Image and Vision Computing 116 (2021) 104313

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Incremental human action recognition with dual memory

Matheus Gutoski *, André Eugenio Lazzaretti, Heitor Silvério Lopes

Federal University of Technology – Paraná, Av. Sete de Setembro, 3165 – Rebouças, Curitiba 80230-901, Brazil

ARTICLE INFO

Article history:
Received 26 July 2021
Received in revised form 21 September 2021
Accepted 22 September 2021
Available online 01 October 2021

Keywords:
Incremental learning
Human Action Recognition
Metric Learning
Triplet Networks
Dual-memory Extreme Value Machine

ABSTRACT

Incremental learning is a topic of great interest in the current state of machine learning research. Real-world problems often require a classifier to incorporate new knowledge while preserving what was learned before. One of the most challenging problems in computer vision is Human Action Recognition (HAR) in videos. However, most of the existing works approach HAR from a non-incremental point of view. This work proposes a framework for performing HAR in the incremental learning scenario called Incremental Human Action Recognition with Dual Memory (IHAR-DM). IHAR-DM contains three main components: a 3D convolutional neural network for capturing Spatio-temporal features; a Triplet Network to perform metric learning; and the dual-memory Extreme Value Machine, which is introduced in this work. The proposed method is compared with 10 other state-of-the-art incremental learning models. We propose five experimental settings containing different numbers of tasks and classes using two widely known HAR datasets: UCF-101 and HMDB51. Our results show superior performance in terms of Normalized Mutual Information (NMI) and Inter-task Intransigence (ITI), which is a new metric proposed in this work. Overall results show the feasibility of the proposal for real HAR problems, which mostly present the requirements imposed by incremental learning.

1. Introduction

Incremental learning is a topic of great importance in the current state of machine learning research. Some real-world problems require incorporating new knowledge into existing models using limited time and computational resources. Such limitations become even more apparent when dealing with video data, as the cost of storing and processing a video is substantially higher than that of static images. Conventional machine learning models are inadequate for incremental learning because they need to be retrained from scratch using new and old data. This approach implies storing all past data in an unbound memory, quickly becoming impractical.

Different from the conventional models, incremental learners gradually incorporate knowledge while preserving computational resources. Memory size remains fixed or grows very slowly to accommodate new classes [27], and model updates are more cost-efficient in terms of computational cost when compared to retraining from scratch.

Incremental learning is divided into two subcategories: Task-incremental learning and Class-incremental learning. Task-incremental learning assumes the task-id to be known during the prediction phase. Therefore, the task-incremental classifier only needs to consider classes from the given task to classify a new instance. We approach the more difficult problem of Class-incremental learning, which does not require a task-id to perform prediction [20]. In Class-incremental learning, the model must be able to classify a data point as any of the classes learned so far, regardless of the task.

While most of the research in Class-incremental Learning was conducted using image datasets such as ImageNet [41,26], we tackle a more challenging problem: Human Action Recognition in videos. Most previous works approached HAR under the classical non-incremental scenario [44,25,11]. We show that it is possible to achieve high performance in HAR, despite the Class-incremental learning restrictions – a subject still underexplored in the literature. Therefore, to address such a problem, we propose a framework with three main components:

1. A 3D convolutional neural network trained on the initial data to transform input videos into *fixed* representations. We call them fixed representations because the network is trained only once. The I3D model [4] was chosen for this task.
2. A Triplet Network that performs Metric Learning using the fixed representations as input. The Triplet Network outputs *dynamic* representations by learning new tasks incrementally.
3. An incremental classifier capable of working with dynamic representations. The original Extreme Value Machine (EVM) [29] performs incremental learning by fitting different models to the new data. However, such a model is limited to fixed feature representations. On the other hand, the proposed approach cannot operate under fixed representations because the characterizations of the previous

* Corresponding author.
E-mail address: matheusgutoski@alunos.utfpr.edu.br (M. Gutoski).

https://doi.org/10.1016/j.imavis.2021.104313
0262-8856/

EVs change as the Triplet Network learns new classes. Hence, the EVM was extended to a dual-memory scheme to allow dynamic representation incremental learning. The dual-memory EVM stores both fixed and dynamical representations to achieve incremental learning coupled with deep representation learning. The dual-memory EVM only requires storing the representations of a few Extreme Vectors, thus having a low memory cost. Although this work is focused only on incremental learning, the dual-memory EVM is also capable of rejection, which is a required step to extend it into open-world recognition.

Accordingly, the main contributions of this work are summarized as follows:

- A framework for performing Incremental HAR in videos with low memory cost;
- The introduction of the dual-memory EVM to work with dynamical representations;
- The introduction of Inter-task Intransigence, a new metric for measuring Intransigence in incremental learning problems;
- The proposal of five experimental settings using the UCF-101 and the HMDB51 datasets, which can be used as a future reference for evaluating Incremental HAR.

The remainder of this paper is organized as follows. Section 2 points out the main gaps of related works and highlights the original aspects of this research. Section 3 presents an overview of the proposed approach supporting the main contributions of this work. The detailed theoretical aspects that support our method are presented in Section 4. Next, Section 5 presents in detail the proposed incremental learning procedure, task generation, and evaluation protocol. Section 6 details the results, comparisons, and discussions for different experiments. Finally, Section 7 shows the conclusions and suggestions for future works.

2. Related works

Incremental learning has been a topic of interest since the early stages of Neural Network research [9]. In the past few years, this topic has received increased attention from the research community [20,8].

Incremental Learning models face additional challenges in comparison to traditional classifiers, such as catastrophic Forgetting and Intransigence [24,6]. These challenges require a model to remember previous knowledge while being able to incorporate new knowledge. Another recurrent problem in Incremental Learning is task-recency bias. This problem refers to the tendency of incremental learning models to wrongly classify data as one of the recently-learned classes [20].

Recent works have devised a variety of strategies to perform Incremental Learning while facing these challenges. Delange et al. [8] categorized these works according to three types of strategies: replay, regularization, and parameter isolation methods, which will be discussed in the following sections. We also discuss some works related to HAR in the incremental learning setting.

2.1. Replay methods

Replay methods use a small number of exemplars from previous classes while learning a new task. Replaying samples alleviate Forgetting by forcing the network to maintain past knowledge while incorporating new knowledge. *Incremental Classifier and Representation Learning* (iCaRL) [27] was the first method to address Class Incremental Learning using replay. iCaRL selects and stores exemplars based on herding, such that the selected samples are close to their respective class mean in the feature space. Wu et al. [41] introduced the *Bias Correction* (BIC) model, which uses exemplars to train a bias correction layer. The model was developed to handle large-scale incremental learning, emphasizing the problem of task-recency bias. Similarly, Belouadah and Popescu [3] approached task-recency bias by using the *Incremental Learning with dual memory* (IL2M) model. IL2M rectifies predictions by leveraging information from two memories: stored exemplars and statistics from previous classes. Hou et al. [14] introduced LUCIR (*Learning a Unified Classifier Incrementallu via Rebalancing*), which adopts cosine normalization, less-forget constraint, and inter-class separation to avoid task-recency bias and alleviate Forgetting. *End-to-End Incremental Learning* (EEIL) [5] performs a balanced training phase using exemplars to reduce task-recency bias.

The main drawback of replay methods is the storage cost. Storage may become quite expensive depending on the application, especially when working with video data. To address this issue, IHAR-DM employs *feature-rehearsal* [20]. Instead of storing raw videos, we store only their feature representations. This requires much smaller memory usage than regular exemplar-rehearsal strategies.

2.2. Regularization methods

Regularization methods alleviate Forgetting by introducing penalty terms to the loss function or by performing knowledge distillation. Penalty terms are obtained by estimating the importance of each weight at performing a given task. *Elastic Weight Consolidation* (EWC) [15] was the first method in this category. EWC slows down the training of the most relevant weights while learning a new task. Another approach, *Path Integral* [43], calculates parameter importance by observing changes to each weight during the whole training process. *Memory Aware Synapses* (MAS) [1] estimates parameter importance in an unsupervised manner by using a hold-out dataset. *Riemanian Walk* (RWalk) [6] combines EWC and Path Integral for online parameter importance estimation. Moreover, exemplars can be used to improve performance. As pointed out by Delange et al. [8], regularization strategies may not suffice to avoid Forgetting in earlier tasks with long sequences.

Knowledge distillation is a technique that transfers knowledge from an old model to a new model. *Learning Without Forgetting* (LWF) [17] performs Incremental Learning by using a modified distillation loss. This approach prevents weights from drifting too much from their original values. The drawback of this method is its vulnerability to domain shift between tasks, as shown in Delange et al. [8] and Aljundi et al. [2]. It is worth mentioning that the method proposed in the present work does not belong to the regularization family.

2.3. Parameter isolation methods

To prevent Forgetting, parameter isolation methods maintain task-specific parameters after learning a different model or branch for each task. However, parameter isolation models may be prohibitive in memory-constrained applications. Moreover, this type of method usually requires the task-id to be given during the classification phase. The task-id can be provided by either a human in the loop or by another classifier. Some parameter isolation methods learn task-specific masks to differentiate between tasks while keeping the same network weights. *Piggyback* [19] computes binary masks over the weights of a backbone network. Conversely, *Ternary Feature Masks* (TFM) [21] computes masks at the feature level instead of at the weights level. Both methods require the task-ID to be given. Other parameter isolation methods grow the network architecture to accommodate new tasks. *Progressive Neural Networks* (PNN) [30] duplicates the network for each new task. The main drawback of this method is the quadratic growth of parameters. This issue was later improved in *Progress and Compress* [31]. Expert Gate [2] also used a growing network architecture. However, Expert Gate uses an Autoencoder to predict the task-id, thus avoiding the need for a human oracle.

IHAR-DM also contains parameter isolation elements. The EVM [29] incrementally adds new models to accommodate new classes. Unlike some other methods in this category, the EVM does not require replicating any parameters to incorporate new classes, leading to linear growth in memory size. The rate of growth can also be controlled by adjusting

the model reduction parameter. Moreover, it does not require the task-id to be given in advance. The EVM is able to classify samples into any of the previously learned classes, regardless of the task-id.

