Online Skeleton-based Action Recognition with Continual Spatio-Temporal Graph Convolutional Networks

Lukas Hedegaard, Negar Heidari, and Alexandros Iosifidis
Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm, negar.heidari, ai}@ece.au.dk

***Abstract*—Graph-based reasoning over skeleton data has emerged as a promising approach for human action recognition. However, the application of prior graph-based methods, which predominantly employ whole temporal sequences as their input, to the setting of online inference entails considerable computational redundancy. In this paper, we tackle this issue by reformulating the Spatio-Temporal Graph Convolutional Neural Network as a Continual Inference Network, which can perform step-by-step predictions in time without repeat frame processing. To evaluate our method, we create a continual version of ST-GCN, *Co*ST-GCN, alongside two derived methods with different self-attention mechanisms, *Co*AGCN and *Co*S-TR. We investigate weight transfer strategies and architectural modifications for inference acceleration, and perform experiments on the NTU RGB+D 60, NTU RGB+D 120, and Kinetics Skeleton 400 datasets. Retaining similar predictive accuracy, we observe up to 109× reduction in time complexity, on-hardware accelerations of 26×, and reductions in maximum allocated memory of 52% during online inference.**

***Index Terms*—Continual Inference Networks, Graph-Convolution, Attention, Convolutional Neural Network, Skeleton-based Action Recognition, Human Activity Recognition, Online Inference**

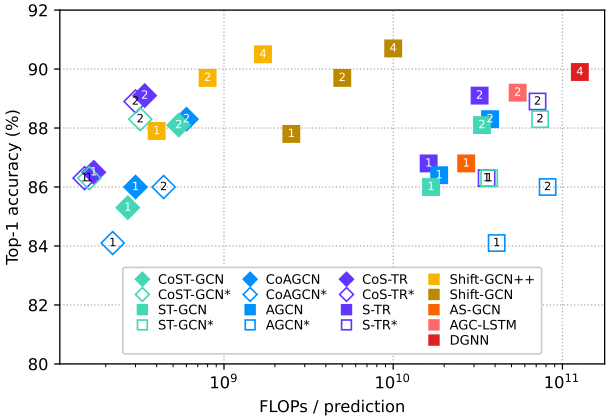

Fig. 1: **Accuracy/complexity trade-off** on NTU RGB+D 60 X-Sub for ◆ *Continual* and ■ prior methods during online inference. Numbers denote streams for each method. *Architecture modification with stride one and no padding.

I. Introduction

A human action can be described by a temporal sequence of human body poses, each of which is represented by a set of spatial joint coordinates forming a body skeleton. Accordingly, skeleton-based action recognition methods process a sequence of skeletons (instead of an image sequence) to recognize the performed action. Compared with predicting actions from videos, a sequence of skeleton data not only gives the spatial and temporal features of the body poses, but also provides robustness against different background variations and context noise [1]. The estimation of such skeletal data has become a staple in the human action recognition toolkit thanks to publicly available toolboxes such as OpenPose [2].

Early deep learning methods for skeleton-based action recognition either rearrange the body joint coordinates of each skeleton to make a pseudo-image which is used to train a CNN model [3, 4, 5, 6, 7, 8], or concatenate the human body joints as a sequence of feature vectors and train a RNN model [9, 10, 11, 12, 13, 14]. However, these methods cannot take advantage of the non-Euclidean structure of the skeletons. Recently, Graph Convolutional Networks (GCNs) have shown prowess in the modeling of skeleton data. ST-GCN [15] was the first GCN-based method proposed for skeleton-based action recognition. It uses spatial graph convolutions to extract the per time-step features of each skeleton and employs temporal convolutions to capture time-varying dynamics throughout the skeleton sequence. Since its publication, several methods have sprung from ST-GCN, which enhance feature extraction or optimize the structure of the model.

2s-AGCN [16] proposed to learn the graph structure in each GCN layer adaptively based on input graph node similarity and also utilized an attention method which highlights both the existing spatial connections in the graph (bones) and new potential connections between them. MS-AAGCN [17] extended 2s-AGCN by proposing a multi-stream framework which uses four different data streams for training the model. Moreover it enhanced the adaptive graph convolution in 2s-AGCN with a spatio-temporal channel attention module to highlight the most important skeletons, nodes in each skeleton, and features of each node. DGNN [18] modeled the spatial connections between the graph nodes with a directed graph and utilized both node features and edge features simultaneously. Hyper-GNN [19] captured the non-physical connections between the nodes by constructing hyperedges which help to extract both

local and global features in each graph. FGCN [20] proposed to extract coarse to fine spatio-temporal features by a multi-stage temporal sampling strategy and introduced a feedback mechanism in graph convolution to transfer the high-level features to the shallower layers of the network. Similarly, MS-G3D [21] has proposed multi-scale graph convolutions for long-range feature extraction.

Unfortunately, the high computational complexity of these GCN-based methods makes them infeasible in real-time applications and resource-constrained online inference settings. Multiple approaches have been explored to increase the efficiency of skeleton-based action recognition recently: GCN-NAS [22] and PST-GCN [23] are neural architecture search based methods which try to find an optimized ST-GCN architecture to increase the efficiency of the classification task; ShiftGCN [24] replaces graph and temporal convolutions with a zero-FLOPs shift graph operation and point-wise convolutions as an efficient alternative to the feature-propagation rule for GCNs [25]; ShiftGCN++[26] boost the efficiency of ShiftGCN further via progressive architecture search, knowledge-distillation, explicit spatial positional encodings, and a Dynamic Shift Graph Convolution; SGN [27] utilizes semantic information such as joint type and frame index as side information to design a compact semantics-guided neural network (SGN) for capturing both spatial and temporal correlations in joint and frame level; TA-GCN [28] tries to make inference more efficient by selecting a subset of key skeletons, which hold the most important features for action recognition, from a sequence to be processed by the spatio-temporal convolutions.

Yet, none of the above-described GCN-based methods are tailored to online inference, were the input is a continual stream of skeletons and step-by-step predictions are required. During online inference, these methods would need to rely on sliding window-based processing, i.e., storing the $T-1$ prior skeletons, appending the newest skeleton to get a sequence of length T, and then performing their prediction on the whole sequence. In this paper, we reduce such redundant computations by reformulating the ST-GCN and its derived methods as a Continual Inference Network, which processes skeletons one by one and produces updated predictions for each time-step without the need to include past skeletons in every input as is the case for the prior GCN-based methods. This is achieved by using Continual Convolutions in place of regular ones for aggregating temporal information. In particular, we propose the *Continual* Spatio-Temporal Graph Convolutional Network (*Co*ST-GCN), *Co*AGCN, and *Co*TR-S and evaluate them on the skeleton-based action recognition datasets NTU RGB+D 60 [29], NTU RGB+D 120 [30], and Kinetics Skeleton 400 [31] with striking results: Our continual models achieve up to 108× FLOPs reduction, 26× speedup, and 52% reduction in max allocated GPU memory compared to the corresponding non-continual models.

The remainder of the paper is structured as follows: Section II provides an introduction to skeleton-based action recognition and of the related methods, from which we derive a continual counterpart, Section III describes Continual Inference Networks, and Section IV presents our proposed *Continual* Spatio-temporal Graph Convolutional Networks. Experiments on weight transfer strategies, performance benchmarks, and comparisons with prior works are offered in Section V, and a conclusion is given in Section VI.

