Mask and Compress: Efficient Skeleton-based Action Recognition in Continual Learning

Matteo Mosconi[1 [0009−0008−1989−5779]], Andriy Sorokin[1 [0009−0002−3250−4249]], Aniello Panariello[1 [0000−0002−1940−7703]], Angelo Porrello[1 [0000−0002−9022−8484]], Jacopo Bonato[2 [0000−0001−6751−3407]], Marco Cotogni[2 [0000−0001−7950−7370]], Luigi Sabetta[2 [0000−0002−0865−5891]], Simone Calderara[1 [0000−0001−9056−1538]], and Rita Cucchiara[1 [0000−0002−2239−283X]]

[1] AImageLab - University of Modena and Reggio Emilia, Modena, Italy
name.surname@unimore.it
[2] Leonardo S.p.A.
name.surname.ext@leonardo.com

Abstract. The use of skeletal data allows deep learning models to perform action recognition efficiently and effectively. Herein, we believe that exploring this problem within the context of Continual Learning is crucial. While numerous studies focus on skeleton-based action recognition from a traditional offline perspective, only a handful venture into online approaches. In this respect, we introduce CHARON (Continual Human Action Recognition On skeletoNs), which maintains consistent performance while operating within an efficient framework. Through techniques like uniform sampling, interpolation, and a memory-efficient training stage based on masking, we achieve improved recognition accuracy while minimizing computational overhead. Our experiments on Split NTU-60 and the proposed Split NTU-120 datasets demonstrate that CHARON sets a new benchmark in this domain. The code is available at https://github.com/Sperimental3/CHARON.

Keywords: Continual Learning · Skeleton Based Action Recognition · Class Incremental Learning · Masked Autoencoder.

1 Introduction

Human Action Recognition (HAR) has become critical in various domains such as surveillance [27,29], rehabilitative healthcare [51], and sports analysis [23,39]. Early HAR approaches focused on exploiting RGB or gray-scale videos due to their widespread availability. However, recent advancements have explored alternative modalities, including skeletal joints [10,25,51], depth [36], point clouds [15], acceleration [24], and WiFi signals [42]. Among these, **skeleton-based action recognition** stands out as particularly efficient and concise, especially for actions not involving objects or scene context. Skeleton sequences capture the trajectory of key points (*i.e.*, joints) in the human body (*e.g.*, elbows, knees, wrists) [48]. As joints can be represented by 2D or 3D spatial coordinates,

skeletal data offer greater efficiency than images due to the sparsity of skeleton graphs. Moreover, this data structure is robust against changes in appearance, cluttered backgrounds, and occlusion while inherently privacy-preserving [42].

The traditional learning approach to HAR assumes that all necessary data is readily available during training. However, this assumption often does not hold in real-world contexts, as instances or classes may emerge incrementally over time. In such a dynamic context, Deep Neural Networks struggle to acquire new knowledge, often displacing the capabilities acquired during the previous stages. This phenomenon – widely known as *catastrophic forgetting* – leads to worse performance and is the focal point of **Continual Learning (CL)**. Specifically, in the CL setting, models must adapt to address a series of tasks presented sequentially, preserving performance on previously seen ones.

While tasks such as classification [22,35,5,46,41] and video-based action recognition [30,6,45] have been widely explored in a Continual Learning setting, skeleton-based HAR has been the subject of limited study in this domain. Although the authors of [26] have made efforts to address this task, they employ an expandable architecture, which can append a new learnable module to the network each time a new class arises. While such a technique aids in alleviating catastrophic forgetting, the computational footprint of the model gradually grows, making the approach memory-hungry and poorly scalable. Additionally, their setting adds constraints that diverge from real-world scenarios. Namely, they pre-train the network on most training instances and retain only a few classes for the incremental stage.

In this work, we exploit the structure of skeletal data to efficiently store samples in an episodic memory, *i.e.*, a continuously updated *buffer* containing a small subset of past data. Specifically, we enhance the memory efficiency of each sample, thus expanding the effective capacity of the buffer within the same memory allocation. We can do so as skeleton sequences present redundancy in time [23], so they can be compressed by sampling a subset of skeletal poses (*e.g.*, only one every s frames). This operation reduces the temporal resolution of the sequence with minimal information loss. Finally, in later tasks, we reconstruct each retained sample through linear interpolation, which remarkably does not require additional parameters.

We further exploit the redundancy of skeleton sequences by leveraging an approach based on Masked Image Modeling (MIM) [17,2,43]. Such self-supervised pre-training techniques have recently gained popularity due to the reduced wall-clock time and memory footprint. These methods pre-train a network by feeding it only a portion of the input data and reconstructing it with a lightweight decoder module. Once the pre-training is completed, they discard the decoder and feed the entire input to the model. However, unlike previous works [47,49], which employ masking techniques on skeletal data only for pre-training, our approach jointly optimizes both the self-reconstruction and the recognition tasks. Such a choice brings two benefits: *i)* the training time and memory requirements remarkably decrease, and *ii)* the additional reconstruction task acts as a regularizer for the encoder, leading to more meaningful representations.

Finally, at the end of each task, we introduce a *linear probing* phase to better conciliate the self-reconstruction approach with online scenarios. Indeed, if no countermeasures are involved, the encoder may suffer from a covariate shift issue [19] during inference, as it has been trained only on a portion of the input but is tested on the whole data. As reported in Sec. 3.2, this may be heavily detrimental to the final classification layer, specifically for high masking ratios. To mitigate such a problem, we freeze the encoder parameters and re-align the classifier in the presence of unmasked input sequences. This process is remarkably lightweight (*i.e.*, optimizing less than 4K parameters for Split NTU-60), yet significantly enhances overall performance.

To assess the proposed approach, we conduct a comprehensive evaluation on the incremental version of two popular datasets, NTU RGB+D 60 [38] and NTU RGB+D 120 [28], achieving state-of-the-art performance for class-incremental action recognition in the skeletons domain.

We remark on the following main contributions:

- We reduce the memory requirements of skeleton sequences in the buffer.
- We introduce a MIM approach for efficiently handling skeletal data in CL.
- We employ a linear probing phase to seamlessly integrate the encoder-decoder approach to the incremental learning setting.