2.4. Incremental Learning for HAR

Very few works have explored HAR in the Incremental Learning setting. ODN [32] performs Incremental Learning for HAR in the broader task of Open-world Recognition. The authors introduce the concepts of Emphasis Initialization and Allometry Training. The method shows promising results in terms of accuracy but appears to suffer from class-recency bias. However, the focus of their work is not exclusively on Incremental Learning, and thus, it lacks an in-depth evaluation of important metrics such as Forgetting and Intransigence. Reddy et al. [28] proposed a feature-tree to perform HAR in the KTH [39] and the IXMAS [38] datasets. However, these datasets were staged by actors and contain only 6 and 11 classes, respectively [33]. Hence, they became outdated in comparison to more recent datasets. The work presented a small incremental learning experiment in which a model was trained with 4 classes and evaluated with 5, that is, only one class was learned incrementally.

Other works perform HAR under the related field of online learning, which operates over streams of data [35]. Some of those works perform classification at frame-level in very controlled video datasets [40,18,23,7]. Moreover, older works in this field do not consider more recent evaluation metrics such as Forgetting and Intransigence [8]. To the best of our knowledge, ours is the first work to address Incremental HAR with modern evaluation protocols.

2.5. IHAR-DM highlights

IHAR-DM combines the advantages of replay and parameter isolation, while also minimizing the drawbacks of these categories. The main drawback of replay methods is the memory size, which we mitigate using *feature-rehearsal*. As for parameter isolation methods, we avoid excessive growth of parameters and perform prediction without a task-id. Moreover, to the best of our knowledge, this work is the first that applies these concepts to HAR in modern datasets.

3. Overview

This section presents an overview of the IHAR-DM, supporting the main contributions of this work: (i) the proposed framework for performing Incremental HAR; (ii) the proposed experimental settings for evaluating Incremental HAR; and (iii) the proposed Inter-Task Intransigence metric.

3.1. Proposed framework for Incremental HAR

Our framework for Incremental Human Action Recognition, named IHAR-DM, consists of three main blocks: A 3D convolutional neural network (I3D), a Triplet Network, and the Dual-Memory EVM. The relationship between the blocks is shown in Fig. 1.

First, the input videos go through the I3D to generate their fixed 1024-dimensional feature vector representation. It is called “fixed representation” because the I3D is trained only once, using the initial training data. The weights of the I3D are then fixed for the remaining steps of the incremental learning process.

The second step is to train a Triplet Network that performs metric learning using the fixed representations as input. This network is fully trained using the initial training data and, later, it is updated with a quick fine-tuning during each of the incremental phases. The Triplet Network outputs a 256-dimensional feature vector, which we call “dynamical representations”, since they change with each new task.

The third step is to train the Dual-Memory EVM. It uses both the fixed and the dynamical representations of the video to perform incremental learning and classification. The purpose of the dynamical representations is to fit the dual-memory EVM and perform predictions, while the fixed representations are used to update the already existing models during the incremental learning stages. The fixed representations of the Extreme Vectors found by the Dual-Memory EVM participate in the incremental training of the Triplet Network. More details about the proposed framework are presented in Section 4.

3.2. Proposed experimental settings

Because of the lack of recent models for Incremental HAR, there are currently no general guidelines on how to perform their evaluation. Therefore, we propose three experimental settings for the UCF-101 dataset [33] and two experimental settings for the HMDB51 dataset [16]. The difference between each experimental setting is the number of tasks and the number of classes within each task. This allows evaluating Incremental HAR models under different scenarios, such as small to a large number of tasks, or small to a large number of classes contained in each task. The different settings will be presented in Section 6.1.

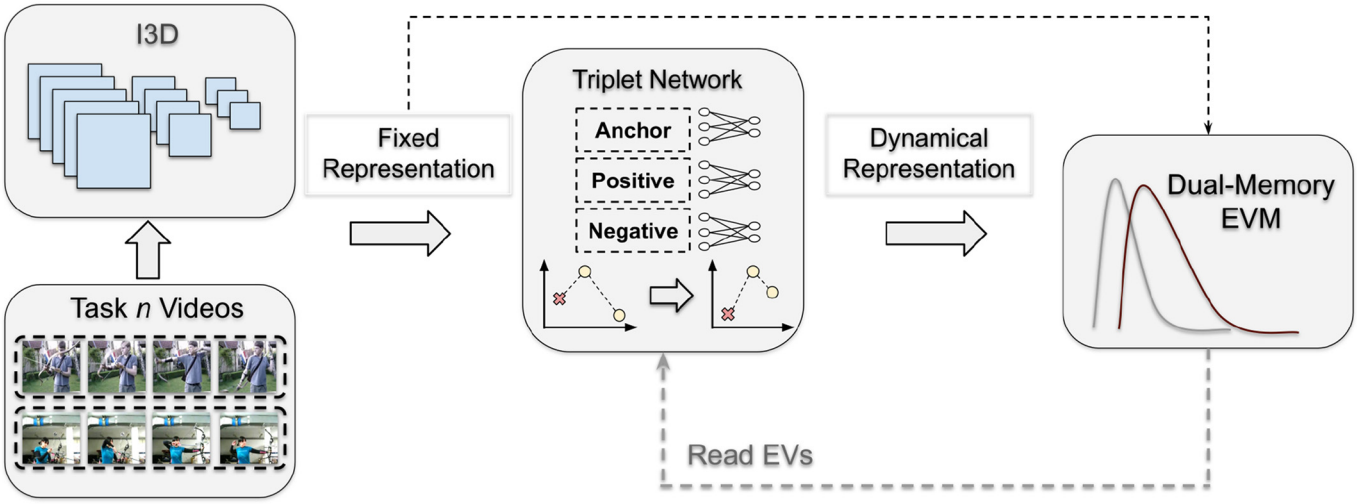

Fig. 1. Incremental Learning Process of the proposed framework. First, videos go through the I3D network and become a 1024-dimensional fixed representation. The fixed representation goes through the Triplet Network and becomes a 256-dimensional dynamical representation. Both of the representations are used by the Dual-Memory EVM to perform incremental learning and classification. The EVs defined by the Dual-Memory EVM are used to update the Triplet Network in subsequent tasks.

3.3. Inter-task intransigence

Chaudhry et al. [6] defined Intransigence as the inability of a model to learn new tasks. The authors proposed to compute Intransigence as the difference in the performance between an incremental model and a reference model. The reference model was trained with access to all tasks at once and it was considered the gold standard. Such a metric is difficult to compute due to the high computational cost of training a reference model for the only purpose of measuring Intransigence. Moreover, the reference model is also susceptible to parameter tuning. Therefore, the use of Intransigence leads to a scenario where the measured performance of the model under test is directly affected by the parameters of a second completely unrelated model. This also makes the comparison with other state-of-the-art models quite difficult, because each work may use a different reference model to measure Intransigence.

Hence, to overcome the above-mentioned drawbacks of the existing methods, another metric is proposed here: the Inter-Task Intransigence (ITI). This metric can be computed without a reference model, and it can be interpreted as the performance decay of a new task with respect to the previous one. In other words, ITI measures the inability to learn new tasks with the same effectiveness as in the previous tasks. Section 5.4 presents a formal definition of ITI, along with the other evaluation metrics.

4. Theoretical aspects

This section details the most relevant theoretical aspects of the proposed method, including the Inflated 3D Convolutional Neural Network (I3D), Metric Learning with Triplet Networks, and the Dual-Memory Extreme Value Machine.

4.1. Inflated 3D Convolutional Neural Network

The Inflated 3D Convolutional Neural Network is a model proposed by Carreira and Zisserman [4] for performing video classification. The I3D contains two streams of the Inception model with inflated kernels. Inflation is a process that extends the filters learned on ImageNet from 2D to 3D as an initialization method. The original I3D model contains two streams of data. The first stream uses RGB images, while the second stream uses a precomputed Optical Flow. In this work, we employ only the RGB stream. The model is trained using backpropagation with softmax cross-entropy loss. We choose the I3D for its state-of-the-art performance. However, IHAR-DM supports any kind of video classification model as the backbone.

4.2. Metric Learning with Triplet Networks

Metric Learning (ML) aims at mapping features into a space where the distance between two points corresponds to a measure of semantic similarity. The Large Margin Nearest Neighbors (LMNN) [37] is one of the most relevant ML models in classic Machine Learning. Inspired by models such as LMNN, Triplet Networks were first introduced for learning face embeddings [13].

The Triplet Network receives three inputs: an Anchor, a Positive, and a Negative. In our work, the Anchor (a) corresponds to the feature representation (obtained by the I3D) of a video of any given class, the Positive (p) is the representation of a video from the same class, and the Negative (n) is the representation of a video from a different class.

Given N (a, p, n) triplets, the Triplet loss function L_Θ can be defined as

$$L_\Theta = \sum_{i=1}^{N} \left[\Theta(g(\mathbf{x}_i^a), g(\mathbf{x}_i^p)) - \Theta(g(\mathbf{x}_i^a), g(\mathbf{x}_i^n)) + \alpha\right]_+, \tag{1}$$

in which i is the triplet index, g is the Triplet Network, $g(\mathbf{x}^a), g(\mathbf{x}^p), g(\mathbf{x}^n)$ are the Anchor, Positive, and Negative output representations, α is the margin parameter, and $_+$ indicates that the loss is ≥0. Θ represents the cosine distance between two feature representations $\mathbf{d}$ and $\mathbf{e}$. The cosine distance Θ is defined as

$$\Theta(\mathbf{d}, \mathbf{e}) = 1 - \frac{\mathbf{d} \cdot \mathbf{e}}{\|\mathbf{d}\|\|\mathbf{e}\|}. \tag{2}$$

Our Triplet Network contains three fully connected layers with 1024, 512, and 256 neurons. The weights were initialized using the Glorot uniform initialization method [10].

Triplet mining is of great importance for Triplet Networks [13]. Using all possible combinations of triplets may require huge computational resources. Moreover, most triplets generated in this way do not contribute to the learning process since they imply in $L_\Theta \leq 0$. This effect occurs when an anchor is already closer to the positive sample than it is to the negative sample, and the difference is larger than the margin parameter α.

