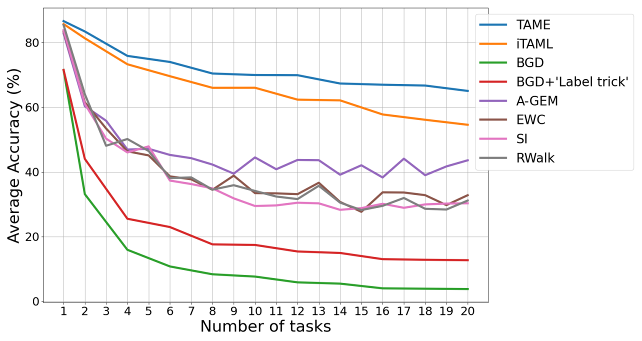
This paper addresses a more challenging scenario of continual learning - a task-agnostic setting, where the model is not provided with task descriptors during training or testing



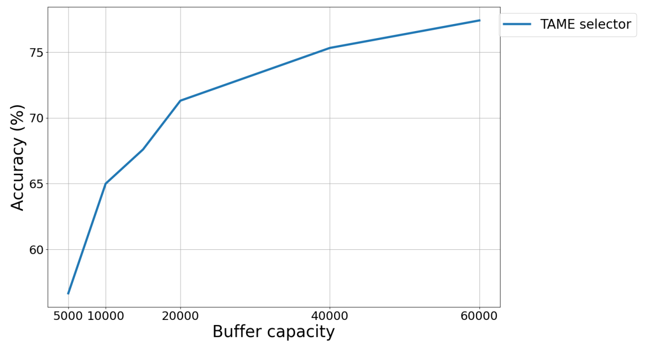
(a) Permuted MNIST



(b) Split MNIST

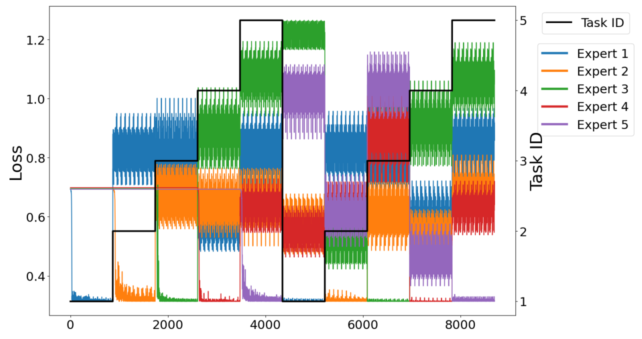


(c) Split CIFAR-100

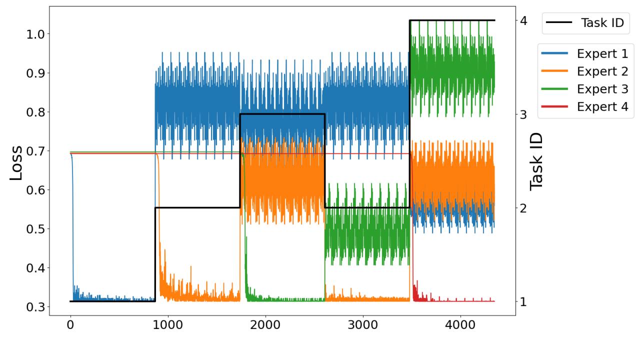


(d) `TAME` selector, Split CIFAR-100

Figure 3. (**a, b, c**) Average Accuracy versus the number of tasks for various data sets. (**d**) Effect of the buffer capacity on the accuracy of the selector network in `TAME` for Split CIFAR-100 data set for 20 tasks.



(a) Loss value vs. iterations



(b) Loss value vs. iterations

Figure 4. The value of loss function of different task expert networks during training on Split MNIST data set. Shown for two sequences of tasks: (**a**) $\mathcal{T} = \{t_1, \dots, t_5, t_1, \dots, t_5\}$ (**b**) $\mathcal{T} = \{t_1, t_2, t_3, t_2, t_4\}$. The algorithm switches to existing experts when a previously seen task occurs later in the sequence

time. We devise a new continual learning algorithm for this purpose, that we call `TAME`, which is based on multiple expert networks associated with various tasks. These expert networks are added sequentially to the model in an online manner. During training, the algorithm automatically detects the task-switches based on statistically-significant deviation in the values of the loss function. At testing, the task identity is estimated by a selector network that is trained on a subset of training data that was drawn uniformly at random from all tasks. Experimental results show the efficacy of our approach on standard continual learning data sets, outperforming previous state-of-the-art techniques in terms of performance and model size. Specifically, we outperform the previous task-agnostic methods BGD, iTAML, HCL, and CN-DPM on various data sets, as well as the other techniques that take advantage of the knowledge of task descriptors at least during training.

References

[1] Alessandro Achille, Tom Eccles, Loic Matthey, Chris Burgess, Nicholas Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 3

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017. 3

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018. 3

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 3

[5] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3, 5

[6] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 3, 5

[7] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. Recent Developments on Deep Big Vision. 1

[8] D Hassabis, D Kumaran, C Summerfield, and M Botvinick. Neuroscience-inspired artificial intelligence. In *Neuron*, 2017. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[10] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Mach. Learn.*, 32(2):151–178, 1998. 1

[11] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2019. 1

[12] Steven CY Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Chu-Song Chen, et al. Compacting, picking and growing for unforgetting continual learning. *arXiv preprint arXiv:1910.06562*, 2019. 4

[13] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. *AAAI*, 2018. 3

[14] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C Kanan. Measuring catastrophic forgetting in neural networks. *AAAI*, 2018. 1

[15] Polina Kirichenko, Mehrdad Farajtabar, Dushyant Rao, Balaji Lakshminarayanan, Nir Levine, Ang Li, Huiyi Hu, Andrew Gordon Wilson, and Razvan Pascanu. Task-agnostic continual learning with hybrid probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. 3, 5

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 3, 5

[17] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 2015. 1

[18] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations, ICLR 2020*, 2020. 1, 3, 5

[19] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *arXiv preprint arXiv:1904.00310*, 2019. 4

[20] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 3

[21] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017. 3

[22] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 4

[23] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018. 4

[24] D Maltoni and V Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Netw*, 2019. 1

[25] JL McClelland, BL McNaughton, and RC O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological Review 102*, 102:419–457, 1995.

[26] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989. 1

[27] Claire Monteleoni. Online learning of non-stationary sequences. *SM Thesis, MIT Artificial Intelligence Technical Report*, 2003. 1

[28] Claire Monteleoni and Tommi Jaakkola. Online learning of non-stationary sequences. *NeurIPS*, 2003. 1

[29] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 5

[30] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 1, 2

[31] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml : An incremental task-agnostic meta-learning approach. *2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 5

[32] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Whye Yee Teh, and Raia Hadsell. Continual unsupervised representation learning. *NeurIPS*, pages 7645–7655, 2019. 1, 3

[33] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018. 1, 3

[34] Mohammad Rostami, Soheil Kolouri, and Praveen K. Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3339–3345, 2019. 3

[35] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. 3

[36] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 3

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations, ICLR 2015*, 2015. 6

[38] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2011. 6

[39] Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1):25–46, 1995. The Biology and Technology of Intelligent Autonomous Agents. 1

[40] Gido M. van de Ven and Andreas S. Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2019. 1

[41] Wikipedia. Moving average. 4

[42] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 4, 5

[43] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 3, 5

[44] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2019. 1, 2, 5