2 Related works

Skeleton-based Action Recognition. In early skeleton-based action recognition works, sequences were treated as time series, thus processed employing Recurrent Neural Networks (RNNs) [11,53,18,8] to capture dynamics over time. These approaches struggled to integrate the spatial context of joints and proved slow and challenging to parallelize. Following works exploited Convolutional Neural Networks (CNNs) [21,20], treating skeletal data in various ways to make them compatible with CNNs; some handle coordinates as image channels [10,25], while others reshape skeletons by combining joints in space and time [20].

However, these models faced a common limitation: they failed to effectively represent the relationships between skeletal joints moving together in time. Graph Convolutional Networks (GCNs) resolve such shortcomings by exploiting nodes (*i.e.*, joints) temporally and spatially [50,12,13,7,40]. Subsequently, the emergence of ViT [9] marked the introduction of transformer-based architectures into computer vision, leading to solutions that integrate self-attention layers into convolutional architectures. One such work, STTFormer [31], divides the sequence in tuples of joints and retains some concepts of CNNs (*i.e.*, pooling aggregation) for in-time features processing. Nonetheless, such an approach under-exploits the sparsity and redundancy of skeletal data. In recent years, masking approaches [47,49] have been employed to take advantage of these characteristics for pre-training models. In contrast, our proposal adopts the reconstruction objective even during the optimization of the downstream task. Such a choice brings the benefit of reducing the training requirements of the whole pipeline, avoiding the pre-training phase.

Continual Learning. The Continual Learning setting makes a more realistic assumption w.r.t. standard learning paradigms. Specifically, data arrival is continuous and incremental. A subset of CL is Class-Incremental Learning (Class-IL) [44], where the dataset is re-arranged into multiple subsequent tasks, each containing a unique and disjoint set of classes. In this setting, the task identity is not known during inference.

Classical CL methods employ a regularization term that penalizes the alterations of weights to avoid forgetting [22,52,37]. Rehearsal methods [34,32,5,1], on the other hand, employ a limited memory buffer in which they store samples from past tasks and replay them. Another paradigm is represented by dynamic architectures [35,3] in which new network components are instantiated for each incoming task; unfortunately, this often leads to a rapid increase in the number of parameters. This approach has been employed by the authors of Else-Net [26] to tackle skeleton-based HAR in Class-IL. They use the first 50 classes of NTU RGB+D 60 to pre-train their network, and perform incremental training across 10 tasks, each focusing on a different class. We retain that such a benchmark diverges from classical CL ones, as it is simplified and far from real-world scenarios. In our work, we utilize the same setting presented by the authors of [4], who split NTU RGB+D 60 into 6 tasks, each involving *multiple* classes.

3 Method

3.1 Preliminaries

Class-Incremental Learning. In Class-IL, a deep model $f(\cdot;\theta)$ parametrized by θ is presented with a sequence of tasks $\mathcal{T}_i$ with $i \in \{1,\dots,T\}$, with T denoting the number of tasks. The i-th task provides N_i data entries $\{x_i^{(n)}, y_i^{(n)}\}_{n=1}^{N_i}$ with $y_i^{(n)} \in \mathcal{Y}_i$; importantly, each task relies on a set of classes disjoint from others s.t. $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset \iff i \neq j$. The objective of Class-IL is to minimize the empirical risk over all tasks:

$$\mathcal{L}_{\text{Class-IL}} = \sum_{i=1}^{T} \mathbb{E}_{(x,y)\sim\mathcal{T}_i} \left[\mathcal{L}(f(x;\theta), y)\right], \tag{1}$$

where $\mathcal{L}$ is the loss function (*e.g.*, the cross entropy for classification) and y is the ground truth label. Since the model observes one task at a time, tailored strategies are required to prevent catastrophic forgetting. Specifically, some rehearsal approaches [5,33] employ an additional regularization term $\mathcal{L}_{\mathcal{M}}$ exploiting samples stored in the memory buffer. The objective at the current task $\mathcal{T}_c$ is:

$$\hat{\mathcal{L}}_{\text{Class-IL}} = \mathbb{E}_{(x,y)\sim\mathcal{T}_c} \left[\mathcal{L}(f(x;\theta), y)\right] + \mathcal{L}_{\mathcal{M}}. \tag{2}$$

Spatio-Temporal Tuples Transformer (STTFormer). We adopt as main backbone of our architecture STTFormer [31], a transformer-based model designed for skeleton-based action recognition. It exploits self-attention to capture

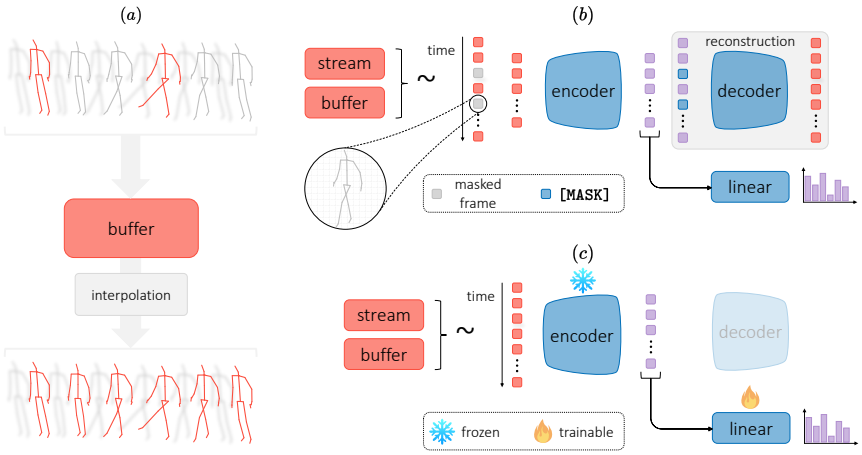

Fig. 1: Figure showing the key components of CHARON. Our efficient buffer strategy is shown on the left (a). In the upper right (b), we showcase the training phase with the reconstruction regularization, while linear probing is displayed at the bottom (c). Best seen in colors.

the cross-joint correlations across adjacent frames. Specifically, a raw skeleton sequence $x \in \mathbb{R}^{C \times F \times V}$, where C is the number of channels (*i.e.*, spatial coordinates), F the number of frames, and V the number of joints, is given as input to the model. This sample is divided into tuples, *i.e.*, sequences of n adjacent frames:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\lfloor F/n \rfloor}], \text{ where } \mathbf{x}_i \in \mathbb{R}^{C \times n \times V}. \tag{3}$$

Each layer of STTFormer comprises two distinct modules, which target either *intra*- or *inter*-tuple relationships. Every element of $\mathbf{X}$ (*i.e.*, each tuple) is first fed to a self-attention layer, which attends the joints in $\mathbf{x}_i$. This phase aims to model the *intra*-tuple characteristics. Then, an *inter*-tuple representation is extracted via temporal pooling.