Triplet mining guarantees that only relevant triplets are used for training. In our work, we employ semi-hard and hard triplets. Semi-hard triplets are those in which the distance between the Anchor and Positive is smaller than the distance between the Anchor and Negative, and also smaller than the margin α:

$$\begin{aligned}\Theta(g(\mathbf{x}^a),\, g(\mathbf{x}^p)) &< \Theta(g(\mathbf{x}^a), g(\mathbf{x}^n)) \\ &< \Theta(g(\mathbf{x}^a), g(\mathbf{x}^p)) + \alpha.\end{aligned} \tag{3}$$

Hard triplets, on the other hand, are those in which the distance between the Anchor and Positive is larger than the distance between the Anchor and Negative:

$$\Theta(g(\mathbf{x}^a), g(\mathbf{x}^p)) > \Theta(g(\mathbf{x}^a), g(\mathbf{x}^n)). \tag{4}$$

4.3. Dual-Memory Extreme Value Machine

The Extreme Value Machine (EVM) was first introduced by Rudd et al. [29] for open-set recognition. In the original EVM, each class is represented by a number of extreme vectors (EVs), which are represented by a Probability of Sample Inclusion Ψ.

In the Dual-Memory EVM, we redefine Ψ as Ψ'. The first difference between the original EVM and the Dual-Memory EVM is that we incorporate the Triplet Network g into the Ψ' models. The second difference is that we store both $\mathbf{x}_i''$ and $g(\mathbf{x}_i'')$ once it is computed for the first time to avoid recomputations during the test phase. Finally, the third difference lies in the Incremental Learning algorithm, which will be presented in Section 5.3. Eq. (5) defines Ψ', as follows:

$$\Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i) = \exp\left(-\frac{\Theta(g(\mathbf{x}_i''), g(\mathbf{x}'))}{\lambda_i}\right)^{\kappa_i}, \tag{5}$$

where $\Theta(g(\mathbf{x}_i''), g(\mathbf{x}'))$ is the cosine distance between $g(\mathbf{x}_i'')$ and $g(\mathbf{x}')$. The Weibull shape and scale parameters λ_i and κ_i are computed by fitting a Weibull distribution to the half-margins of the τ nearest points to $g(\mathbf{x}_i'')$ using Maximum Likelihood Estimation.

Let $\mathbf{x}_i''$ be a training sample (fixed representation obtained by the I3D) and y_i'' be its label, and given $\mathbf{x}_i''$ and $\mathbf{x}_j''$, where $\forall j, y_j'' \neq y_i''$, and $\mathbf{x}_j''$ is the nearest point to $\mathbf{x}_i''$, the half-margin for the pair $(\mathbf{x}_i'', \mathbf{x}_j'')$ is given by:

$$\mathbf{m}_{ij} = \frac{\Theta(g(\mathbf{x}_i''), g(\mathbf{x}_j''))}{2}. \tag{6}$$

Ψ' is a rejection model where $\Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i)$ corresponds to the probability that a sample is not beyond the negative margin. Even though a point has zero probability around the margin, the EVM also supports a soft margin. The probability that a point $\mathbf{x}'$ is a member of class C_l is given by:

$$\hat{P}(C_l|g(\mathbf{x}')) = \text{argmax}_{i:y_i=C_l} \Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i). \tag{7}$$

The final EVM classification function is:

$$\hat{y} = \begin{cases} \text{argmax}_{i:y_i=C_l} \hat{P}(C_l|g(\mathbf{x}')), & \text{if } \hat{P}(C_l|g(\mathbf{x}')) \geqslant \delta \\ \text{unknown}, & \text{otherwise} \end{cases}, \tag{8}$$

where δ is a parameter that defines the boundary between known and unknown.

Intending to reduce the size of the EVM, most redundant $[\mathbf{x}_i'', \Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i)]$ models can be discarded with minimal loss on performance [29]. Let $\mathbf{x}_i''$ be a data point and $\Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i)$ be its associated model. Let $\mathbf{x}_j''$ be a point belonging to the same class, $\Psi'(\mathbf{x}_j'', \mathbf{x}', \kappa_j, \lambda_j)$ its associated model, and ς the probability threshold above which the pair $[\mathbf{x}_j'', \Psi'(\mathbf{x}_j'', \mathbf{x}', \kappa_j, \lambda_j)]$ is considered redundant with respect to $[\mathbf{x}_i'', \Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i)]$. If $\Psi_i'(\mathbf{x}_i'', \mathbf{x}_j'', \kappa_i, \lambda_i) \geqslant \varsigma$, then $[\mathbf{x}_j'', \Psi'(\mathbf{x}_j'', \mathbf{x}', \kappa_j, \lambda_j)]$ is redundant. Finally, $I(\mathbf{x}_i'')$ is the indicator function that maintains or discards a pair as:

$$\begin{cases} I(\mathbf{x}_i'') = 1, & \text{if} \langle \mathbf{x}_i'', \Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i) \rangle \text{ kept} \\ I(\mathbf{x}_i'') = 0, & \text{otherwise.} \end{cases} \tag{9}$$

A pair becomes an extreme vector if $[\mathbf{x}_i'', \Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i)]$ is kept. The model reduction is defined by the following integer linear programming objective function:

$$\text{minimize} \sum_{i=1}^{N_l} I(\mathbf{x}_i''), \tag{10}$$

$$s.t. \ \ \forall j, \exists i \mid I(\mathbf{x}_i'') \Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i) \geqslant \varsigma,$$

in which N_l is the number of points belonging to class C_l.

5. Method

IHAR-DM strives to learn classes in a sequence of tasks. The first task is considered the Initial Training data, while the subsequent tasks must be learned under the Incremental Learning restrictions. Fig. 2 shows a macro view of the Incremental Human Action Recognition problem.

First, the video dataset is split into n tasks, defined as $\{0, 1, \ldots, n-1\}$. The first task goes through an Initial Training process, which is similar to other non-incremental methods. After the Initial Training, we start the Incremental Learning steps. IHAR-DM learns each task incrementally while having access only to the data belonging to a specific task and to the dual-memory EVM. Each step of the process will be detailed in the following sections.

5.1. Datasets and task generation

Two HAR datasets were used to evaluate the proposed method: UCF-101 [33] and HMDB51 [16]. These datasets are well-known and were widely used in previously published works [4,42,36].

The UCF-101 dataset contains over 27 hours of people performing actions such as typing, biking, and bowling, distributed in 13,320 clips. The dataset contains five main categories: human-object interaction, body-motion, human-human interaction, playing musical instruments, and sports. The clips contain camera motion and cluttered background, making it a challenging dataset.

The HMDB51 dataset contains 6766 video clips distributed into 51 classes. The classes are grouped into five main types: general facial actions; facial actions with object manipulation; general body movements; body movements with object interaction; and body movements for human interaction [16]. Clips were extracted from movies and YouTube videos. The use of this dataset is very challenging, because it includes many scenes with large camera motion, cluttered backgrounds, viewpoint variations, and low-quality clips. The authors also provide 3 training/test splits for evaluating this dataset. Fig. 3 shows sample frames from the UCF-101 and the HMDB51 datasets.

For the UCF-101 dataset, data were split into 30% for testing, and 70% for training and validation. Since the UCF101 dataset [33] contains subgroups of very similar videos inside each category (snippets cut from the same long-duration video), care was taken to ensure that the

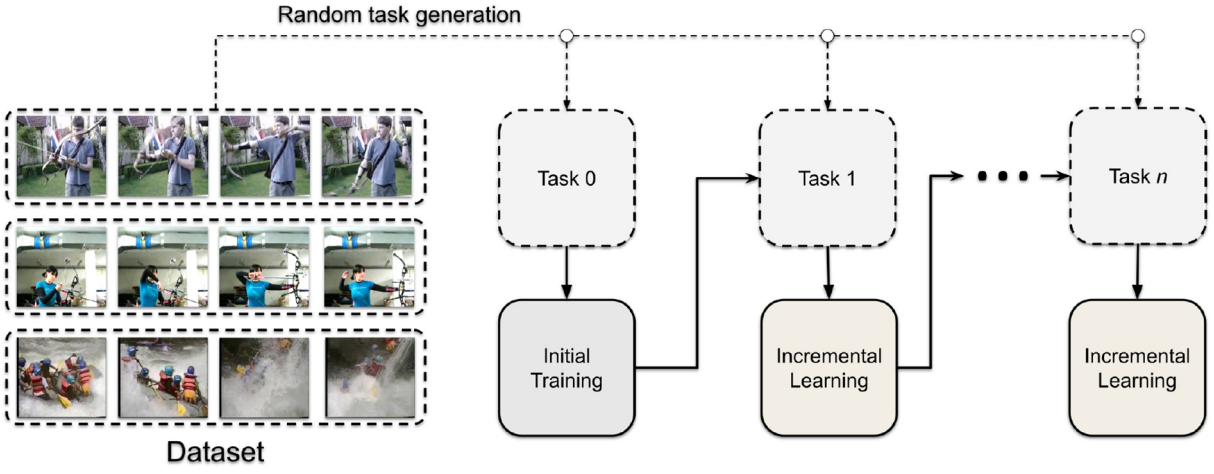

Fig. 2. Macro view of the Incremental HAR problem. First, videos go through a random task generation process. Then, tasks are learned sequentially. The first task goes through an Initial Training process, which does not suffer the restrictions of Incremental Learning. Then, the subsequent tasks are learned using our Incremental Learning approach.

train/test splits encompass every video of the same group. This prevents the test set from containing "easy" videos, which have been seen during the training phase. For the HMDB51 dataset, we used the 3 training/test splits provided by the authors.

With the training and test splits, tasks were generated by randomly selecting classes for each task. Each class appears in only one of the tasks. The number of tasks varies according to each experiment, as will be described in Section 6.1.

5.2. Initial Training process

The Initial Training process seeks to generate the initial models of the proposed framework. First, videos were transformed into feature representations. This was achieved by training a 3D Convolutional Neural Network (CEL) with standard cross-entropy loss:

$$\mathrm{CEL}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \log \hat{y}_i, \tag{11}$$

where $\mathbf{y}$ are the true labels, and $\hat{\mathbf{y}}$ are the predictions for each data point i. The I3D model [4] was chosen for this task. Both, the parameters and training protocol proposed by the original author were used here. The input at training time consisted of a window of 64 consecutive frames. This window was selected at random at each training epoch for augmentation purposes. A random spatial cuboid of $224 \times 224 \times 64$ was also selected to perform a horizontal flip (mirror) with 50% probability. The batch size was set to six. We used 30% of the training set for validation and optimize the network using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.1, weight decay of 10^{-5}, and the Nesterov momentum of 0.9. The network was trained for 10 epochs or until the loss stagnates. At the test time, a window of 250 frames was selected from the center of the video. A spatial cuboid of $224 \times 224 \times 250$ was also taken from the center of the video.