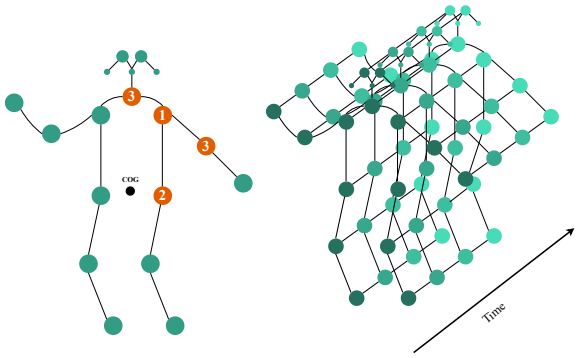

Fig. 2: **Graph illustration** for a spatially partitioned skeleton (left) and spatio-temporal graph (right).

II. Related Works

A. Spatio-Temporal Graph Convolutional Network

GCN-based models for skeleton-based action recognition [15, 16, 18, 22, 23, 27, 28] operate on sequences of skeleton graphs. The spatio-temporal graph of skeletons $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has the human body joint coordinates as nodes $\mathcal{V}$ and the spatial and temporal connections between them as edges $\mathcal{E}$. Figure 2 (right) illustrates such a spatio-temporal graph where the spatial graph edges encode the human bones and the temporal edges connect the same joints in subsequent time-steps. We model this graph as a tensor $\mathbf{X} \in \mathbb{R}^{C^{(0)} \times T \times V}$, where $C^{(0)}$ is the number of input-channels of each joint, T denotes the number of skeletons in a sequence, and V is the number of joints in each skeleton. A binary adjacency matrix $\mathbf{A} \in \mathbb{R}^{V \times V}$ encodes the skeleton-structure with ones in positions connecting two vertices in a skeleton and zeros elsewhere.

The ST-GCN [15] and AGCN [16] methods refine the spatial structure of each skeleton by employing a partitioning method which categorizes neighboring nodes of each body joint into three subsets: (1) the root node itself, (2) the root's neighboring nodes which are closer to the skeleton's center of gravity (COG) than the root itself, and (3) the remaining neighboring nodes of the root node. An example of this subset partitioning is shown in Figure 2 (left). Accordingly, the graph-structure of each skeleton is represented by three normalized binary adjacency matrices $\left\{\mathbf{A}_p \in \mathbb{R}^{V \times V} \mid p = 1, 2, 3\right\}$, each of which is defined as

$$\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}, \tag{1}$$

where $\mathbf{D}_p$ denotes the degree matrix of the neighboring subset p. Inspired by the GCN aggregation rule [25], the spatial graph convolution receives the hidden representation of the previous

layer $\mathbf{H}^{(l-1)}$ as input, where $\mathbf{H}^{(0)} = \mathbf{X}$, and performs the following graph convolution (GC) transformation:

$$\text{GC}\left(\mathbf{H}^{(l-1)}\right) = \sigma\left(\text{Res}(\mathbf{H}^{(l-1)}) + \text{BN}\left(\sum_p (\hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)})\mathbf{H}^{(l-1)}\mathbf{W}_p^{(l)}\right)\right) \tag{2}$$

where $\sigma(\cdot)$ denotes a ReLU non-linearity, $\mathbf{W}_p^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$ is the weight matrix which transforms the features of the neighboring subset p and $BN(\cdot)$ denotes batch normalization. Moreover, a learnable matrix $\mathbf{M}_p^{(l)} \in \mathbb{R}^{V \times V}$ is multiplied element-wise with its corresponding adjacency matrix $\hat{\mathbf{A}}_p$ as an attention mechanism that highlights the most important connections in each spatial graph. In order to retain the model's stability, the input to a layer is added to the transformed features through a residual connection $\text{Res}(\mathbf{H}^{(l-1)})$ which is defined as:

$$\text{Res}(\mathbf{H}^{(l-1)}) = \begin{cases} \mathbf{H}^{(l-1)}, & C^{(l)} = C^{(l-1)}, \\ \mathbf{H}^{(l-1)}\mathbf{W}_{res}^{(l)}, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathbf{W}_{res}^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$ is a learnable mapping matrix which transforms the layer's input to have the same channel dimension as the layer's output.

The graph convolution block is followed by a temporal convolution, $\text{TC}(\cdot)$, which propagates the features of the graph nodes through different time steps to capture the motions taking place in an action. In the temporal graph, each node only has two fixed neighbors which are its corresponding nodes in the previous and next skeletons. The adjacency matrices and partitioning process are not involved in temporal feature propagation. In practice, the temporal convolution is a standard 2D convolution which receives the output of the graph convolution obtained in Eq. 2 and performs a transformation with a kernel of size $C^{(l)} \times K \times 1$ to keep the node feature dimension unchanged and aggregate the features through K consecutive time steps.

The whole spatio-temporal convolution block has the form

$$\mathbf{H}^{(l)} = \sigma\left(\text{Res}(\mathbf{H}^{(l-1)}) + \text{BN}(\text{TC}(\text{GC}(\mathbf{H}^{(l-1)})))\right). \tag{4}$$

The ST-GCN model is composed of multiple such spatio-temporal convolutional blocks. A global average pool and fully connected layer perform the final classification.

B. Adaptive Graph Convolutional Neural Networks

The fixed graph structure used in Eq. (2) is defined based on natural connections in the human body skeleton which restricts the model's capacity and flexibility in representing different action classes. However, for some action classes such as "touching head" it makes sense to model a connection between hand and head even though such a connection is not naturally present in the skeleton. AGCN [16] allows for such possibilities by adopting an adaptive graph convolution which utilizes a data-dependent graph structure as follows:

$$\text{AGC}\left(\mathbf{H}^{(l-1)}\right) = \sigma\left(\text{Res}(\mathbf{H}^{(l-1)}) + \text{BN}\left(\sum_p (\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)})\mathbf{H}^{(l-1)}\mathbf{W}_p^{(l)}\right)\right), \tag{5}$$

where $\mathbf{M}_p^{(l)}$ is defined as:

$$\mathbf{M}_p^{(l)} = \mathbf{B}_p^{(l)} + \mathbf{C}_p^{(l)} \tag{6}$$

The attention matrix in this definition is composed of two learnable matrices which are optimized along with other model parameters in an end-to-end manner. $\mathbf{B}_p^{(l)} \in \mathbb{R}^{N \times N}$ is a squared matrix that can be unique for each layer and each sample, and $\mathbf{C}_p^{(l)} \in \mathbb{R}^{N \times N}$ is a similarity matrix whose elements determine the strength of the pair-wise connections between nodes. This matrix is computed by first transforming the feature matrix $\mathbf{H}^{(l-1)} \in \mathbb{R}^{C^{(l-1)} \times T \times V}$ with two embedding matrices $\mathbf{W}_{p\theta}^{(l)}$, $\mathbf{W}_{p\phi}^{(l)}$ of size $C^{de} \times C^{(l-1)}$. The obtained feature maps are then reshaped to $C^{de}T \times V$ and multiplied to obtain the $\mathbf{C}_p^{(l)} \in \mathbb{R}^{N \times N}$ matrix as follows:

$$\mathbf{C}_p^{(l)} = \text{softmax}({\mathbf{H}^{(l-1)}}^\top {\mathbf{W}_{p\theta}^{(l)}}^\top \mathbf{W}_{p\phi}^{(l)} \mathbf{H}^{(l-1)}), \tag{7}$$

where softmax normalizes the matrix values. The additive attention mechanism in Eq. (5), thus, lets the adaptive graph convolution in Eq. (7) model the skeleton structure as a fully connected graph.