3.2 CHARON

In this section, we present CHARON, which encompasses three components: *i)* a technique to populate the memory buffer, employing linear interpolation to decompress memory samples; *ii)* an efficient training phase with masked inputs; *iii)* a linear probing stage, which refines the classifier and updates the logits stored in the memory buffer. We depict these elements in Fig. 1.

Efficient buffer. A raw skeleton sequence $x \in \mathbb{R}^{C \times F \times V}$ collects the C coordinates (*e.g.*, *xyz* in Split NTU-60 and Split NTU-120) of V joints at F time instants. Unlike RGB video frames, skeletal data inherently reside in Euclidean

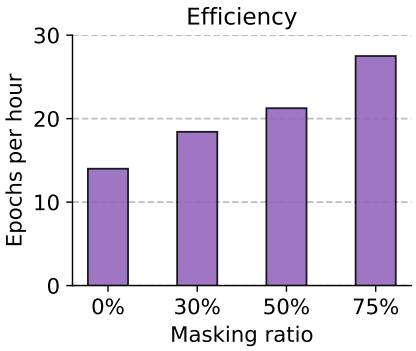

Fig. 2: Epochs per hour at different masking ratio values.[1]

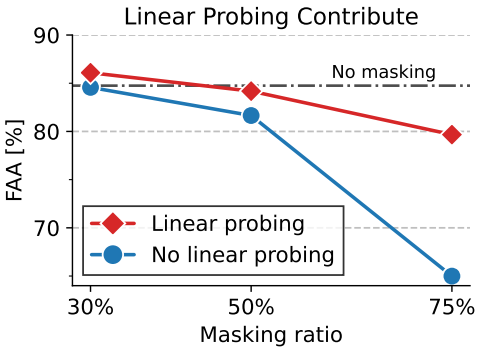

Fig. 3: Linear probing contribute on joint training with varying masking ratios.

space where the concept of distance between points (three-dimensional joints in our case) is well-defined. Additionally, skeleton sequences often exhibit temporal redundancy [16]. In light of these peculiarities, skeletal data can be easily compressed upon need: for instance, we do so before storing a sequence into the memory buffer. Notably, the compressed sequences can also be reconstructed with minimal loss through a simple linear interpolation. In particular, even with a sampling interval of $s = 5$ frames – *i.e.*, one kept every five, yielding a compression ratio of 80% – the reconstructions are close to the raw samples. Based on that, a greater number of instances can be stored within the same memory constraints: in other words, we can accumulate a number of samples s times larger in the buffer.

When a sample has to be extracted from the buffer for rehearsal, we reconstruct it to obtain F frames again and then treat it as a complete sample. It is noted that, since linear interpolation does not require learnable parameters, the reconstruction of temporal skeletal sequences requires low computational effort.

Training phase. As we mentioned above, a transformer-based architecture founded on [31] is adopted as our backbone. We build upon it to derive an encoder-decoder framework inspired by masked autoencoders [17]. Notably, this allows us to reduce the computational effort during training, as depicted in Fig. 2. Specifically, given a sample x coming from the current task or the buffer, the first step consists of a linear projection, followed by positional encoding to inject temporal dependencies. Afterward, we feed the encoder $e(\cdot;\theta_e)$ with a temporally masked sample $\tilde{x} \in \mathbb{R}^{C\times\lfloor(1-\eta)\cdot F\rfloor\times V}$ obtained by dropping a random subset of frames from the input sequence, where $\eta \in [0, 1)$ is the masking ratio.

The encoder projects the input $\tilde{x}$ into the latent space, obtaining features $\tilde{h} = e(\tilde{x};\theta_e)$. From this point, the architecture devises two branches: the first one (*recognition*) features a fully connected layer $f(\cdot;\theta_f)$ to yield pre-softmax logits $z = f(\tilde{h};\theta_f)$. The second branch (*reconstruction*) realizes the self-supervised

[1] Tests are performed on a single GTX 1080 Ti graphics card.

regularization through a decoder module $d(\cdot;\theta_d)$. Specifically, given the latent feature vector $\tilde{h}$ which has $\lfloor(1-\eta)\cdot F\rfloor$ tokens, the input of the decoder is formed by filling the missing ones with learnable mask vectors denoted with `[MASK]`. We place these vectors in the same position as the original masked ones, $h = \text{CONCAT}(\tilde{h}, \texttt{[MASK]})$. The training objective is:

$$\mathcal{L}_{\text{stream}} = \mathcal{L}_{\text{CE}}(z, y) + \gamma \cdot ||d(h;\theta_d) - x||_2^2, \tag{4}$$

where γ is a hyper-parameter weighting the impact of the reconstruction loss.

To mitigate forgetting, we incorporate the objective defined in Eq. (4) into a rehearsal-based framework. Drawing inspiration from [5], we retrieve a mini-batch of samples $x_{\mathcal{M}}$ from the memory buffer at each training step. This mini-batch includes associated predictions $z_{\mathcal{M}}$ (*i.e.*, logits) and labels $y_{\mathcal{M}}$, which are added to the episodic memory along with the corresponding samples. The loss functions for these two components are:

$$\mathcal{L}_{\text{logits}} = ||f(\tilde{h}_{\mathcal{M}};\theta_f) - z_{\mathcal{M}}||_2^2 + \gamma \cdot ||d(h_{\mathcal{M}};\theta_d) - x_{\mathcal{M}}||_2^2, \tag{5}$$

$$\mathcal{L}_{\text{labels}} = \mathcal{L}_{\text{CE}}(f(\tilde{h}_{\mathcal{M}};\theta_f), y_{\mathcal{M}}) + \gamma \cdot ||d(h_{\mathcal{M}};\theta_d) - x_{\mathcal{M}}||_2^2. \tag{6}$$

The mini-batch of samples $x_{\mathcal{M}}$ undergoes the same pipeline of the input stream x, producing the latent features $\tilde{h}_{\mathcal{M}} = e(\tilde{x}_{\mathcal{M}};\theta_e)$ and $h_{\mathcal{M}} = \text{CONCAT}(\tilde{h}_{\mathcal{M}}, \texttt{[MASK]})$.