The I3D outputs a 1024-dimensional feature vector which represents an input video. The next step of the framework is to use the I3D representations to train a fully connected Triplet Network, which performs the Metric Learning. The margin parameter α was set to 0.2, the learning rate to 0.001, and the batch size to 128. These parameters were defined after a small set of preliminary experiments. In the Initial Training phase, the Triplet Network was trained for 10 epochs. Hard and semi-hard triplets were mined at the start of each epoch as shown in Section 4.2.

The next step of the process was training the Extreme Value Machine, which procedure was described in Section 4.3. The tail size τ for fitting the Weibull distribution was set to 20% of the training set size, and the classification threshold δ was set to 0.001. These parameter values were chosen after a set of preliminary experiments that pointed out a good trade-off between performance and computational cost.

The variable δ controls the rejection rate of the EVM. Although Incremental Learning does not require a rejection mechanism, the framework supports this possibility. The ability to reject samples from unknown classes becomes a requirement once the problem evolves towards an open-world setting, which will be addressed in the future.

The cover threshold ς was set to 0.99. Setting ς to a higher value may cause more EVs to remain after the model reduction process. It was experimentally found that the method achieved better performance with higher values of ς, at the cost of a larger Extreme Vector Memory. Nevertheless, the size of the Extreme Vector Memory is still smaller than that of other concurrent models. Table 1 shows the number of EVs obtained by the model in five runs of experiments using the UCF-101 dataset with different random seeds. The larger ς, the more EVs are stored in memory. At the maximum value of ς, an average of 18.7 EVs per class were stored. For a fair comparison, the memory size of the concurrent methods was set to 20 exemplars per class.

Since the proposed method requires two different feature representations to be stored (fixed with 1024 features and dynamical with 256 features), the memory cost becomes 1280 32-bits floating points per EV. Table 1 shows the number of EVs for a wide range of ς values. According to this Table, and using an average of 1892 EVs, the dual-memory costs around 9 megabytes to learn all the 101 classes of videos of the UCF-101 dataset.

5.3. Incremental learning process

Algorithm 1. Incremental learning process

Input : New task data and existing Ψ' models
Output: Updated Triplet Network g and Ψ' models

```
/* Compute fixed representations x'' using
   the I3D Network g''                          */
```
foreach *video* $\mathbf{v}_j$ *in the new task* **do**
  Compute fixed representations $\mathbf{x}''_j = g''(\mathbf{v}_j)$
  where j is the new video index;
end

```
/* Update Triplet Network g                     */
```
foreach *existing* Ψ'_i *model* **do**
  Get fixed representations $\mathbf{x}''_i$
  where i is the Ψ'_i model index;
end
Finetune Triplet Network g for 1 epoch using
$\mathbb{X}'' = \{\mathbf{x}''_i \mid \forall i\} \bigcup \{\mathbf{x}''_j \mid \forall j\}$

```
/* Update existing Ψ' models of the
   dual-memory EVM                              */
```
foreach *existing* Ψ' *model* **do**
  Recompute parameters of $\Psi'(\mathbf{x}''_i, \mathbf{x}', \kappa_i, \lambda_i)$;
end

```
/* Learn new Ψ' models for each class in
   the new task                                 */
```
foreach *Class in the new task* **do**
  Compute parameters of $\Psi'(\mathbf{x}''_j, \mathbf{x}', \kappa_j, \lambda_j)$;
  Perform model reduction ;
end

With the arrival of a new task, the Incremental Learning process begins by forwarding the current videos through the already trained I3D to obtain their 1024-dimensional (fixed) representations. Next, both, the Triplet Network and the dual-memory EVM are incremented. For the former, the fixed representations $\mathbf{x}''$ from all previous Ψ' models (EVs) were gathered and added to the triplet mining pool along with the new data. The Triplet Network was initialized with the weights from the last task and perform a single-epoch fine-tuning.

For incrementing the dual-memory EVM, first, all the existing Ψ' models (EVs) were recomputed using the representations obtained from the updated triplet network g. In practice, this step involves recomputing $g(\mathbf{x}''_i)$, κ_i, and λ_i. The term $g(\mathbf{x}''_i)$ is recomputed with a single forward through a branch of the Triplet Network, while κ_i and λ_i are computed by fitting a Weibull distribution using Maximum Likelihood Estimation to the half-margins of the nearest τ points

Fig. 3. (a) Samples from the UCF-101 dataset. (b) Samples from the HMDB51 dataset.

(Eq. (6)). Once $g(\mathbf{x}_i'')$ is computed for the first time, it is stored in the dual-memory along with $\mathbf{x}_i''$, hence the term dual-memory EVM. Storing $g(\mathbf{x}_i'')$ is beneficial because $\Psi'(\mathbf{x}_i'', \mathbf{x}', \kappa_i, \lambda_i)$ can be computed multiple times during the prediction phase, i.e., once for each test point $\mathbf{x}'$.

We included the new task data as well as the existing EVs as eligible for recomputing the Weibull parameters. This process is significantly faster than learning from scratch because the EVs are sparse and have already been defined. Model reduction is performed only when first learning a class and, therefore, it is not executed at this step. New tasks are added incrementally by learning new Ψ' models for each class and performing model reduction to define the new EVs. The entire proposed incremental learning process is shown in Algorithm 1.

5.4. Evaluation protocol and metrics

All results reported in this work present the averages of 5 initializations with different random seeds on the UCF-101 dataset, and 3 initializations on the HMDB51 dataset (each with a different training/test split provided by the authors of the dataset). Different initializations change the order in which classes appear within the tasks. All methods received the same sequence of classes/tasks at each run. We omit the initialization averaging from the following equations in this section for simplicity.

The Normalized Mutual Information (NMI) score [34] was used as the main performance measure. NMI is independent of exact label matching and is often used in clustering applications [22]. NMI is also a suitable metric for the future natural extension of this work, the open-world recognition. NMI for task t after learning up to task k is defined as:

$$\mathrm{NMI}_k^t(\mathbf{y}, \hat{\mathbf{y}}) = \frac{I(\mathbf{y}, \hat{\mathbf{y}})}{\sqrt{H(\mathbf{y})H(\hat{\mathbf{y}})}}, \tag{12}$$

where H is the entropy, $\mathbf{y}$ are the ground truth labels, and $\hat{\mathbf{y}}$ are the predictions. $I(\mathbf{y}, \hat{\mathbf{y}})$ is the mutual information between $\mathbf{y}$ and $\hat{\mathbf{y}}$:

$$I(\mathbf{y}, \hat{\mathbf{y}}) = \sum_i \sum_j \frac{|y_i \cap \hat{y}_j|}{N} \log\left(\frac{N|y_i \hat{y}_j|}{|y_i||\hat{y}_j|}\right), \tag{13}$$

in which N is the total number of samples, and $|\cdot|$ represents the cardinality. The entropy H is defined as:

$$H(\mathbf{y}) = -\sum_i \frac{|y_i|}{N} \log\left(\frac{|y_i|}{N}\right). \tag{14}$$

Forgetting is computed at the task level, based on the maximum NMI achieved by a task since it has first arrived, minus its NMI after the final task has been learned, as suggested by Chaudhry et al. [6]. Forgetting for task t after incremental training up to task k is defined as:

$$f_k^t = \max_{l \in \{0\ldots,k-1\}} \mathrm{NMI}_l^t - \mathrm{NMI}_k^t, \quad \forall t < k. \tag{15}$$

The Mean Forgetting of a model is obtained by averaging the Forgetting of each task (after training up to the final task k).

Moreover, we also used the Mean ITI as a performance metric in our experiments. For computing ITI, the NMI of each task t is subtracted by the NMI of the previous task $t-1$, up to the final task k. Then, all values are averaged, as shown in Eq. (16):

$$\mathrm{ITI}_k^t = \frac{1}{k-1} \sum_{t=1}^{k} \mathrm{NMI}_k^t - \mathrm{NMI}_k^{t-1}. \tag{16}$$

ITI can be either positive or negative. The former reflects an overall gain in performance as tasks are learned, while the latter means a decay in the performance of new tasks as they are learned.

6. Experiments, results and discussion

6.1. Experimental settings

As previously mentioned in Section 3.2, we propose five experimental settings for evaluating Incremental HAR. Table 2 shows the experimental settings and their parameters. The first column contains the reference code for each experiment. In the following sections, we present and discuss the results obtained in each setting.

Table 1
Number of EVs obtained per ς value.

Threshold	Experiment #					Mean	Mean EVs
ς	1	2	3	4	5	EVs	Per Class
0.1	580	573	601	582	647	596.6	5.907
0.25	784	771	832	769	871	805.4	7.974
0.5	1031	1013	1100	1010	1125	1055.8	10.453
0.75	1256	1288	1359	1271	1354	1305.6	12.927
0.9	1458	1504	1613	1501	1570	1529.2	15.141
0.99	1790	1870	2010	1850	1940	1892	18.733

Table 2
Experimental settings proposed for each dataset.

UCF-101 setting	# Initial CLASSES	# Classes/task	# Tasks
UCF-S1	6	5	20
UCF-S2	11	10	10
UCF-S3	21	20	5
HMDB51 setting	**# Initial classes**	**# Classes/task**	**# Tasks**
HMDB-S1	6	5	10
HMDB-S2	11	10	5