C. Skeleton-based Spatial Transformer Networks

S-TR [32] is an attention-based method which models dependencies between body joints at each time step using the self-attention operation found in Transformers [33]. In this method, a Spatial Self-Attention (SSA) module is designed to adaptively learn data-dependent pairwise body joint correlations using multi-head self-attention.

The SSA module at each layer l applies trainable query, key, and value transformations $\mathbf{W}_q^{(l)} \in \mathbb{R}^{C^{(l-1)} \times dq}$, $\mathbf{W}_k^{(l)} \in \mathbb{R}^{C^{(l-1)} \times dk}$, $\mathbf{W}_v^{(l)} \in \mathbb{R}^{C^{(l-1)} \times dv}$ on the feature vector $\mathbf{h}_i^t \in \mathbb{R}^{C^{(l-1)}}$ of node i at time step t to obtain the query, key, and value vectors $\mathbf{q}_i^t \in \mathbb{R}^{dq}$, $\mathbf{k}_i^t \in \mathbb{R}^{dk}$, $\mathbf{v}_i^t \in \mathbb{R}^{dv}$. The correlation weight for each pair of i, j nodes at time t is obtained using a query-key dot product

$$\alpha_{ij}^t = {\mathbf{q}_i^t}^\top \mathbf{k}_j^t. \tag{8}$$

The updated feature vector of node i at time t has size $C^{(l)}$ and is obtained using a weighted feature aggregation of value vectors:

$$\bar{\mathbf{h}}_i^t = \sum_j \text{softmax}_j\left(\frac{\alpha_{ij}^t}{\sqrt{dk}}\right)\mathbf{v}_j^t. \tag{9}$$

For each attention head, the feature transformation is performed with a different set of learnable parameters while the transformation matrices are shared across all the nodes. The output features of the SSA module are finally computed by

applying a learnable linear transformation on the concatenated features from S attention heads:

$$\bar{\mathbf{h}}_i^t = (\Big\|_{s=1}^{S} \bar{\mathbf{h}}_{is}^t)\mathbf{W}_o. \tag{10}$$

SSA has similarities to a graph convolution operation on a fully connected graph for which the node connection weights are learned dynamically. The first three layers of the S-TR model extract features with GC and TC blocks as defined in Eq. 4 while in the remaining layers of the model SSA substitutes GC.

III. Continual Inference Networks

First introduced in [34] and subsequently formalized in [35], Continual Inference Networks are Deep Neural Networks that can operate efficiently on both fixed-size (spatio-)temporal batches of data, where the whole temporal sequence is known up front, as well as on continual data, where new input steps are collected continually and inference needs to be performed efficiently in an online manner for each received frame.

Definition (Continual Inference Network). *A Continual Inference Network is a Deep Neural Network, which*

- *is capable of continual step inference without computational redundancy,*
- *is capable of batch inference corresponding to a non-continual Neural Network,*
- *produces identical outputs for batch inference and step inference given identical receptive fields,*
- *uses one set of trainable parameters for both batch and step inference.*

Recurrent Neural Networks (RNNs) are a common family of Deep Neural Networks, which possess the above-described properties. 3D Convolutional Neural Networks (3D CNNs), Transformers, and Spatio-Temporal Graph Convolutional Networks are not Continual Inference Networks since they cannot make predictions time-step by time-step without considerable computational redundancy; they need to cache a sliding window of prior input frames and assemble them into a fixed-size sequence that is subsequently passed through the network to make a new predictions during online inference.

Recently, *Continual* 3D CNNs were made possible through the proposal of *Continual* 3D Convolutions [34]. Likewise, shallow *Continual* Transformers based on *Continual* Dot-product Attentions were introduced in [35]. We continue this line of work by extending Spatio-Temporal Graph Convolutional Networks (ST-GCNs) with a *Continual* formulation as well. To do so, let us first present and expand on the theory on Continual Convolutions.

A. Continual Convolution

The Continual Convolution operation produces the exact same output as the regular convolution does, but performs the computation in a streaming fashion while caching intermediary results. Consider a single channel 2D convolution over an input $\mathbf{X} \in \mathbb{R}^{T \times V}$ with temporal dimension T and a dimension of V vertices. Given a convolutional kernel with weights

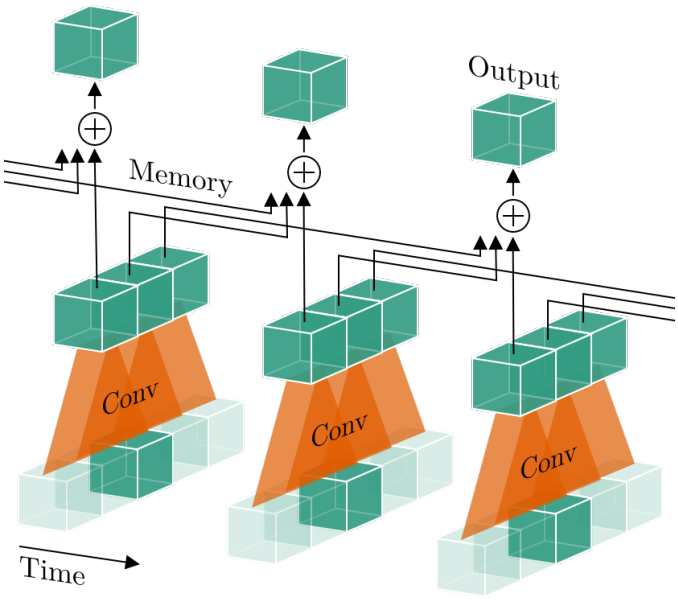

Fig. 3: **Continual Convolutions** are performed in two stages: First, the input is zero-padded and convolved with the convolutional kernel ($K = 3$ in illustration) to produce intermediary results. Subsequently, these are cached and summed up to produce the final output.

$\mathbf{W} \in \mathbb{R}^{K \times V}$, where K is the temporal kernel size, and a bias w_0, a regular convolution would compute the output $\mathbf{y}^{(t)}$ for time-step $t \in K..T$ as

$$\mathbf{y}^{(t)} = w_0 + \sum_{k=1}^{K} \sum_{v=1}^{V} \mathbf{W}_{k,v} \cdot \mathbf{X}_v^{(t-k-1)}. \tag{11}$$

Considering this computation in the context of online processing, where $T \to \infty$ and one input slice $\mathbf{X}^{(t)}$ is revealed in each time step, we find that $K-1$ previous slices, i.e. $(K-1) \cdot V$ values, need to be stored between time-steps.

An alternative computational sequence is used in Continual Convolutions. Here, the input slice $\mathbf{X}^{(t)}$ is convolved with the kernel $\mathbf{W}$ in the same time-step it is received. This is specified in Eq. (12a). The intermediate results are then cached in memory $\mathbf{m}$ ($K-1$ values stored between time-steps) and aggregated according to Eq. (12b).

$$\mathbf{m}^{(t)} = \left[\sum_{v=1}^{V} \mathbf{W}_{k,v} \cdot \mathbf{X}_v^{(t)} : \; k \in 1..K \right] \tag{12a}$$

$$\mathbf{y}^{(t)} = w_0 + \sum_{k=1}^{K} \mathbf{m}_k^{(t-k-1)} \tag{12b}$$

A graphical representation of this is shown in Fig. 3.