The final objective of this phase is:

$$\mathcal{L} = \mathcal{L}_{\text{stream}} + \alpha \cdot \mathcal{L}_{\text{logits}} + \beta \cdot \mathcal{L}_{\text{labels}}, \tag{7}$$

where α and β are two balancing hyperparameters.

Linear probing. As described above, the model is trained with partial skeleton sequences. While providing an efficient training strategy, there is a factor that could hinder the overall performance during evaluation. Indeed, we argue that the classification heads $f(\cdot;\theta_f)$ could be subject to possible misalignment due to the different conditions we have at training (masking *on*) and test time (masking *off*). To address this issue, highlighted in Fig. 3, we devise an auxiliary linear probing stage at the end of each task, which lasts for a few epochs (*i.e.*, 10% of the number employed for the main training stage). During this phase, only the parameters of the classifier are allowed to change, while the encoder remains frozen. In doing so, we feed each full (*i.e.*, not masked) sample $x \in \mathbb{R}^{C\times F\times V}$ to the encoder.

In formal terms, as for the main training phase, the encoder projects the input x into the latent space obtaining hidden features $h = e(x;\theta_e)$. The fully connected linear layer $f(\cdot;\theta_f)$ produces then the logits $z = f(h;\theta_f)$ to which a cross-entropy loss is finally applied. In this phase, we still employ the regularization from [5]. Thus, the resulting objective $\mathcal{L}_{\text{lp}}$ can be written as:

$$\mathcal{L}_{\text{lp}} = \mathcal{L}_{\text{CE}}(z, y) + \alpha \cdot ||f(h_{\mathcal{M}};\theta_f) - z_{\mathcal{M}}||_2^2 + \beta \cdot \mathcal{L}_{\text{CE}}(f(h_{\mathcal{M}};\theta_f), y_{\mathcal{M}}). \tag{8}$$

Algorithm 1 Training CHARON at the current task

Requires: dataset $D_{\mathcal{T}_c}$, parameters θ $(\theta_e, \theta_f, \theta_d)$, scalars α, β and γ, learning rate λ, masking ratio η, buffer $\mathcal{M}$.
Main training phase:
for (x, y) **in** $D_{\mathcal{T}_c}$ **do**
  $(x_{\mathcal{M}}, y_{\mathcal{M}}, z_{\mathcal{M}}) \leftarrow \text{interpolate}(\text{extract}(\mathcal{M}))$
  $\tilde{x}, \tilde{x}_{\mathcal{M}} \leftarrow \text{random\_masking}(x, \eta), \text{random\_masking}(x_{\mathcal{M}}, \eta)$
  $\mathcal{L} \leftarrow$ Eqs. (4) to (7)
  $\theta \leftarrow \theta - \lambda \cdot \nabla_\theta \mathcal{L}$
end for
Linear probing:
for (x, y) **in** $D_{\mathcal{T}_c}$ **do**
  $(x_{\mathcal{M}}, y_{\mathcal{M}}, z_{\mathcal{M}}) \leftarrow \text{interpolate}(\text{extract}(\mathcal{M}))$
  $\mathcal{L} \leftarrow$ Eq. (8)
  $\theta_f \leftarrow \theta_f - \lambda \cdot \nabla_{\theta_f} \mathcal{L}$
  $\mathcal{M} \leftarrow \text{populate}(\mathcal{M}, (\text{uniform\_sampling}(x), z, y))$
end for

Traditional works using masked autoencoders [17,43] typically distinguish between a pre-train phase and one of linear probing to adapt to downstream tasks. However, we argue that leading these stages separately can result in a more cumbersome approach, potentially undermining the efficiency we seek. To solve this, Eqs. (7) and (8) are computed sequentially during each task, according to the incremental setting (*i.e.*, holding only a partial amount of data, the one belonging to the current task). The complete algorithmic procedure for a single task is described in Alg. 1.

4 Experimental analysis

4.1 Datasets

Split NTU-60 and Split NTU-120. NTU is one of the most popular benchmarks for action recognition on skeletal data. Initially comprising 60 classes and 56 578 samples in its original version [38], and later expanded to 120 classes and 113 945 samples [28], this dataset encompasses a diverse range of actions involving up to two individuals. The data collection process involves three Kinect cameras [54], positioned with different angles w.r.t. the subject. They provide RGB videos, IR videos, depth map sequences, and 3D skeletal data. Participants of various ages have contributed to the datasets construction, ensuring its broad applicability and relevance.

We adopt the extraction process employed by [31]. As original raw sequences contain a varying number of frames, we apply bilinear interpolation to obtain fixed-length sequences x (*i.e.*, 120 frames) s.t. $x \in \mathbb{R}^{(C=3)\times(F=120)\times(V=25)\times(B=2)}$. The axis identified by B regards the poses of the potentially two subjects involved in the action.

To test our approach in the CL scenario, we embrace Split NTU-60, introduced in [4], an incremental learning benchmark derived from the standard NTU dataset. The authors of [4] divide NTU RGB+D data into 6 tasks, each defining a 10-class classification problem. We also introduce Split NTU-120, an extension of the previous benchmark. Such a version brings a significant additional challenge, as seen in recent offline literature [14,12], leaving the way open for future works in the continual domain. To be as compliant as possible with the previous literature, we keep the original 6 tasks split and add another 6, each consisting of 10 classes, resulting in a 12 tasks incremental scenario. We describe in the supplementary material the exact order in which classes are split into tasks.

We report results for the cross-subject (XSub) and cross-view (XView) data modalities [38] for Split NTU-60, and cross-subject (XSub) and cross-setup (XSet) [28] for Split NTU-120.