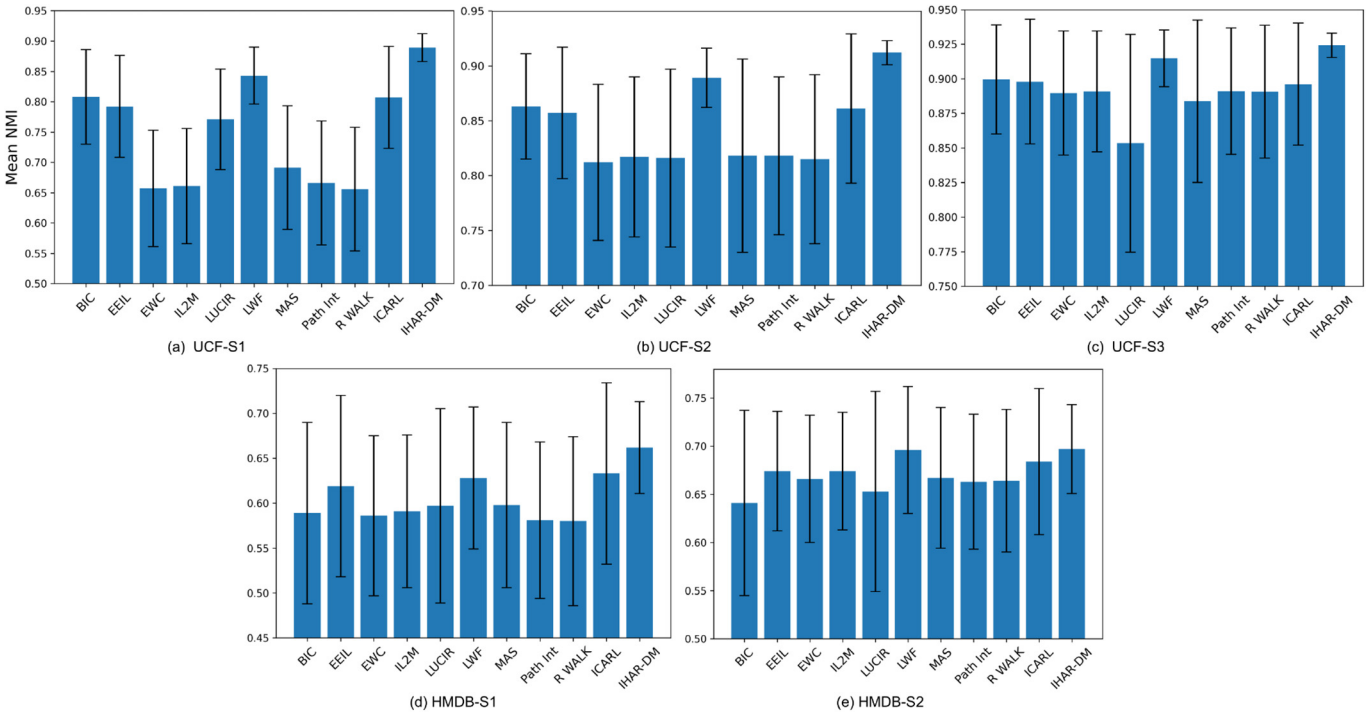

Fig. 4. Mean NMI between all tasks in five experimental settings. The error bars represent the standard deviation.

6.2. Comparison settings

The proposed method was compared with several state-of-the-art models that were presented in Section 2: BIC [41], EEIL [5], EWC [15], ICARL [27], IL2M [3], LUCIR [14], LWF [17], MAS [1], Path Integral [43], and Riemannian Walk [6]. The implementation provided by Masana et al. [20][1] was used in the experiments. It greatly facilitated the comparison with existing methods.

Since this work tackles video classification, the I3D network was used as the backbone for all methods. The Initial Training step was followed by first training the I3D network on the first task, using a regular softmax cross-entropy loss function. The I3D converts videos from all tasks to feature representations. These representations are used as input to all the above-mentioned state-of-the-art methods.

Since all comparison methods are based on neural networks, we employ a Multilayer Perceptron (MLP) to receive the input features. The MLP had four layers with 1024, 512, and 128 neurons respectively, each one followed by Rectified Linear Units (ReLUs). The output layer contains the number of neurons equals to the number of classes in the current task.

For the methods that use exemplars, the sampling strategy was set to "herding" [27]. Herding selects exemplars based on their feature representation so that they are close to the class mean. The memory size was set to 20 exemplars per class. For methods that use knowledge distillation, the Temperature Scaling parameter was set to 2, the same value in most works in the literature [41]. The remaining parameters were set according to Masana et al. [20].

6.3. Results

This section is organized as follows. First, in Section 6.3.1, we discuss the Mean Forgetting and the Mean NMI obtained by each method in all experimental settings. The second part (Section 6.3.2) presents the results concerning the ITI. Finally, in the third part (Section 6.3.3), we present and analyze the confusion matrices of each method in each dataset. All table data used to generate the charts can be found in the Appendix Section.

6.3.1. Experimental Results in Terms of NMI and Forgetting

The average NMI among all tasks indicates the general classification performance of a model. Fig. 4 shows the performance of each model in five experimental settings (refer to Table 2). In general, IHAR-DM presented both higher NMI and smaller standard deviation in comparison with other methods in all experimental settings, indicating a more stable performance throughout tasks.

Fig. 4 also shows that all methods achieved a better overall performance in settings with fewer tasks. In the UCF-101 experiments, shown in (a), (b), and (c), there is an overall positive trend in performance from settings UCF-S1 (20 tasks) to UCF-S2 (10 tasks), and finally to UCF-S3 (5 tasks). The same trend can be observed in the HMDB-51 experiments, as shown in (d) and (e), going from HMDB-S1 (5 tasks) to HMDB-S2 (10 tasks). However, IHAR-DM had the least difference in NMI between UCF-S1, UCF-S2, and UCF-S3 when compared to all other methods. The same holds true for HMDB-S1 and HMDB-S2. This indicates that IHAR-DM is robust with respect to the number of tasks, while other methods struggle to maintain performance in settings with a large number of tasks.

All methods presented higher values of NMI on the UCF-101 experiments when compared to the HMDB51 experiments. This was expected, since that HMDB51 is a more challenging dataset than UCF-101. This difference in performance among the datasets was also observed in Carreira and Zisserman [4].

Next, we investigate the Forgetting obtained by each method, as shown in Fig. 5. Overall, LUCIR and ICARL presented the lower Forgetting values considering all experimental settings. IHAR-DM presented a slightly higher forgetting when compared to other methods in (b), (c), and (d). However, Forgetting often comes as a trade-off with Intransigence [6]. Next, we investigate whether this trade-off is also true for Forgetting and ITI.

6.3.2. Experimental results in terms of ITI

ITI is a metric that measures the ability of an incremental classifier to maintain its performance on new tasks, relative to the previous tasks.

[1] https://github.com/mmasana/FACIL.

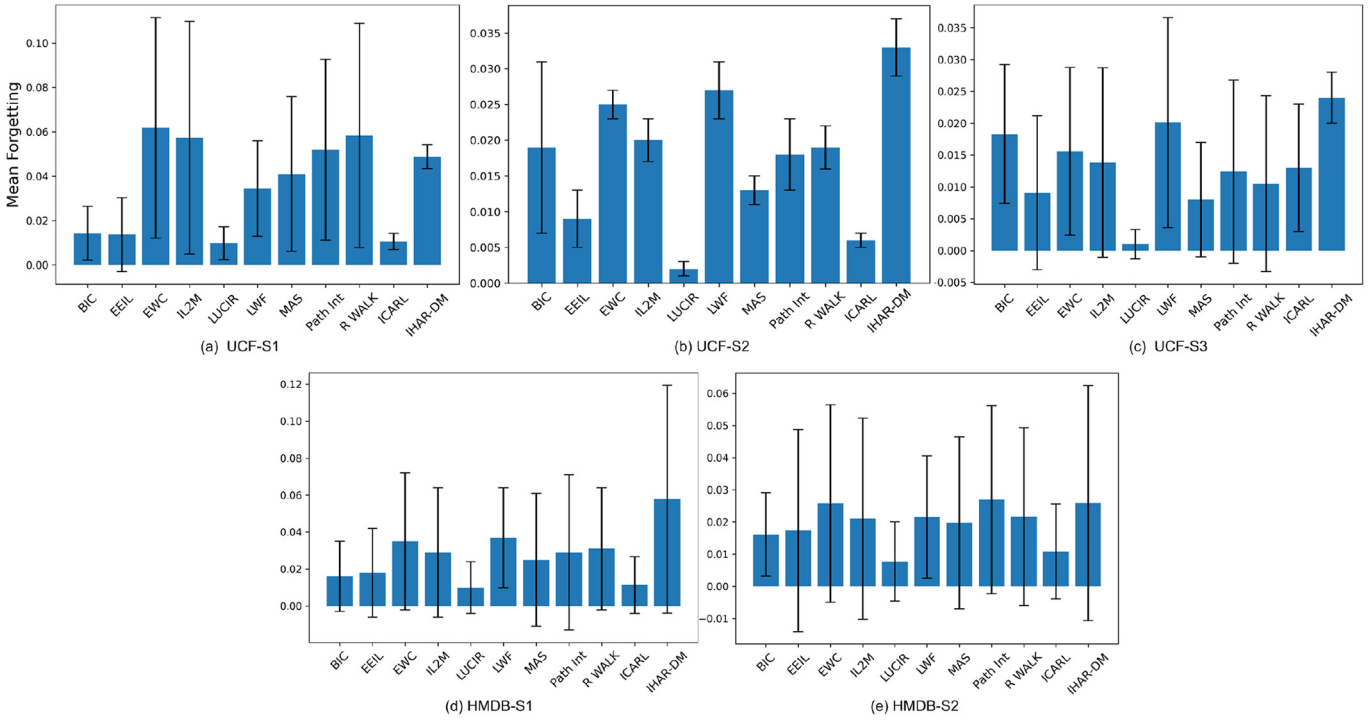

Fig. 5. Mean Forgetting between all tasks in five experimental settings. The error bars represent the standard deviation.

Fig. 6 shows the mean NMI of each method in each task. A serious problem is clearly observed: most methods suffered a performance decay as new tasks were learned. This performance decay is reflected by the ITI, as shown in Fig. 7. On the other hand, the proposed IHAR-DM keeps a more stable performance along with the tasks. IHAR-DM was the only to achieve a positive ITI among all methods, indicating a performance gain instead of decay when learning new tasks. The second method with the least performance decay was LWF. This contrasts with the values of Forgetting (Fig. 5), in which IHAR-DM and LWF presented the worst results. Additionally, LUCIR, which has shown the lowest Forgetting, presented the largest performance decay as measured by the ITI. This fact suggests that there exists a trade-off between Forgetting and ITI.