B. Delayed Residual

The temporal convolutions of regular Spatio-Temporal Graph Convolution blocks usually employ zero-padding to ensure equal temporal shape for input and output feature maps. This zero-padding is discarded for Continual Convolutions to

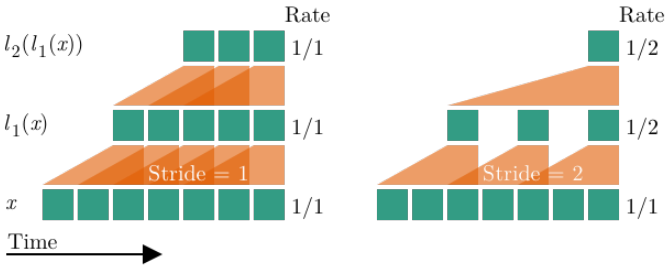

Fig. 4: **Temporal stride** in a Continual Convolution layer l_1 with temporal stride larger than one (right) reduces the prediction rate compared to a layer with stride one (left). The rate reduction is inherited by subsequent layers.

avoid continual redundancies [34]. To retain weight compatibility between the regular and continual networks, a delay to the residual connection is necessary. This delay amounts to

$$k_T + (k_T - 1)(d_T - 1) - p_T - 1 \tag{13}$$

steps, where k_T, d_T, and p_T are respectively the temporal kernel size, dilation, and zero-padding of the corresponding regular convolution.

C. Temporal Stride

In Section III-A, it is assumed that one output is produced for each input received. However, many spatio-temporal networks including ST-GCN [15], AGCN [16], and S-TR [32], use temporal stride > 1 in their temporal convolutions. For offline computation, this has the beneficial effect of reducing the computational and memory complexity, but in the online computational setting, it also reduces the prediction rate. This is illustrated in Fig. 4. For a neural network with L layers, each with a temporal stride s, the effective network stride is given by

$$s_{NN} = \prod_{l=1}^{L} s_l \tag{14}$$

and the corresponding network prediction rate is

$$r_{NN} = 1/s_{NN}. \tag{15}$$

Since a ST-GCN network has two layers with stride two, the corresponding Continual ST-GCN (*Co*ST-GCN) has a prediction rate one fourth the input rate.

IV. Continual Spatio-Temporal Graph Convolutional Networks

Many well-performing methods for skeleton-based action recognition, including the ST-GCN [15], AGCN [16], and S-TR [32], share a common block structure, which can be described by Eq. (4). Here, the main difference between methods lies in how the graph information is processed, i.e. in their definition of GC$(\cdot)$.

The regular skeleton-based methods successively extract complete spatio-temporal skeleton features from the whole sequence with each block before classifying an action. Considering one block in isolation, the spatio-temporal feature extraction is given by a spatial (graph) convolution followed by a regular temporal convolution. Here, graph convolutions operate locally within a time-step[1], whereas the temporal convolution does not. Since the next block l takes as input $\mathbf{H}^{(l-1)}$, the output of the prior block and thereby its temporal convolution, the output of the next spatial (graph) convolution becomes a function of multiple prior time-steps. With regular temporal convolutions, features produced by multiple blocks cannot be trivially disentangled and cached in time. Accordingly online operation with per-skeleton predictions can be attained by caching $T-1$ prior skeletons, concatenating these with the newest skeleton, and performing regular spatio-temporal inference. However, this comes with significant computational redundancy, where the complexity of online frame-wise inference is the same as for clip-based inference.

To alleviate this issue, we propose to employ Continual Convolutions in the temporal modeling of Spatio-temporal Graph Convolutional Networks. By restricting the GC$(\cdot)$ function to only operate locally within a time-step, we can define a *Continual* Spatio-Temporal block by replacing the original temporal 2D convolution with a continual one. To retain weight-compatibility with regular (non-continual) networks we moreover need to delay the residual to keep temporal alignment. Given $\mathbf{H}_{l-1}^{(t)}$, i.e. the features of layer $l-1$ in a time-step t, the feature in layer l at time t is given by

$$\mathbf{H}_{l}^{(t)} = \sigma\left(\text{Delay}(\text{Res}(\mathbf{H}_{l-1}^{(t)})) + \text{BN}(Co\text{TC}(\text{GC}(\mathbf{H}_{l-1}^{(t)})))\right). \tag{16}$$

Here, Delay(Res($\mathbf{H}_{l-1}^{(t)}$)) outputs the delayed residual in a first-in-first-out manner corresponding to the delay of the *Continual* Temporal Convolutional as computed by Eq. (13). A graphical illustration of such a block is seen in Fig. 5. It should be noted that the restriction of temporal locality does influence the computations of some skeleton-based action recognition methods. For example, the AGCN originally computes one vertex attention weighting based on the whole spatio-temporal feature-map, whereas a *Continual* AGCN (*Co*AGCN) computes separate vertex attentions for each time-step.

The resulting *Continual* Spatio-temporal Graph Convolutional Network is defined by stacking multiple such blocks[2] followed by *Continual* Global Average Pooling [34] and a fully connected layer. The Continual Inference Networks retain the same computational complexity as regular networks during clip-based inference, but can perform online frame-by-frame predictions much more efficiently, as detailed in Section IV-A. We should note that all methods, which share the the same structure as ST-GCN, i.e. a decoupled temporal and spatial convolution to perform feature transformation and aggregation over the time domain can be transformed to continual version using the approach outlined above.

A. Computational Complexity

Denote the time complexity of passing a single skeleton frame through the convolutional blocks with stride 1 by $\mathcal{O}(B)$

[1]AGCN is an exception to this, since the additive attention considers a node's features over all time-steps.

[2]Following the original ST-GCN, AGCN, and S-TR architectures, ten blocks were used for the networks in this paper.

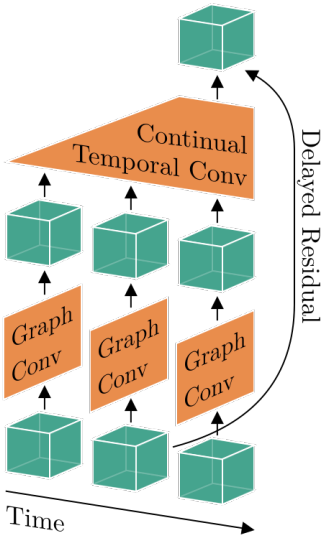

Fig. 5: **Continual Spatio-temporal Graph Convolution Blocks** consist of an in-time Graph Convolution followed by an across-time Continual Convolution (here a kernel size of three is depicted). The residual connection is delayed to ensure temporal alignment with the continual temporal convolution that is weight-compatible with non-continual networks.

Conversion Strategy	**Acc. (%)**	**FLOPs (G)**
$\text{Reg}_{s=4}^{p=\text{eq}}$ (baseline)	93.4	16.73
$\text{Reg}_{s=4}^{p=\text{eq}} \xrightarrow{FT} \text{Reg}_{s=1}^{p=0}$	93.8 (+0.4)	36.90 ($\uparrow 2.2\times$)
$\text{Reg}_{s=4}^{p=\text{eq}} \rightarrow \text{Co}_{s=4}^{p=0}$	93.1 (−0.3)	0.27 ($\downarrow 63.2\times$)
$\text{Reg}_{s=4}^{p=\text{eq}} \rightarrow \text{Co}_{s=1}^{p=0}$	24.0 (−69.4)	0.16 ($\downarrow 107.7\times$)
$\text{Reg}_{s=4}^{p=\text{eq}} \rightarrow \text{Co}_{s=1}^{p=0} \xrightarrow{FT} \text{Co}*$	93.2 (−0.2)	0.16 ($\downarrow 107.7\times$)
$\text{Reg}_{s=4}^{p=\text{eq}} \xrightarrow{FT} \text{Reg}_{s=1}^{p=0} \rightarrow \text{Co}*$	93.8 (+0.4)	0.16 ($\downarrow 107.7\times$)