4.2 Implementation details

The custom version we adopt for STTFormer [31] reduces the width of intermediate layers to obtain a more lightweight model. We set the number of frames in each tuple $n = 6$ as in the original paper. Following the asymmetric design proposed in [17], we employ 8 layers for the encoder and 3 for the decoder. We refer the reader to the supplementary material for further details. Additionally, we employ an α of 0.3 and a β of 0.8 for Eqs. (7) and (8), while we use a γ of 0.5 in approaches using the reconstruction regularization (Eqs. (4) to (6)). We adopt a batch size of 16 for all our experiments with a vanilla SGD optimizer and a learning rate of 0.05. Each task of the incremental setting lasts for 30 epochs. With the same hyperparameters as above, we perform 3 epochs for the linear probing phase. Finally, concerning data augmentation, we follow the original STTFormer implementation, applying a simple random rotation to each input sample.

4.3 Results

For the experimental comparison, we indicate with Joint Training (JT) the upper bound of our approach. It consists of training the model on the unified dataset (*i.e.*, without splitting it into tasks). For the lower bound, we adopt an incremental training approach that does not employ tailored techniques against catastrophic forgetting. We refer to it as Fine Tuning (FT).

In Tab. 1 we report the results for buffer sizes $\mathcal{M}_{size}$ of dimensions 500 and 2000. Following other works [5,26,4], we measure the recognition performance in terms of Final Average Accuracy (FAA), defined as:

$$\text{FAA} = \frac{1}{T}\sum_{i=1}^{T} a_{\mathcal{T}_i}, \tag{9}$$

where $a_{\mathcal{T}_i}$ is the accuracy of the i-th task after the model has seen all T of them. Additionally, we repeat each experiment three times, thus reporting the mean and standard deviation of the FAA.

Table 1: FAA (%) results on Split NTU-60 and Split NTU-120. For **CHARON**, we report the results with a masking ratio equal to 30%. We highlight in green the gains achieved by our approach w.r.t. the best competing method.

	Split NTU-60				**Split NTU-120**			
Method	**XView**		**XSub**		**XSet**		**XSub**	
FT	$16.05_{\pm 0.07}$		$15.64_{\pm 0.05}$		$7.19_{\pm 0.06}$		$6.97_{\pm 0.23}$	
JT	$84.75_{\pm 0.02}$		$77.32_{\pm 0.54}$		$71.18_{\pm 1.07}$		$70.15_{\pm 0.98}$	
$\mathcal{M}_{size}$	500	2000	500	2000	500	2000	500	2000
iCaRL	$51.54_{\pm 1.3}$	$53.41_{\pm 1.1}$	$47.12_{\pm 1.4}$	$50.69_{\pm 1.2}$	$32.91_{\pm 0.9}$	$34.74_{\pm 0.7}$	$33.03_{\pm 1.3}$	$36.68_{\pm 1.0}$
Else-Net	$40.81_{\pm 0.8}$	$59.10_{\pm 0.2}$	$39.72_{\pm 0.4}$	$57.00_{\pm 1.0}$	$19.37_{\pm 0.6}$	$33.52_{\pm 0.6}$	$18.43_{\pm 0.7}$	$33.95_{\pm 0.3}$
ER	$51.00_{\pm 1.6}$	$68.27_{\pm 0.1}$	$45.80_{\pm 0.5}$	$62.74_{\pm 1.9}$	$26.35_{\pm 1.1}$	$43.12_{\pm 0.4}$	$26.19_{\pm 1.7}$	$45.06_{\pm 0.7}$
DER	$51.36_{\pm 0.9}$	$66.74_{\pm 0.1}$	$49.97_{\pm 1.9}$	$63.48_{\pm 1.3}$	$27.83_{\pm 1.7}$	$40.19_{\pm 0.9}$	$30.10_{\pm 1.5}$	$36.10_{\pm 1.8}$
DER++	$60.41_{\pm 0.5}$	$73.09_{\pm 1.3}$	$57.22_{\pm 1.0}$	$67.64_{\pm 1.6}$	$34.27_{\pm 1.4}$	$50.06_{\pm 0.6}$	$36.29_{\pm 0.3}$	$49.81_{\pm 0.8}$
CHARON	**73.60**$_{\pm 0.3}$ **+13.19**	**77.77**$_{\pm 0.2}$ **+4.68**	**68.30**$_{\pm 0.6}$ **+11.08**	**72.70**$_{\pm 0.2}$ **+5.06**	**52.19**$_{\pm 0.6}$ **+17.92**	**61.63**$_{\pm 0.1}$ **+11.57**	**48.64**$_{\pm 0.0}$ **+12.35**	**59.23**$_{\pm 0.4}$ **+9.42**

As outlined by Tab. 1, the main competitor of this work, Else-Net [26], did not achieve performance comparable to those of the setting proposed by its authors, which devises a massive pre-training phase. Therefore, we can conclude that such a method suffers when trained from scratch.

Furthermore, even classical replay methods such as iCaRL [32], ER [33] and DER(++) [5] outperform Else-Net. CHARON reveals to be SOTA in the Class-IL skeleton-based action recognition domain, across both Split NTU-60 and Split NTU-120. In particular, this holds when employing a *masking ratio* of 30%; for higher percentages, we observe a decrease in performance, as discussed in the following. Significantly, the most substantial improvement is observed with a buffer size of 500 (surpassing the second-best, *i.e.*, DER++, when using a buffer size of 2000). This highlights the pivotal role of the sample quantity in the efficacy of replay methods. Consequently, it underscores the importance of researching techniques to increase sample numbers within a fixed buffer size.