6.3.3. Analysis of confusion matrices

Confusion matrices have been previously used to diagnose task-recency bias in Incremental Learning models [27]. Task recency bias is visible when many samples get misclassified as belonging to the recently learned task. Fig. 8(a) shows the confusion matrices of a single experiment on the UCF-S1 setting. In this experiment, none of the models have shown task-recency bias. Notwithstanding, some of the models have shown the opposite effect, where samples were misclassified as one of the classes in earlier tasks. This *preceding-task bias* can be clearly observed in EWC, ICARL, IL2M, LUCIR, MAS, Path Integral, and R WALK by inspecting the higher density of errors in the left-most columns of the confusion matrices. The most likely cause for this

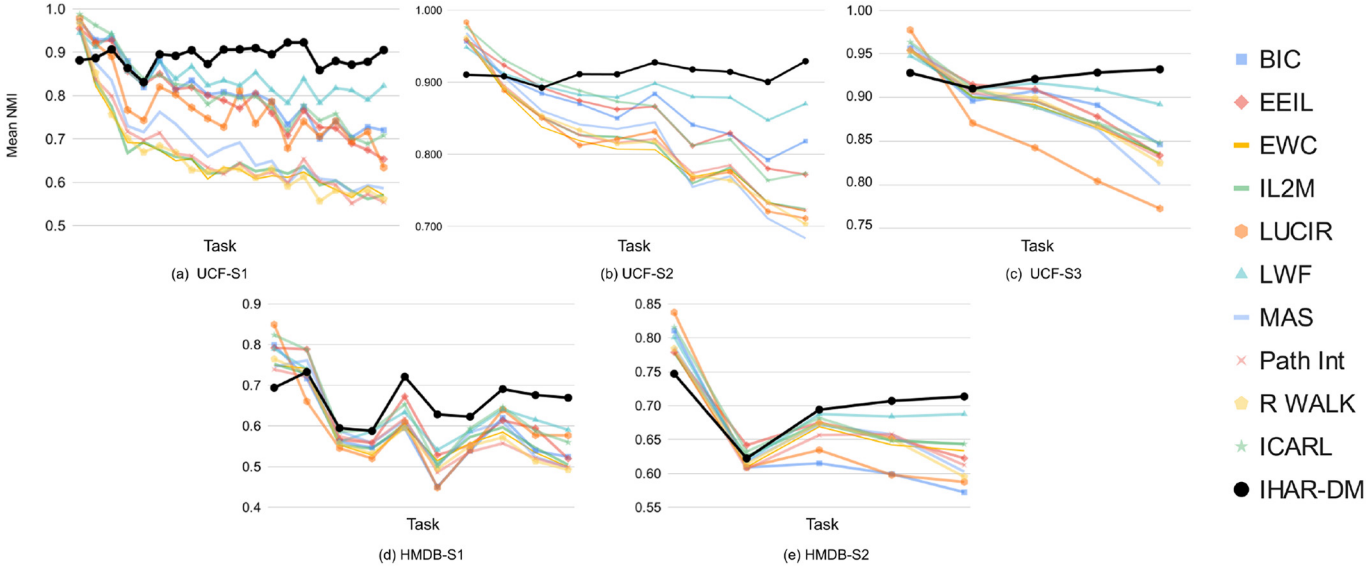

Fig. 6. Mean NMI per task in five experimental settings. Figure best viewed in color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

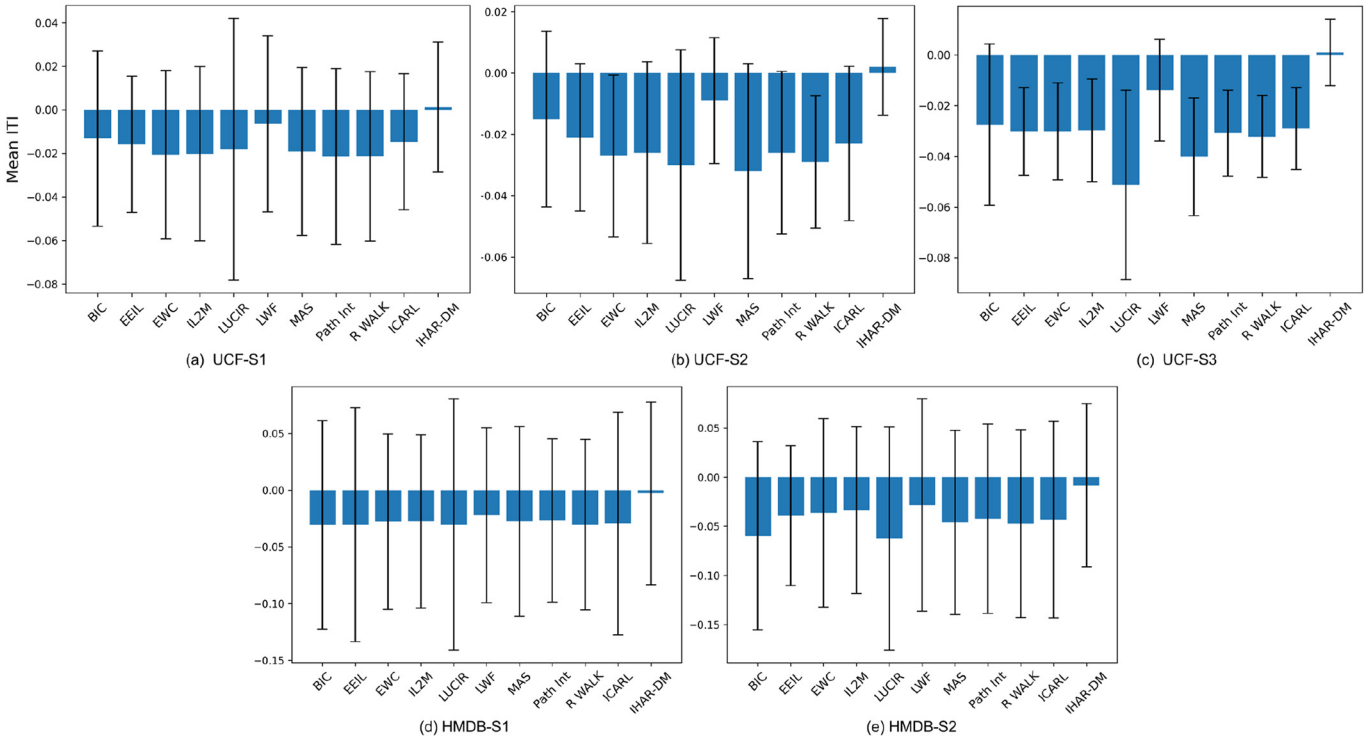

Fig. 7. Mean ITI between all tasks in five experimental settings. The error bars represent the standard deviation.

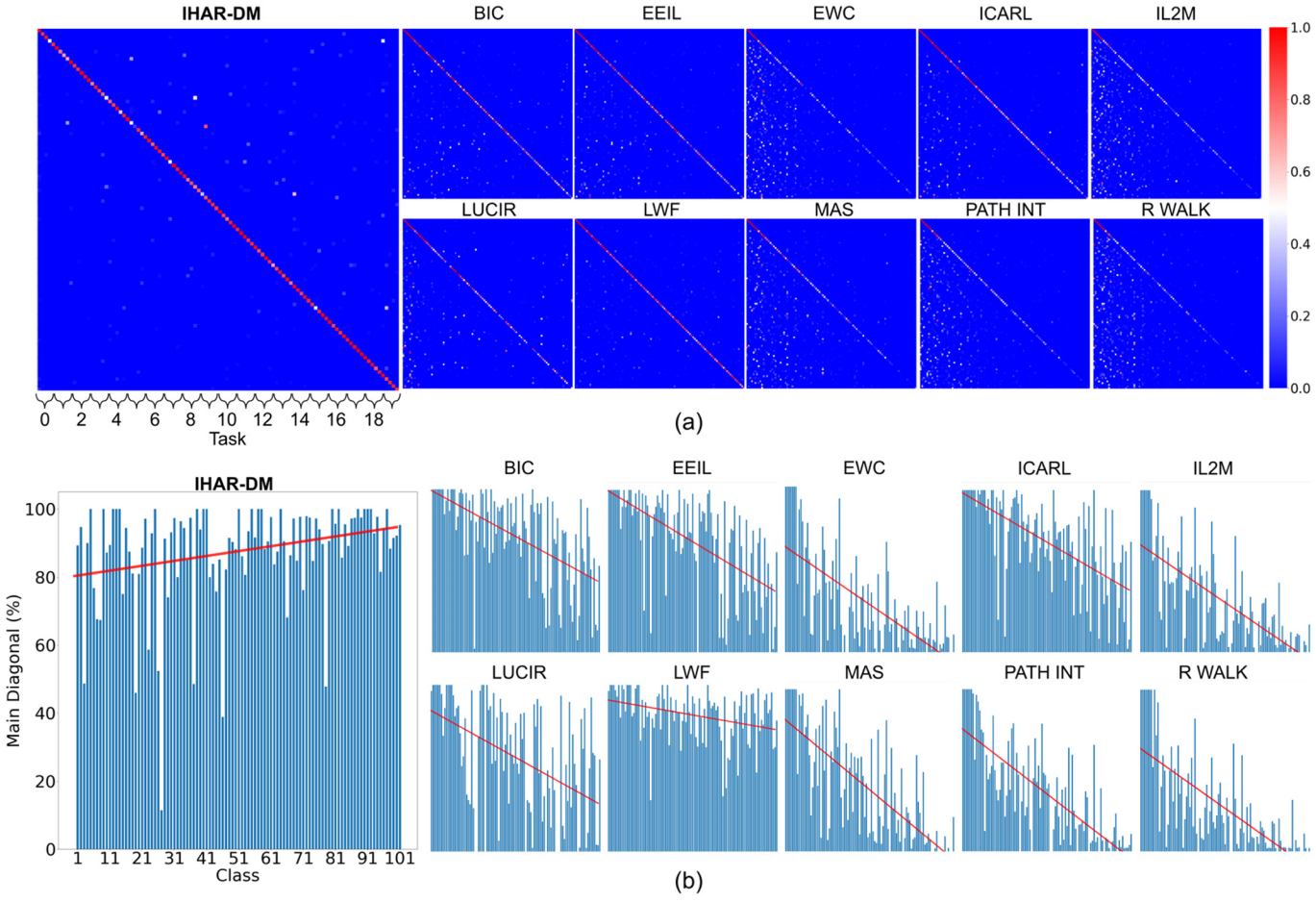

Fig. 8. (a) Confusion matrices obtained from a single experiment on the UCF-S1 setting. (b) Percentage of data on the main diagonal of the corresponding confusion matrix. The red line represents the tendency line, obtained through a Ridge Regression. Figure best viewed in color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