TABLE I: **Conversion Strategies** from regular (Reg) to Continual (Co) ST-GCN. Noted is the top-1 X-View validation accuracy on NTU RGB+D 60 and the FLOPs per prediction. The superscript p and subscript s indicate network padding and stride respectively. The arrows $\rightarrow$ and $\xrightarrow{FT}$ denote direct conversion and conversion with subsequent fine-tuning. Parentheses show the change relative to the baseline with colours indicating improvement / deterioration.

and time complexity of utilizing the prediction head by $\mathcal{O}(H)$. Given an effective clip-size T, the complexity of producing a prediction with a regular CNN is approximately $\mathcal{O}(\text{CNN}) \approx T \cdot \mathcal{O}(B) + \mathcal{O}(H)$. For a Continual CNN, the corresponding complexity is $\mathcal{O}(Co\text{CNN}) \approx \mathcal{O}(B) + \mathcal{O}(H)$. Computational savings thus scale linearly with the effective clip-size T and are more prominent the larger $\mathcal{O}(B)$ is compared to $\mathcal{O}(H)$.

V. Experiments

A. Datasets

a) NTU RGB+D 60 [29]*:* A large indoor-captured dataset which is widely used for evaluating skeleton-based action recognition methods. This dataset contains 56,880 action clips and their corresponding 3D skeleton sequences captured by three Microsoft Kinect-v2 cameras from three different views. The clips are performed by 40 different subjects and constitute 60 action classes. The NTU RGB+D 60 dataset comes with two benchmarks, Cross-View (X-View) and Cross-Subject (X-Sub). The X-View benchmark provides 37,920 skeleton sequences coming from the camera views #2 and #3 as training data, and 18,960 skeleton sequences coming from the first camera view as test set. The X-Sub benchmark provides 40,320 skeleton sequences from 20 subjects as training data and 16,560 skeleton sequences from the other 20 subjects as test data. In this dataset, each skeleton has 25 body joints with three different channels each, and each action clip comes with a sequence of 300 skeletons.

b) NTU RGB+D 120 [30]*:* An extension of the NTU RGB+D 60 dataset containing an additional 57,600 skeleton sequences from extra 60 classes. NTU RGB+D 120 is currently the largest dataset providing 3D body joint coordinates for skeletons and in total, it contains 114,480 skeleton sequences from 120 action classes. The action clips in this dataset are performed by 106 subjects and 32 different camera setups are used for capturing the videos. This dataset comes with two benchmarks: Cross-Subject (X-Sub) and Cross-Setup (X-Set). The X-Sub benchmark provides the skeleton sequences of 53 subjects as training data and the remaining skeleton sequences from the other 53 subjects as test data. In the X-Set benchmark, the skeleton sequences with even camera setup IDs are provided as training data and test data contains the remaining skeleton sequences with odd camera setup IDs.

c) Kinetics Skeleton 400 [31]*:* A widely used dataset for action recognition containing 300,000 video action clips of 400 different classes which are collected from YouTube. Skeletons were extracted from each frame of these video clips using the OpenPose toolbox [2]. Each skeleton is represented by 18 body joints and each body joint contains spatial 2D coordinates and the estimation confidence score as its three features. We use the dataset version provided by [15], which contains 240,000 skeleton sequences as training data and 20,000 skeleton sequences as test data, in our experiments.

B. Experimental Settings

All models were implemented within the PyTorch framework [36] using the Ride library [37]. Models were trained using a SGD optimizer with learning rate 0.1 at batch size 64, momentum of 0.9, and a one-cycle learning rate policy [38] using a cosine annealing strategy. For models which could not fit a batch size of 64 on a Nvidia RTX 2080 Ti, the learning rate was adjusted following the linear scaling rule [39]. Our source code is available at www.github.com/lukashedegaard/continual-skeletons.

C. Conversion and Fine-tuning Strategies

Though regular and Continual CNNs are weight-compatible, the direct transfer of weights is imperfect if the regular CNN was trained with zero-padding [34]. As in most CNNs, it is common practice to utilize padding in skeleton-based spatio-temporal networks to retain the temporal feature size in

Model	Frames per pred	Accuracy (%)		Params (M)	Max mem. (MB)	FLOPs per pred (G)	Throughput (preds/s)	
		X-Sub	X-View				CPU	GPU
ST-GCN	300	86.0	93.4	3.14	45.3	16.73	2.3	180.8
ST-GCN*	300	86.3 (+0.3)	93.8 (+0.4)	3.14	72.6 (160%)	36.90 (↑ 2.2×)	1.1 (↓ 2.1×)	90.4 (↓ 2.0×)
*Co*ST-GCN	4	85.3 (−0.7)	93.1 (−0.3)	3.14	36.0 (79%)	0.27 (↓ 63.2×)	32.3 (↑ 14.0×)	2375.2 (↑ 13.1×)
*Co*ST-GCN*	1	86.3 (+0.3)	93.8 (+0.4)	3.14	36.1 (80%)	0.16 (↓ 107.7×)	46.1 (↑ 20.0×)	4202.2 (↑ 23.2×)
AGCN	300	86.4	94.3	3.47	48.4	18.69	2.1	146.2
AGCN*	300	84.1 (−2.3)	92.6 (−1.7)	3.47	76.4 (158%)	40.87 (↑ 2.2×)	1.0 (↓ 2.1×)	71.2 (↓ 2.0×)
*Co*AGCN	4	86.0 (−0.4)	93.9 (−0.4)	3.47	37.3 (77%)	0.30 (↓ 63.2×)	25.0 (↑ 11.9×)	2248.4 (↑ 15.4×)
*Co*AGCN*	1	84.1 (−2.3)	92.6 (−1.7)	3.47	37.4 (77%)	0.17 (↓ 108.8×)	30.4 (↑ 14.5×)	3817.2 (↑ 26.1×)
S-TR	300	86.8	93.8	3.09	74.2	16.14	1.7	156.3
S-TR*	300	86.3 (−0.5)	92.4 (−1.4)	3.09	111.5 (150%)	35.65 (↑ 2.2×)	0.8 (↓ 2.1×)	85.1 (↓ 1.8×)
*Co*S-TR	4	86.5 (−0.3)	93.3 (−0.5)	3.09	35.9 (48%)	0.22 (↓ 63.2×)	30.3 (↑ 17.8×)	2189.5 (↑ 14.0×)
*Co*S-TR*	1	86.3 (−0.3)	92.4 (−1.4)	3.09	36.1 (49%)	0.15 (↓ 107.6×)	43.8 (↑ 25.8×)	3775.3 (↑ 24.2×)

TABLE II: **NTU RGB+D 60 transfer accuracy and performance benchmarks**. Noted is the top-1 validation accuracy using joints as the only modality. Max mem. is the maximum allocated memory on GPU during inference noted in megabytes. Max. mem, FLOPs, and throughput on CPU account for one new prediction with batch size one while throughput on GPU uses the largest fitting power of two as batch size. Parentheses indicate the improvement / deterioration relative to the original model.

consecutive layers (though temporal shrinkage is not a concern given the long input clips).

Another common design choice, which has a significant impact in on the performance of Continual Inference Networks, is the utilization of temporal stride larger than one. For regular networks, this has the benefit of reducing the computational complexity per clip prediction. In Continual Inference Networks, however, it reduces the prediction rate, and actually increases the complexity per prediction (see Section III-C). In the continual case, it would thus be computationally beneficial to reduce the stride of all layers to one. However, this results in a stride-inflicted *model-shift*.