On the sampling interval. To further evaluate the effectiveness of our buffer strategy, we conduct a comparative study on varying *sampling interval* s (which we recall indicates the step length in the uniform sampling procedure). Given $s \in \mathbb{N}^+$, we obtain the *compression ratio* as:

$$\textit{compression ratio} = \frac{s-1}{s} \cdot 100. \tag{10}$$

We report in Fig. 4 (*left*) the FAA at varying sampling interval s for both the buffer sizes tested. For each tested sampling interval, we scale the buffer size accordingly (as documented in Sec. 3.2). For instance, when $s = 10$, a memory with a nominal capacity of 500 examples could hence contain at most $s \cdot 500 = 5000$ (compressed) examples. As can be appreciated, the sampling interval $s = 5$ (*i.e.*, 80% of *compression*) yields the best results in terms of final accuracy. Namely, when sampling one skeletal pose every five frames, the memory buffer

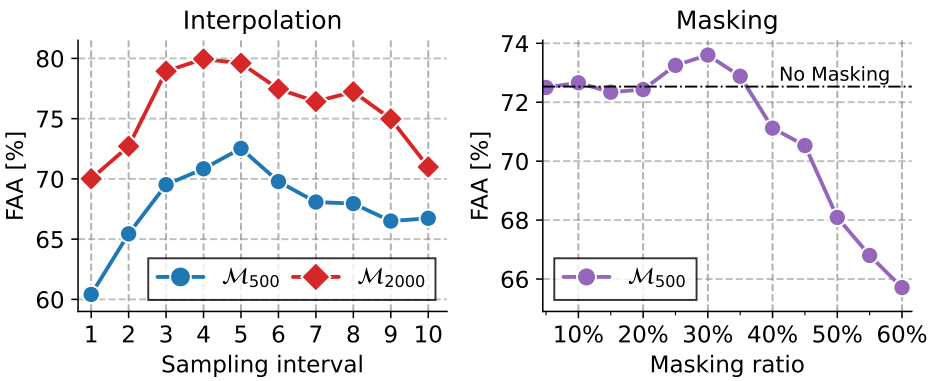

Fig. 4: (*left*) FAA for the DER++ baseline employing different values of the sampling interval s. (*right*) FAA obtained by CHARON as the masking ratio varies.

attains the best compromise between sample *fidelity* (which can be achieved with lower sampling intervals) and sample *diversity* (*i.e.*, higher intervals). Moreover, we note that the presence of a prior compression phase ($s > 1$) brings a stable and remarkable gain w.r.t. the standard replaying paradigm ($s = 1 \rightarrow$ no compression at all). Such a result shows the crucial role of the trade-off between the quality and quantity of samples.

On the masking ratio. We herein assess the impact of the *masking ratio*, which indicates the number of frames discarded before feeding the input sequence to the model. The results are illustrated in Fig. 4 (*right*) and reveal an increase in performance up to a value of 30%. For higher masking ratios, performance begins to decline, despite the notable efficiency gains (see Fig. 2). In quantitative terms, even with 50% of masking, CHARON achieves an acceptable final average accuracy of around 68%, while it decreases to $\approx 66\%$ with a masking ratio equal to 60%. Interestingly, both of these results are still higher than those of DER++, the second-best method reported in Tab. 1.

4.4 Ablations

We herein report the ablative studies; all the experiments are performed on the XView modality of Split NTU-60.

On the importance of the reconstruction-based objective. Our approach not only seeks good classification capabilities but also devises an auxiliary reconstruction term targeting the entire input sequence. To shed further light on the effects of such an auxiliary objective, we provide an ablative experiment in which we discard both the decoder module and the subsequent reconstruction loss. In doing so, we still apply random masking (testing two ratios equal to 30% and 60%) and linear probing at the end of each task.

The results of these ablative studies are reported in Tab. 2: remarkably, CHARON experiences a significant performance drop when removing the de-

Table 2: Impact of the reconstruction loss at different masking ratios.

	Masking ratio	
	30%	60%
w/o recon. loss	70.61	61.59
CHARON	**73.60**	**65.72**

Table 3: Ablative outcomes about sampling strategy and masking position.

	Position	
Strategy	*pre*	*post*
Deterministic	72.08	72.43
Random	71.89	**73.60**

coder and the reconstruction loss, especially for the higher masking ratio of 60%. We consider such a finding as noteworthy, as it highlights the importance of auxiliary learning techniques when leveraging higher compression ratios to pursue efficiency.

Masking strategy and positioning. Our approach adopts a masking strategy that builds upon random guessing to drop frames, thus following most of the literature dealing with masked autoencoders. Herein, we want to compare our approach with a deterministic strategy, that drops one frame every k. We also assess different possible positions to introduce the masking operation. Specifically, *post* indicates that masking is placed after splitting the sequence into tuples (see Sec. 3.1), as carried out by our approach. Results for the combinations of these two alternatives are reported in Tab. 3: as can be observed, the random strategy with post-hoc masking emerges as the best configuration.

5 Conclusions

Skeleton-based action recognition is a relevant task in modern human-centric Artificial Intelligence. We addressed such a long-standing computer vision task from the perspective of incremental learning, thus enabling those applications (*e.g.*, sports analysis, rehabilitative healthcare) where the set of actions to be recognized may change over time. Differently from existing proposals dealing with action recognition, our work appoints *efficiency* as a crucial aspect of an ideal incremental learner.

Our method, named CHARON, could be considered a step forward, as it achieves state-of-the-art performance with a remarkable reduction of the computational footprint (in terms of both memory and training time). In a few words, these capabilities derive from a proper application of input sub-sampling and random masking. Importantly, our experiments show that the addition of a reconstruction-based auxiliary objective grants further robustness in the presence of higher masking ratios, thus encompassing settings demanding efficiency. In future studies, we are going to deepen the concepts discussed in this paper, to apply our proposal even in the case of extreme masking (*e.g.*, up to 95%).

Acknowledgements Andriy Sorokin was supported by Marie Sklodowska-Curie Action Horizon 2020 (Grant agreement No. 955778) for the project "Personalized Robotics as Service Oriented Applications" ("PERSEO"). Additionally, the research activities of Angelo Porrello have been partially supported by the Department of Engineering "Enzo Ferrari" through the program FAR_2023_DIP – CUP E93C23000280005.