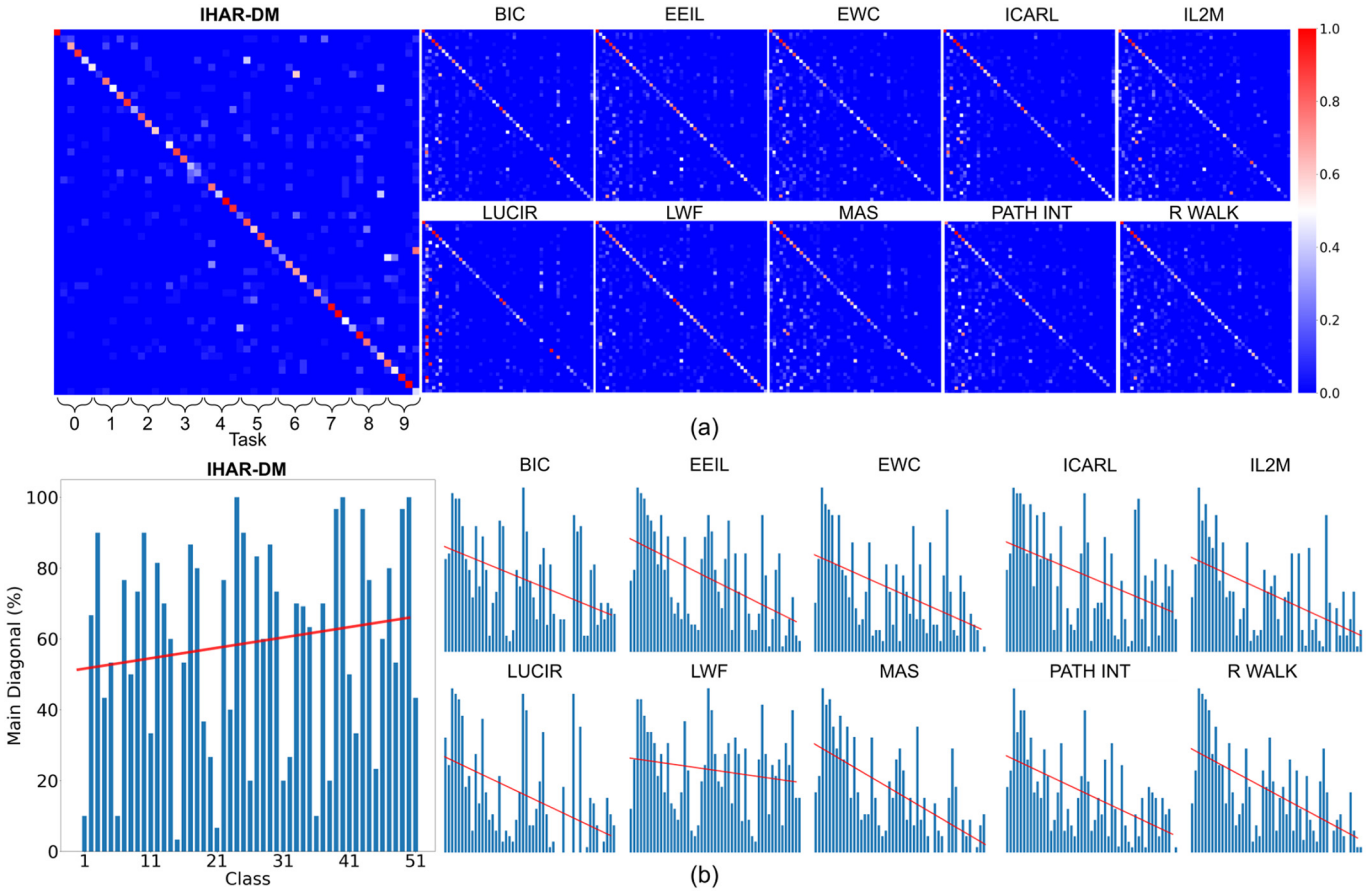

Fig. 9. (a) Confusion matrices obtained from a single experiment on the HMDB-S1 setting. (b) Percentage of data on the main diagonal of the corresponding confusion matrix. The red line represents the tendency line, obtained through a Ridge Regression. Figure best viewed in color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

preceding-task bias is overcompensating to alleviate Forgetting in the trade-off with Intransigence. The methods that did not suffer from *preceding-task bias* were LWF and IHAR-DM, where errors are evenly distributed across tasks. These methods were also the least impacted by performance decay as reflected by the ITI, suggesting an overcompensation to alleviate Forgetting while sacrificing the ability to maintain a stable performance as new classes were learned.

The main diagonal of most methods also shows a performance decay after the earlier tasks. A more detailed view of the main diagonal of the confusion matrices is shown in Fig. 8(b). Here, each class in the confusion matrix is represented by a vertical bar that indicates the percentage of data at a specific point of the main diagonal (correct classifications). In order to show the overall trend in performance as new tasks are learned, a linear interpolation was added to the plot, shown as a red line. It was obtained by fitting a Ridge regression [12]. This Figure shows that IHAR-DM was the only method that did not suffer from performance decay. The performance of IHAR-DM had a positive trend, suggesting that the proposed model became even more robust as new classes were learned. The same tendencies can be observed in Fig. 9. The figure shows the confusion matrices and main diagonals of a single experiment on the HMDB-S1 setting. Despite using a different and more challenging dataset, IHAR-DM still maintained a stable performance as new tasks were learned.

7. Conclusions and future works

This work presented a framework for performing Incremental HAR in videos. We introduced the dual-memory EVM, which synergizes with a Convolutional Neural Network and a Triplet Network to perform incremental learning coupled with deep representation learning.

The proposed method was compared with ten state-of-the-art incremental learning models. Results show that the proposed method achieved superior performance in terms of mean NMI. This result suggests that Metric Learning, as done by the Triplet Network, combined with a task-agnostic parameter isolation classifier (dual-memory EVM), forms a powerful framework for incremental video classification. Hence, representation learning should receive more attention in future research.

Regarding Forgetting and ITI, the proposed method achieved interesting results. Despite having a higher value of Forgetting, IHAR-DM did not suffer from performance decay as indicated by the ITI, unlike other methods. We observed that most methods tend to overcompensate for alleviating Forgetting, thus leading to a significant performance decay. It should be kept in mind that a stable performance is vital for real-world applications, where the number of tasks may grow indefinitely over time. In this sense, classifiers that suffer from performance decay quickly become obsolete.

Regarding the different experimental settings, we observed that the overall performance of most methods was lower in settings with a larger number of tasks. In terms of NMI, IHAR-DM was the least affected by this phenomenon. This may also be related to the performance decay measured by the ITI.

Future works include the extension of the Incremental HAR problem to open-world HAR. Open-world HAR is a challenging problem that involves dealing with previously unseen classes in unlabeled data. This

task requires differentiating between known and unknown data in a growing world. We believe that the dual-memory EVM coupled with a metric representation learning offers a promising framework for this task.

Declaration of Competing Interest

The authors Matheus Gutoski, Andre Eugenio Lazzaretti, and Heitor Silverio Lopes declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in the paper entitled "Incremental Human Action Recognition with Dual Memory".

Acknowledgment

Author M. Gutoski would like to thank CNPq for the scholarship number 141983/2018-3. Author H. S. Lopes would like to thank to CNPq for the research grant 311785/2019-0, and Fundação Araucária for grant PRONEX 042/2018. All authors would like to thank NVIDIA Corp. for the donation of the Titan-Xp GPUs used in the experiments.

Appendix A

In this section we provide the experimental results in table format to facilitate future comparisons. Table 3 shows the NMI and standard deviation of the experiments presented in Fig. 4. Table 4 shows the Forgetting and standard deviation of the experiments presented in Fig. 5, and Table 5 shows the ITI and standard deviation of the experiments presented in Fig. 7.

Table 3
Mean NMI and standard deviation between all tasks in five experimental settings. The values are equivalent to those of Fig. 4.

Setting	BIC	EEIL	EWC	IL2M	LUCIR	LWF	MAS	Path Int	R WALK	ICARL	**IHAR-DM**
UCF-S1	0.808 ± 0.078	0.792 ± 0.084	0.657 ± 0.096	0.661 ± 0.095	0.771 ± 0.083	0.843 ± 0.047	0.691 ± 0.102	0.666 ± 0.102	0.656 ± 0.102	0.807 ± 0.084	0.889 ± 0.023
UCF-S2	0.863 ± 0.048	0.857 ± 0.060	0.812 ± 0.071	0.817 ± 0.073	0.816 ± 0.081	0.889 ± 0.027	0.818 ± 0.088	0.818 ± 0.072	0.815 ± 0.077	0.861 ± 0.068	0.914 ± 0.014
UCF-S3	0.899 ± 0.039	0.898 ± 0.045	0.890 ± 0.045	0.891 ± 0.044	0.853 ± 0.079	0.915 ± 0.021	0.884 ± 0.059	0.891 ± 0.046	0.891 ± 0.048	0.896 ± 0.044	0.924 ± 0.009
HMDB-S1	0.589 ± 0.101	0.619 ± 0.101	0.586 ± 0.089	0.591 ± 0.085	0.597 ± 0.108	0.628 ± 0.079	0.598 ± 0.092	0.581 ± 0.087	0.580 ± 0.094	0.633 ± 0.101	0.662 ± 0.051
HMDB-S2	0.641 ± 0.096	0.674 ± 0.062	0.666 ± 0.066	0.674 ± 0.061	0.653 ± 0.104	0.696 ± 0.066	0.667 ± 0.073	0.663 ± 0.070	0.664 ± 0.074	0.684 ± 0.076	0.697 ± 0.046

Table 4
Mean Forgetting and standard deviation between all tasks in five experimental settings. The values are equivalent to those of Fig. 5.

Setting	BIC	EEIL	EWC	IL2M	LUCIR	LWF	MAS	Path Int	R WALK	ICARL	**IHAR-DM**
UCF-S1	0.014 ± 0.012	0.014 ± 0.017	0.062 ± 0.050	0.057 ± 0.053	0.010 ± 0.007	0.034 ± 0.021	0.041 ± 0.035	0.052 ± 0.041	0.058 ± 0.051	0.011 ± 0.004	0.049 ± 0.005
UCF-S2	0.019 ± 0.012	0.009 ± 0.004	0.025 ± 0.002	0.020 ± 0.003	0.002 ± 0.001	0.027 ± 0.004	0.013 ± 0.002	0.018 ± 0.005	0.019 ± 0.003	0.006 ± 0.001	0.032 ± 0.005
UCF-S3	0.018 ± 0.011	0.009 ± 0.012	0.016 ± 0.013	0.014 ± 0.015	0.001 ± 0.002	0.020 ± 0.017	0.008 ± 0.009	0.012 ± 0.014	0.010 ± 0.014	0.013 ± 0.010	0.024 ± 0.004
HMDB-S1	0.016 ± 0.019	0.018 ± 0.024	0.035 ± 0.037	0.029 ± 0.035	0.010 ± 0.014	0.037 ± 0.027	0.025 ± 0.036	0.029 ± 0.042	0.031 ± 0.033	0.011 ± 0.015	0.058 ± 0.062
HMDB-S2	0.016 ± 0.013	0.017 ± 0.031	0.026 ± 0.031	0.021 ± 0.031	0.008 ± 0.012	0.022 ± 0.019	0.020 ± 0.027	0.027 ± 0.029	0.022 ± 0.028	0.011 ± 0.015	0.026 ± 0.037

Table 5
Mean ITI and standard deviation between all tasks in five experimental settings. The values are equivalent to those of Fig. 7.