Thus far, the *model-shift* inflicted by padding removal and stride reduction, as well as how to best perform the conversion from a regular CNN to a Continual CNN in such cases has not been studied. In this set of experiments, we explore strategies on how to best convert and fine-tune regular networks to achieve good frame-by-frame performance. We use a standard ST-GCN [15] trained on joints only as our starting-point, and explore the accuracy achieved by:

1) Converting to from regular network with equal padding and stride four ($\text{Reg}_{s=4}^{p=\text{eq}}$) to a Continual Inference Network, where zero-padding is omitted ($\text{Co}_{s=4}^{p=0}$).
2) Reducing the network stride to one without fine-tuning ($\text{Co}_{s=1}^{p=0}$).
3) Fine-tuning the $\text{Co}_{s=1}^{p=0}$ network (= Co∗).
4) Fine-tuning a conversion-optimal regular network which has no zero-padding and a stride of one ($\text{Reg}_{s=1}^{p=0}$).
5) Converting from $\text{Reg}_{s=1}^{p=0}$ to Continual (= Co∗).

As seen in Table I, the direct transfer of weights was found to have a modest negative impact on the accuracy (by -0.3%) due the removal of zero-padding. This is considerably less than was found in [34]. Our conjecture is that the smaller amount of zeros relative to clip size used in skeleton-based recognition (8 zeros per 300 frames or 2.67%) compared to video-based recognition (e.g., 2 zeros per 16 frames or or 12.5%) makes the removal of zero-padding less detrimental since zeros contribute relatively less to the downstream features. Lowering the stride to one and removing zero-padding reduced accuracy by a substantial amount but allowed the Continual Inference Network to operate at much lower FLOPs. This accuracy drop is alleviated equally effectively by either (a) initializing the $\text{Co}_{s=1}^{p=0}$ with standard weights and fine-tuning in the continual regime or (b) first fine-tuning the conversion-optimal regular network ($\text{Reg}_{s=1}^{p=0}$) and subsequently converting to a Continual Inference Network, though the latter had lower training times in practice. We fine-tuned the networks using the settings described in Section V-B. As visualised in Fig. 6, we found 20 epochs of fine-tuning using the settings described in Section V-B recover accuracy on NTU RGB+D 60 with additional training yielding only marginal differences. Following this approach the (padding zero, stride one) optimized Continual ST-GCN (*Co*ST-GCN*) achieves a similar prediction accuracy while reducing the computational complexity by a factor $107.7\times$ relative to original ST-GCN!

D. Conversion of Attention Architectures

As we explored in Section V-C, the ST-GCN network architecture can easily be modified and fine-tuned to achieve high accuracy for frame-by-frame predictions with exceptionally low computational complexity. A natural follow-up question is whether this conversion is equally successful for more complicated spatio-temporal architectures that employ attention mechanisms. To investigate this, we conduct a similar transfer for two recent ST-GCN variants, the Adaptive GCN (AGCN) [16] and the Spatial Transformer Network (S-TR) [32]. While S-TR is easily converted to a Continual Inference Network (*Co*S-TR) by replacing convolutions, residuals and pooling operators with Continual ones, the AGCN requires additional care. In the original version of AGCN, the vertex attention matrix $\mathbf{C}_p$ (see Eq. (7)) is computed from the global representations in the layer over all time-steps. Since this operation would be acausal in the context of a Continual Inference Network, we restrict it to utilize only the frame-specific subset of features. As a fine-tuning strategy, we first make the conversion from regular network to a conversion-

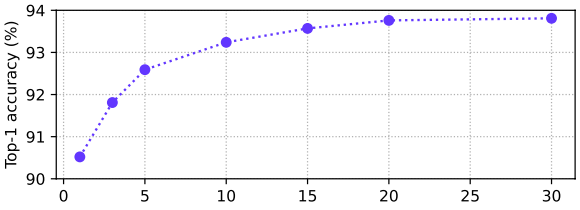

Fig. 6: **Fine-tuning epochs** and associated top-1 accuracy on NTU RGB+D 60 X-View for a transfer from a pre-trained ST-GCN with zero-padding and accumulated stride of four to an equivalent (*Co*)ST-GCN* with no zero-padding and stride one.

optimal network, and subsequently convert and evaluate the continual version.

Our results are presented in Table II. Here we see that all three architectures can be successfully converted to continual versions. The fine-tuned conversion-optimal models (marked by *) generally exhibit a higher computational complexity than their source models due to their stride decrease. While the ST-GCN* attained increased performance by lowering stride, AGCN* and S-TR* suffer slight accuracy deterioration. This may be due to smaller receptive fields of their attention mechanisms, which likely benefit from observing a larger context. Unlike the transfer from the original models with padding and stride four to continual models, the continual models with weights from ST-GCN*, AGCN*, and S-TR*, i.e. *Co*ST-GCN*, *Co*AGCN*, and *Co*S-TR* attain the exact same accuracy as their source models on both the X-Sub and X-View benchmarks, with two orders of magnitude less FLOPs per prediction during online inference.

E. Speed and Memory

Diving deeper into the differences between regular and continual networks, we conduct throughput benchmarks on a MacBook Pro 16" with a 2.6 GHz 6-Core Intel Core i7 CPU and a NVIDIA RTX 2080 Ti GPU. Here, we measure the prediction-time as the time it takes to transfer an input of batch size one from CPU to GPU (if applicable), perform inference, and transfer the results back to CPU again. On CPU, a batch size of one is used, while for GPU, the largest fitting power of two is employed (i.e. {128, 64, 256, 256} for the {Reg, Reg*, *Co*, and *Co**} models). We measure the maximum allocated memory during inference on GPU for batch size one.

As seen in Table II, the change in speed relative to the original models follow a similar trend to those seen for FLOPs. The non-continual stride one variants (denoted by *) exhibit roughly half the speed of the original models, while the continual models enjoy more than a magnitude speed up on both CPU and GPU. As expected, the continual stride one models (*Co**) attain the largest inference throughput. These relative speed-ups are lower than the relative FLOPs reductions due to the read/writes of internal intermediary features in the Continual Convolutions since these are not accounted for by the FLOPs metric while still adding to the runtime. This gap could be reduced on hardware with in- or near-memory computing.

	Model	S.	Accuracy (%) X-Sub	Accuracy (%) X-View	FLOPs (G)
Clip	SGN [27]	1	89.4	94.5	-
	MS-G3D [21]	1	89.4	95.0	-
		2	**91.5**	96.2	-
	ST-TR [32]	1	89.2	95.8	-
		2	90.3	***96.3***	-
	MS-AAGCN [17]	4	90.0	96.2	-
	Hyper-GNN [19]	3	89.5	95.7	-
	FGCN [20]	4	90.2	***96.3***	-
	DGNN [18]	4	89.9	96.1	126.80
	AS-GCN [40]	1	86.8	94.2	27.00
	AGC-LSTM [41]	2	89.2	95.0	54.40
	ShiftGCN [24]	1	87.8	95.1	2.50
		2	89.7	96.0	5.00
		4	***90.7***	**96.5**	10.00
	ShiftGCN++ [26]	1	87.9	94.8	**0.40**
		2	89.7	95.7	***0.80***
		4	90.5	***96.3***	1.70
	ST-GCN†	1	86.0	93.4	16.73
		2	88.1	94.9	33.46
	AGCN†	1	86.4	94.3	18.69
		2	88.3	95.3	37.38
	S-TR†	1	86.8	93.8	16.20
		2	89.1	95.3	32.40
Frame	Deep-LSTM [29]	1	60.7	67.3	-
	VA-LSTM [13]	1	79.2	87.7	-
	*Co*ST-GCN (ours)	1	86.0	93.4	0.27
		2	88.1	94.8	0.54
	*Co*ST-GCN* (ours)	1	86.3	93.8	***0.16***
		2	88.3	***95.0***	0.32
	*Co*AGCN (ours)	1	86.4	94.2	0.30
		2	88.2	**95.3**	0.60
	*Co*AGCN* (ours)	1	84.1	92.6	0.22
		2	86.0	93.1	0.44
	*Co*S-TR (ours)	1	86.5	93.5	0.17
		2	**88.8**	**95.3**	0.34
	*Co*S-TR* (ours)	1	86.3	92.4	**0.15**
		2	***88.9***	94.8	0.30

TABLE III: **NTU RGB+D 60** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** and pareto-optimal results.