References

1. Arani, E., Sarfraz, F., Zonooz, B.: Learning fast, learning slow: A general continual learning method based on complementary learning system. In: International Conference on Learning Representations (2022)
2. Bao, H., Dong, L., Piao, S., Wei, F.: BEit: BERT pre-training of image transformers. In: International Conference on Learning Representations (2022)
3. Bonato, J., Pelosin, F., Sabetta, L., Nicolosi, A.: Mind: Multi-task incremental network distillation. In: Proceedings of the AAAI Conference on Artificial Intelligence (2024)
4. Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A., Calderara, S.: Class-incremental continual learning into the extended der-verse. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
5. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. Advances in Neural Information Processing Systems (2020)
6. Castagnolo, G., Spampinato, C., Rundo, F., Giordano, D., Palazzo, S.: A baseline on continual learning methods for video action recognition. arXiv preprint arXiv:2304.10335 (2023)
7. Chen, Y., Zhang, Z., Yuan, C., Li, B., Deng, Y., Hu, W.: Channel-wise topology refinement graph convolution for skeleton-based action recognition. In: IEEE International Conference on Computer Vision (2021)
8. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS Workshop on Deep Learning (2014)
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
10. Du, Y., Fu, Y., Wang, L.: Skeleton based action recognition with convolutional neural network. In: Proceedings of the Asian Conference on Computer Vision (2015)
11. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2015)
12. Duan, H., Wang, J., Chen, K., Lin, D.: Dg-stgcn: dynamic spatial-temporal modeling for skeleton-based action recognition. arXiv preprint arXiv:2210.05895 (2022)
13. Duan, H., Wang, J., Chen, K., Lin, D.: Pyskl: Towards good practices for skeleton action recognition. In: ACM International Conference on Multimedia (2022)
14. Duan, H., Zhao, Y., Chen, K., Lin, D., Dai, B.: Revisiting skeleton-based action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2022)

15. Fan, H., Yu, X., Ding, Y., Yang, Y., Kankanhalli, M.: Pstnet: Point spatio-temporal convolution on point cloud sequences. In: International Conference on Learning Representations (2021)
16. González-Aparicio, M.T., García, R., Brugos, J., Pañeda, X.G., Melendi, D., Cabrero, S.: Measuring temporal redundancy in sequences of video requests in a news-on-demand service. Telematics and Informatics (2014)
17. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2022)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation (1997)
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (2015)
20. Ke, Q., Bennamoun, M., An, S., Sohel, F., Boussaid, F.: A new representation of skeleton sequences for 3d action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2017)
21. Kim, T.S., Reiter, A.: Interpretable 3d human action analysis with temporal convolutional networks. IEEE International Conference on Computer Vision and Pattern Recognition Workshops (2017)
22. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences (2017)
23. Kong, Y., Fu, Y.: Human action recognition and prediction: A survey. International Journal of Computer Vision (2022)
24. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SigKDD Explorations Newsletter (2011)
25. Li, C., Zhong, Q., Xie, D., Pu, S.: Skeleton-based action recognition with convolutional neural networks. In: IEEE International Conference on Multimedia and Expo Workshops (2017)
26. Li, T., Ke, Q., Rahmani, H., Ho, R.E., Ding, H., Liu, J.: Else-net: Elastic semantic network for continual action recognition from skeleton data. In: IEEE International Conference on Computer Vision (2021)
27. Lin, W., Sun, M.T., Poovandran, R., Zhang, Z.: Human activity recognition for video surveillance. In: IEEE international symposium on circuits and systems (ISCAS) (2008)
28. Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L.Y., Kot, A.C.: Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
29. Panariello, A., Porrello, A., Calderara, S., Cucchiara, R.: Consistency based self-supervised learning for temporal anomaly localization. In: European Conference on Computer Vision Workshops (2022)
30. Park, J., Kang, M., Han, B.: Class-incremental learning for action recognition in videos. In: IEEE International Conference on Computer Vision (2021)
31. Qiu, H., Hou, B., Ren, B., Zhang, X.: Spatio-temporal tuples transformer for skeleton-based action recognition. arXiv preprint arXiv:2201.02849 (2022)
32. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2017)

33. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., , Tesauro, G.: Learning to learn without forgetting by maximizing transfer and minimizing interference. In: International Conference on Learning Representations (2019)
34. Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science (1995)
35. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
36. Sanchez-Caballero, A., Fuentes-Jimenez, D., Losada-Gutiérrez, C.: Exploiting the convlstm: Human action recognition using raw depth video-based recurrent neural networks. arXiv preprint arXiv:2006.07744 (2020)
37. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International Conference on Machine Learning (2018)
38. Shahroudy, A., Liu, J., Ng, T., Wang, G.: Ntu rgb+d: A large scale dataset for 3d human activity analysis. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2016)
39. Shao, D., Zhao, Y., Dai, B., Lin, D.: Finegym: A hierarchical video dataset for fine-grained action understanding. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2020)
40. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition. In: Proceedings of the Asian Conference on Computer Vision (2020)
41. Smith, J.S., Karlinsky, L., Gutta, V., Cascante-Bonilla, P., Kim, D., Arbelle, A., Panda, R., Feris, R., Kira, Z.: Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2023)
42. Sun, Z., Ke, Q., Rahmani, H., Bennamoun, M., Wang, G., Liu, J.: Human action recognition from various data modalities: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
43. Tong, Z., Song, Y., Wang, J., Wang, L.: Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. Advances in Neural Information Processing Systems (2022)
44. Van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv preprint arXiv:1904.07734 (2019)
45. Villa, A., Alcázar, J.L., Alfarra, M., Alhamoud, K., Hurtado, J., Heilbron, F.C., Soto, A., Ghanem, B.: Pivot: Prompting for video continual learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2023)
46. Wang, Z., Zhang, Z., Lee, C.Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., Pfister, T.: Learning to prompt for continual learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2022)
47. Wu, W., Hua, Y., Zheng, C., Wu, S., Chen, C., Lu, A.: Skeletonmae: Spatial-temporal masked autoencoders for self-supervised skeleton action recognition. In: IEEE International Conference on Multimedia and Expo Workshops (2023)
48. Xin, W., Liu, R., Liu, Y., Chen, Y., Yu, W., Miao, Q.: Transformer for skeleton-based action recognition: A review of recent advances. Neurocomputing (2023)
49. Yan, H., Liu, Y., Wei, Y., Li, Z., Li, G., Lin, L.: Skeletonmae: graph-based masked autoencoder for skeleton sequence pre-training. In: IEEE International Conference on Computer Vision (2023)

50. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence (2018)
51. Yin, J., Han, J., Wang, C., Zhang, B., Zeng, X.: A skeleton-based action recognition system for medical condition detection. In: IEEE Biomedical Circuits and Systems Conference (BioCAS) (2019)
52. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning (2017)
53. Zhang, S., Yang, Y., Xiao, J., Liu, X., Yang, Y., Xie, D., Zhuang, Y.: Fusing geometric features for skeleton-based action recognition using multilayer lstm networks. IEEE Transactions on Multimedia (2018)
54. Zhang, Z.: Microsoft kinect sensor and its effect. IEEE MultiMedia (2012)

6 Split NTU-120

We hereby present Split NTU-120, the incremental version of the popular dataset introduced in [28]. Adhering to the structure delineated in [4], which entails 6 tasks split into 10 classes each, we expand upon it by incorporating additional 6 tasks. This results in a total of 12 tasks in an incremental scenario, each comprising a 10-classes classification problem. Below, we present the order of these classes within each task.

Task 1 *put on glasses, cross hands in front, falling down, staggering, fan self, salute, throw, punch/slap, pushing, put palms together;*
Task 2 *hand waving, sneeze/cough, type on a keyboard, nod head/bow, check time (from watch), brush teeth, hugging, wipe face, eat meal, shaking hands;*
Task 3 *put on jacket, reading, take off a shoe, put on a shoe, tear up paper, reach into pocket, point to something, drink water, kicking something, sit down;*
Task 4 *headache, pat on back, play with phone/tablet, writing, stand up, back pain, walking towards, shake head, walking apart, touch pocket;*
Task 5 *take off glasses, point finger, brush hair, taking a selfie, giving object, take off jacket, take off a hat/cap, kicking, phone call, hopping;*
Task 6 *rub two hands, nausea/vomiting, jump up, clapping, drop, chest pain, neck pain, put on a hat/cap, cheer up, pick up;*
Task 7 *cutting nails, yawn, open a box, flick hair, cutting paper, hit with object, hush, knock over, make victory sign, put on bag;*
Task 8 *toss a coin, follow, cross arms, cross toe touch, capitulate, shake fist, support somebody, move heavy objects, whisper, put on headphone;*
Task 9 *take a photo, tennis bat swing, shoot with gun, sniff/smell, thumb down, butt kicks, take object out of bag, counting money, grab stuff, play magic cube;*
Task 10 *snap fingers, shoot at basket, blow nose, cheers and drink, side kick, exchange things, apply cream on face, step on foot, bounce ball, throw up cap/hat;*
Task 11 *high-five, take off headphone, fold paper, arm swings, rock-paper-scissors, make OK sign, squat down, carry object, stretch oneself, thumb up;*
Task 12 *wield knife, staple book, arm circles, put object into bag, run on the spot, ball up paper, juggle table tennis ball, open bottle, apply cream on hand, take off bag.*

7 Skeleton-based MAE

In Tab. A, we present the output dimensions and layer specifications of our Masked AutoEncoder-inspired skeleton-based architecture. As outlined in the paper, we utilize a modified version of STTFormer [31] for the backbone, which is reduced in width. The `qkv_dim` parameter determines the embedding dimensions of the queries, keys, and values, respectively. The number of attention heads in each layer is consistently set at 3. Regarding the IFFA modules, which incorporate temporal pooling aggregation, we opt for a default kernel size of 3×1 and a padding of $(1, 0)$. Lastly, the linear classifier is intended for the 60 classes of Split NTU-60.

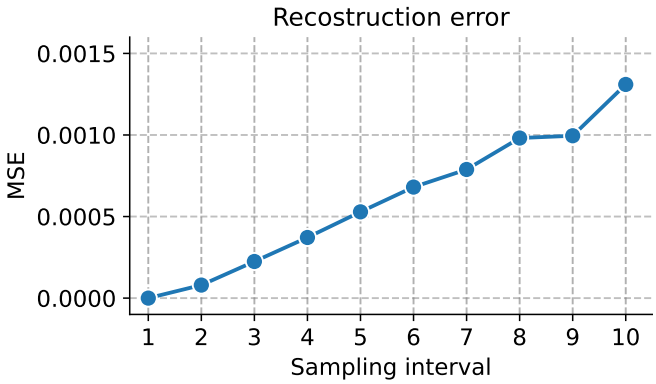

Fig. A: Figure showing the reconstruction error of the interpolation procedure as the sampling interval varies.

8 Interpolation reconstruction error

We display in Fig. A the reconstruction error of the interpolation procedure as the sampling interval varies. To quantify the distance between the reconstructed and the ground truth samples, we employ a simple yet effective Mean Squared Error (MSE) metric.

Table A: The skeleton-based masked autoencoder architecture adopted.

sklMAE - Input Size: $3 \times 120 \times 25$		
Layer Name	**Output Size**	**Layer Details**
`division_in_tuples`	$3 \times 20 \times 150$	`reshape(shape[0], shape[1] // 6, shape[2] * 6)`
`input_map`	$32 \times 20 \times 150$	Conv2D 1×1, 32 BatchNorm2D
`masking`	$32 \times ((1-\eta) \cdot 20) \times 150$	Random masking, $\eta \in [0, 1)$
`encoder_1`	$32 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 16$
`encoder_2`	$32 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 16$
`encoder_3`	$64 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 32$
`encoder_4`	$64 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 32$
`encoder_5`	$128 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 64$
`encoder_6`	$128 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 64$
`encoder_7`	$128 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 64$
`encoder_8`	$128 \times ((1-\eta) \cdot 20) \times 150$	STTFormer layer, `qkv_dim` $= 64$
`classifier`	128×60	Average pooling Linear layer
`mask_concat`	$128 \times 20 \times 150$	Concatenation of the `[MASK]` tokens
`decoder_1`	$64 \times 20 \times 150$	STTFormer layer, `qkv_dim` $= 64$
`decoder_2`	$32 \times 20 \times 150$	STTFormer layer, `qkv_dim` $= 32$
`decoder_3`	$32 \times 20 \times 150$	STTFormer layer, `qkv_dim` $= 16$
`output_map`	$3 \times 20 \times 150$	Conv2D 1×1, 3