Setting	BIC	EEIL	EWC	IL2M	LUCIR	LWF	MAS	Path Int	R WALK	ICARL	**IHAR-DM**
UCF-S1	−0.013 ± 0.040	−0.016 ± 0.031	−0.021 ± 0.039	−0.020 ± 0.040	−0.018 ± 0.060	−0.006 ± 0.040	−0.019 ± 0.039	−0.022 ± 0.040	−0.021 ± 0.039	−0.015 ± 0.031	0.001 ± 0.030
UCF-S2	−0.015 ± 0.029	−0.021 ± 0.024	−0.027 ± 0.026	−0.026 ± 0.030	−0.030 ± 0.038	−0.009 ± 0.021	−0.032 ± 0.035	−0.026 ± 0.026	−0.029 ± 0.022	−0.023 ± 0.025	0.004 ± 0.016
UCF-S3	−0.028 ± 0.032	−0.030 ± 0.017	−0.030 ± 0.019	−0.030 ± 0.020	−0.051 ± 0.037	−0.014 ± 0.020	−0.040 ± 0.023	−0.031 ± 0.017	−0.032 ± 0.016	−0.029 ± 0.016	0.001 ± 0.013
HMDB-S1	−0.031 ± 0.092	−0.030 ± 0.103	−0.028 ± 0.077	−0.028 ± 0.076	−0.030 ± 0.111	−0.022 ± 0.077	−0.027 ± 0.084	−0.027 ± 0.072	−0.030 ± 0.075	−0.029 ± 0.098	−0.003 ± 0.081
HMDB-S2	−0.060 ± 0.096	−0.039 ± 0.071	−0.036 ± 0.096	−0.034 ± 0.085	−0.063 ± 0.113	−0.028 ± 0.108	−0.046 ± 0.094	−0.042 ± 0.097	−0.047 ± 0.096	−0.043 ± 0.100	−0.008 ± 0.083

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, T. Tuytelaars, Memory aware synapses: learning what (not) to forget, Proc. of the European Conference on Computer Vision (ECCV), Springer, Heidelberg 2018, pp. 139–154.

[2] R. Aljundi, P. Chakravarty, T. Tuytelaars, Expert gate: lifelong learning with a network of experts, Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Press, Piscataway, NJ 2017, pp. 3366–3375.

[3] E. Belouadah, A. Popescu, Il2m: class incremental learning with dual memory, Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Press, Piscataway, NJ 2019, pp. 583–592.

[4] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, Proc. of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Press, Piscataway, NJ 2017, pp. 4724–4733.

[5] F.M. Castro, M.J. Marín-Jiménez, N. Guil, C. Schmid, K. Alahari, End-to-end incremental learning, Proc. of the European Conference on Computer Vision (ECCV), Springer, Heidelberg 2018, pp. 233–248.

[6] A. Chaudhry, P.K. Dokania, T. Ajanthan, P.H. Torr, Riemannian walk for incremental learning: understanding forgetting and intransigence, Proc. of the European Conference on Computer Vision (ECCV), Springer, Heidelberg 2018, pp. 532–547.
[7] R. De Rosa, N. Cesa-Bianchi, I. Gori, F. Cuzzolin, Online action recognition via nonparametric incremental learning, Proceedings of the British Machine Vision Conference, BMVA Press, Guildford, Surrey, UK 2014, pp. 1–15.
[8] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: defying forgetting in classification tasks, IEEE Trans. Pattern Anal. Mach. Intell. (2021) 1–26.
[9] R.M. French, Catastrophic forgetting in connectionist networks, Trends Cognit. Sci. 3 (1999) 128–135.
[10] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Proc. of the 13th International Conference on Artificial Intelligence and Statistics, Microtome Publishing, Brookline, MA 2010, pp. 249–256.
[11] M. Gutoski, A.E. Lazzaretti, H.S. Lopes, Deep metric learning for open-set human action recognition in videos, Neural Comput. Appl. 33 (2021) 1207–1220.
[12] A.E. Hoerl, R.W. Kennard, Ridge regression: biased estimation for nonorthogonal problems, Technometrics 12 (1970) 55–67.
[13] E. Hoffer, N. Ailon, Deep metric learning using triplet network, Proc. of the International Workshop on Similarity-Based Pattern Recognition, Springer, Heidelberg 2015, pp. 84–92.
[14] S. Hou, X. Pan, C.C. Loy, Z. Wang, D. Lin, Learning a unified classifier incrementally via rebalancing, Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Press, Piscataway, NJ 2019, pp. 831–839.
[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. USA 114 (2017) 3521–3526.
[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, T. Serre, HMDB: a large video database for human motion recognition, Proceedings of the International Conference on Computer Vision (ICCV) 2011, pp. 2556–2563.
[17] Z. Li, D. Hoiem, Learning without forgetting, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2017) 2935–2947.
[18] Y. Lu, K. Boukharouba, J. Boonært, A. Fleury, S. Lecœuche, Application of an incremental svm algorithm for on-line human recognition from video surveillance using texture and color features, Neurocomputing 126 (2014) 132–140.
[19] A. Mallya, D. Davis, S. Lazebnik, Piggyback: adapting a single network to multiple tasks by learning to mask weights, Proc. of the European Conference on Computer Vision (ECCV), Springer, Heidelberg 2018, pp. 67–82.
[20] M. Masana, X. Liu, B. Twardowski, M. Menta, A.D. Bagdanov, J. van de Weijer, Class-Incremental Learning: Survey and Performance Evaluation, 2020 arXiv:2010.15277 (arXiv preprint).
[21] M. Masana, T. Tuytelaars, J. van Weijer, Ternary Feature Masks: Continual Learning Without Any Forgetting, 2020 arXiv:2001.08714 (arXiv preprint).
[22] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, J. Long, A survey of clustering with deep learning: from the perspective of network architecture, IEEE Access 6 (2018) 39501–39514.
[23] R. Minhas, A.A. Mohammed, Q.M.J. Wu, Incremental learning in human action recognition based on snippets, IEEE Trans. Circuits Syst. Video Technol. 22 (2012) 1529–1541.
[24] B. Pfülb, A. Gepperth, A comprehensive, application-oriented study of catastrophic forgetting in DNNs, Proc. of the International Conference on Learning Representations, OpenReview.net, Amherst, MA 2019, pp. 1–14.
[25] M.F. Pinto, A.G. Melo, A.L.M. Marcato, C.A. Moraes, Aerial human activity recognition through a cognitive architecture and a new automata proposal, Learn. Nonlinear Models 18 (2020) 4–14.
[26] J. Rajasegaran, S. Khan, M. Hayat, F.S. Khan, M. Shah, iTAML: an incremental task-agnostic meta-learning approach, Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Press, Piscataway, NJ 2020, pp. 13588–13597.
[27] S.A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, iCaRL: incremental classifier and representation learning, Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE Press, Piscataway, NJ 2017, pp. 2001–2010.
[28] K.K. Reddy, J. Liu, M. Shah, Incremental action recognition using feature-tree, Proc. of the 12th IEEE International Conference on Computer Vision, IEEE Press, Piscataway, NJ 2009, pp. 1010–1017.
[29] E.M. Rudd, L.P. Jain, W.J. Scheirer, T.E. Boult, The extreme value machine, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2018) 762–768.
[30] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive Neural Networks, 2016 arXiv:1606.04671 (arXiv preprint).
[31] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y.W. Teh, R. Pascanu, R. Hadsell, Progress & compress: a scalable framework for continual learning, Proc. of the International Conference on Machine Learning, PMLR 2018, pp. 4528–4537.
[32] Y. Shu, Y. Shi, Y. Wang, Y. Zou, Q. Yuan, Y. Tian, ODN: opening the deep network for open-set action recognition, Proc. of the IEEE International Conference on Multimedia and Expo (ICME), IEEE Press, Piscataway, NJ 2018, pp. 1–6.
[33] K. Soomro, A.R. Zamir, M. Shah, UCF101: A Dataset of 101 Human Actions Classes From Videos in the Wild, 2012 arXiv:1212.0402 (arXiv preprint).
[34] A. Strehl, J. Ghosh, Cluster ensembles–a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (2002) 583–617.
[35] C. Tang, W. Li, P. Wang, L. Wang, Online human action recognition based on incremental learning of weighted covariance descriptors, Inf. Sci. 467 (2018) 219–237.
[36] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, M. Paluri, A closer look at spatiotemporal convolutions for action recognition, Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Press, Piscataway, NJ 2018, pp. 6450–6459.
[37] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[38] D. Weinland, E. Boyer, R. Ronfard, Action recognition from arbitrary views using 3d exemplars, Proc. of the 2007 IEEE International Conference on Computer Vision, IEEE Press, Piscataway, NJ 2007, pp. 1–7.
[39] S.F. Wong, T.K. Kim, R. Cipolla, Learning motion categories using both semantic and structural information, Proc. of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE Press, Piscataway, NJ 2007, pp. 1–6.
[40] X. Wu, Y. Jia, W. Liang, Incremental discriminant-analysis of canonical correlations for action recognition, Pattern Recognit. 43 (2010) 4190–4197.
[41] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE Press, Piscataway, NJ 2019, pp. 374–382.
[42] S. Xie, C. Sun, J. Huang, Z. Tu, K. Murphy, Rethinking spatiotemporal feature learning: speed-accuracy trade-offs in video classification, Proc. of the European Conference on Computer Vision (ECCV), Springer, Heidelberg 2018, pp. 305–321.
[43] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, Proc. of the International Conference on Machine Learning, PMLR 2017, pp. 3987–3995.
[44] H.B. Zhang, Y.X. Zhang, B. Zhong, Q. Lei, L. Yang, J.X. Du, D.S. Chen, A comprehensive survey of vision-based human action recognition methods, Sensors 19 (2019).