Considering the maximum allocated memory at inference, we find that the continual models reduce memory by 20-52%. While the Continual Convolution and -Pooling layers do add some internal state that adds to the memory consumption, the intermediary features that are passed between network layers are much smaller, i.e. one frame instead of 75 to 300 frames.

F. Comparison with Prior Works

Most current state-of-the-art methods for skeleton-based action recognition are not able to efficiently perform frame-by-frame predictions in the online setting, since they are constrained to operate on whole skeleton-sequences. Some RNN-based methods, e.g. Deep-LSTM [29] and VA-LSTM [13], can be used for redundancy-free frame-wise predictions, but

	Model	S.	Accuracy (%) X-Sub	Accuracy (%) X-Set	FLOPs (G)
Clip	Part-Aware LSTM [42]	1	25.5	26.3	-
	ST-LSTM [10]	1	55.7	57.9	-
	TSRJI [43]	1	67.9	62.8	-
	SGN [27]	1	79.2	81.5	-
	MS-G3D [21]	2	**86.9**	**88.4**	-
	FGCN [20]	4	85.4	87.4	-
	ShiftGCN [24]	1	80.9	83.2	2.50
		2	85.3	86.6	5.00
		4	***85.9***	***87.6***	10.00
	ShiftGCN++ [26]	1	80.5	83.0	**0.40**
		2	84.9	86.2	***0.80***
		4	85.6	87.2	1.70
	ST-GCN†	1	79.0	80.7	16.73
		2	83.7	85.1	33.46
	AGCN†	1	79.7	80.7	18.69
		2	84.0	85.4	37.38
	S-TR†	1	80.2	81.8	16.20
		2	84.8	86.2	32.40
Frame	*Co*ST-GCN (ours)	1	78.9	80.7	0.27
		2	83.7	85.1	0.54
	*Co*ST-GCN* (ours)	1	79.4	81.6	***0.16***
		2	84.0	85.5	0.32
	*Co*AGCN (ours)	1	79.6	80.7	0.30
		2	84.0	85.3	0.60
	*Co*AGCN* (ours)	1	77.3	79.1	0.22
		2	80.4	82.0	0.44
	*Co*S-TR (ours)	1	80.1	81.7	0.17
		2	***84.8***	**86.1**	0.34
	*Co*S-TR* (ours)	1	79.7	81.7	**0.15**
		2	**84.8**	***86.1***	0.30

TABLE IV: **NTU RGB+D 120** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** and pareto-optimal results.

	Model	S.	Accuracy (%) Top-1	Accuracy (%) Top-5	FLOPs (G)
Clip	Feature Enc. [15, 44]	1	14.9	25.8	-
	Deep LSTM [11, 15]	1	16.4	35.3	-
	TCN [4, 15]	1	20.3	40.0	-
	AS-GCN [40]	1	34.8	56.5	-
	ST-GR [45]	1	33.6	56.1	-
	DGNN [18]	4	***36.9***	59.6	-
	MS-G3D [21]	2	**38.0**	***60.9***	-
	MS-AAGCN [17]	4	37.8	**61.0**	-
	Hyper-GNN [19]	3	37.1	60.0	-
	ST-GCN†	1	33.4	56.1	***12.04***
		2	34.4	57.5	24.09
	AGCN†	1	35.0	57.5	13.45
		2	***36.9***	59.6	26.91
	S-TR†	1	32.0	54.9	**11.62**
		2	34.7	57.9	23.24
Frame	*Co*ST-GCN (ours)	1	31.8	54.6	0.16
		2	33.1	56.1	0.32
	*Co*ST-GCN* (ours)	1	30.2	52.4	**0.11**
		2	32.2	54.5	0.22
	*Co*AGCN (ours)	1	33.0	55.5	0.18
		2	**35.0**	**57.3**	0.36
	*Co*AGCN* (ours)	1	23.3	44.3	***0.12***
		2	27.5	49.1	0.25
	*Co*S-TR (ours)	1	29.7	52.6	0.16
		2	***32.7***	***55.6***	0.31
	*Co*S-TR* (ours)	1	27.4	49.7	**0.11**
		2	29.9	52.7	0.22

TABLE V: **Kinetics Skeleton 400** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 and top-5 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** and pareto-optimal results.

their reported accuracy has been sub-par relative to newer methods that sprung from ST-GCN. The recently proposed AGC-LSTM [41] does report results on-par with CNN-based methods, and might also be able to provide redundancy-free frame-wise results, but we cannot validate this due to the lack of publicly available source code and details in the published paper. While ShiftGCN and ShiftGCN++ offer impressively low FLOPs, it should be noted that the shift operation, which is a significant part of their operational load, is not accounted for by the FLOPs metric. Due to the non-causal nature of the temporal shift operation in ShiftGCN and ShiftGCN++, they cannot be transformed into Continual Inference Networks in their current form, though a *Continual* Shift operation could plausibly be devised.

Many works have shown that the inclusion of multiple modalities leads to increased accuracy [15, 16, 18, 21, 24]. In our context, these modalities amount to *joints*, which are the original coordinates of the body joints, and *bones*, which are the differences between connected joints. Additional *joint motion* and *bone motion* modalities can be retrieved by computing the differences between adjacent frames in time for the joint and bone streams respectively. Models are trained individually on each stream and combined by adding their softmax outputs prior to prediction.

We evaluate and compare our proposed continual models, *Co*ST-GCN, *Co*AGCN, *Co*S-TR, with prior works on the NTU RGB+D 60, NTU RGB+D 120, and Kinetics Skeleton 400 datasets as presented in Table III, Table IV, and Table V.

The *Co*ST-GCN and *Co*S-TR models transfer well across all datasets both with (*) and without padding and stride modifications. For *Co*AGCN, we find that the change to stride one deteriorates accuracy. We surmise that the attention matrix in Eq. (7) may need a larger receptive field (basing the attention on more nodes as in AGCN) to provide beneficial adaptations; a per-step change in attention might provide more noise than clarity in middle and late network layers. As found in prior works, the multi-stream approach with ensemble predictions gives a meaningful boost in accuracy across all experiment.

The Continual Skeleton models provide competitive accuracy at multiple orders of magnitude reduction of FLOPs per prediction in the online setting compared to the original non-continual models. While none of our results beat prior state-of-the-art accuracy in absolute terms, this was never the intent with the method. Rather, we have successfully shown that online inference can be greatly accelerated for models in the ST-GCN family with state-of-the-art accuracy/complexity trade-offs to follow. For instance, our one and two-stream *Co*S-

TR* achieve pareto optimal results on all subsets of the NTU RGB+D 60 and NTU RGB+D 120 datasets meaning that no other model improves on either accuracy and FLOPs without reducing the other. Pareto-optimal models have been highlighted in Tables III, IV, and V accordingly. Our approach may be used similarly to accelerate other architectures for skeleton-based human action recognition with temporal convolutions.

VI. Conclusion

In this paper, we proposed *Continual* Spatio-Temporal Graph Convolutional Networks, an architectural enhancement for skeleton-based human action recognition methods, which augments prior methods with the ability to perform predictions frame-by-frame during online inference while attaining weight compatibility for batch inference. We re-implement and benchmark three prominent methods, the ST-GCN, AGCN, and S-TR, as novel Continual Inference Networks, *Co*ST-GCN, *Co*AGCN, and *Co*S-TR, and propose architectural modifications to maximize their frame-by-frame inference speed. Through experiments on three widely used human skeleton datasets, NTU RGB+D 60, NTU RGB+D 120, and Kinetics Skeleton 400, we show up to 26× on-hardware speedups, 109× reduction in FLOPs per prediction, and 52% reduction in maximum memory allocated memory during online inference with similar accuracy to those of the original networks. Our proposed architectural modifications are generic in nature and can be used for many methods in skeleton-based action recognition. It is our hope, that this innovation will make skeleton-based action recognition a viable option for online recognition systems on recourse-constrained devices and systems with real-time requirements.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] F. Han, B. Reily, W. Hoff, and H. Zhang, "Space-time representation of people based on 3D skeletal data: A review," *Computer Vision and Image Understanding*, vol. 158, pp. 85–105, 2017.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[3] H. Liu, J. Tu, and M. Liu, "Two-stream 3D convolutional neural network for skeleton-based action recognition," *arXiv preprint arXiv:1705.08106*, 2017.

[4] T. S. Kim and A. Reiter, "Interpretable 3D human action analysis with temporal convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1623–1631.

[5] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3D action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3288–3297.

[6] M. Liu, H. Liu, and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition," *Pattern Recognition*, vol. 68, pp. 346–362, 2017.

[7] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN," in *IEEE International Conference on Multimedia & Expo Workshops*, 2017, pp. 601–604.

[8] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with convolutional neural networks," in *IEEE International Conference on Multimedia & Expo Workshops*, 2017, pp. 597–600.

[9] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1110–1118.

[10] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 816–833.

[11] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1010–1019.

[12] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI Conference on Artificial Intelligence*, 2017, pp. 4263–4270.

[13] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *IEEE International Conference on Computer Vision*, 2017, pp. 2117–2126.

[14] L. Li, W. Zheng, Z. Zhang, Y. Huang, and L. Wang, "Skeleton-based relational modeling for action recognition," *arXiv preprint arXiv:1805.02556*, vol. 1, no. 2, p. 3, 2018.

[15] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 7444–7452.

[16] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 026–12 035.

[17] ——, "Skeleton-based action recognition with multi-stream adaptive graph convolutional networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 9532–9545, 2020.

[18] ——, "Skeleton-based action recognition with directed graph neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7912–7921.

[19] X. Hao, J. Li, Y. Guo, T. Jiang, and M. Yu, "Hypergraph neural network for skeleton-based action recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 2263–2275, 2021.

[20] H. Yang, D. Yan, L. Zhang, Y. Sun, D. Li, and S. J. Maybank, "Feedback graph convolutional network for skeleton-based action recognition," *IEEE Transactions on Image Processing*, vol. 31, pp. 164–175, 2021.

[21] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, "Disentangling and unifying graph convolutions for skeleton-based action recognition," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 140–149, 2020.

[22] W. Peng, X. Hong, H. Chen, and G. Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching." in *AAAI Conference on Artificial Intelligence*, 2020, pp. 2669–2676.

[23] N. Heidari and A. Iosifidis, "Progressive spatio-temporal graph convolutional network for skeleton-based human action recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3220–3224.

[24] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, and H. Lu,

"Skeleton-based action recognition with shift graph convolutional network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017.
[26] K. Cheng, Y. Zhang, X. He, J. Cheng, and H. Lu, "Extremely lightweight skeleton-based action recognition with shiftgcn++," *IEEE Transactions on Image Processing*, vol. 30, pp. 7333–7348, 2021.
[27] P. Zhang, C. Lan, W. Zeng, J. Xing, J. Xue, and N. Zheng, "Semantics-guided neural networks for efficient skeleton-based human action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
[28] N. Heidari and A. Iosifidis, "Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition," in *International Conference on Pattern Recognition*, 2020.
[29] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.
[30] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.
[31] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *preprint, arXiv:1705.06950*, 2017.
[32] C. Plizzari, M. Cannici, and M. Matteucci, "Skeleton-based action recognition via spatial and temporal transformer networks," *Computer Vision and Image Understanding*, vol. 208, p. 103219, 2021.
[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.
[34] L. Hedegaard and A. Iosifidis, "Continual 3d convolutional neural networks for real-time processing of videos," *preprint, arXiv:2106.00050*, pp. 1–12, 2021.
[35] L. Hedegaard, A. Bakhtiarnia, and A. Iosifidis, "Continual Transformers: Redundancy-Free Attention for Online Inference," *preprint, arXiv:2201.06268*, 2022.
[36] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NeurIPS Workshop*, 2017.
[37] L. Hedegaard, "Ride the lightning," *GitHub. Note: https://github.com/LukasHedegaard/ride*, 2021.
[38] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, International Society for Optics and Photonics. SPIE, 2019, pp. 369 – 386.
[39] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *preprint, arXiv:1706.02677*, 2017.
[40] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3590–3598, 2019.
[41] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional lstm network for skeleton-based action recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1227–1236, 2019.
[42] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2684–2701, 2019.
[43] C. Caetano, F. Brémond, and W. R. Schwartz, "Skeleton image representation for 3d action recognition based on tree structure and reference joints," in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2019, pp. 16–23.
[44] B. Fernando, E. Gavves, M. José Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5378–5387.
[45] B. Li, X. Li, Z. Zhang, and F. Wu, "Spatio-temporal graph routing for skeleton-based action recognition," in *AAAI*, 2019.

Lukas Hedegaard is a PhD candidate at Aarhus University, Denmark. He received his M.Sc. degree in Computer Engineering in 2019 and B.Eng. degree in Electronics in 2017 at Aarhus University, specialising in signal processing and machine learning. With a common theme of efficient deep learning, his research endeavours span from online inference acceleration and human activity recognition to transfer learning and domain adaptation.

Negar Heidari is a Postdoctoral researcher at Aarhus University, Denmark. She completed her PhD in Signal Processing and Machine Learning at the Department of Electrical and Computer Engineering, Aarhus University in 2022. Her current research interests include machine learning, deep learning and computer vision with a focus on computational efficiency.

Alexandros Iosifidis (SM'16) is a Professor at Aarhus University, Denmark. He serves as Associate Editor in Chief for Neurocomputing (for Neural Networks research area), as an Area Editor for Signal Processing: Image Communication, and as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems. He was an Area Chair for IEEE ICIP 2018-2022 and EUSIPCO 2019,2021, and Publicity co-Chair of IEEE ICME 2021. He was the recipient of the EURASIP Early Career Award 2021 for contributions to statistical machine learning and artificial neural networks. His research interests focus on neural networks and statistical machine learning finding applications in computer vision, financial modelling and graph analysis problems.