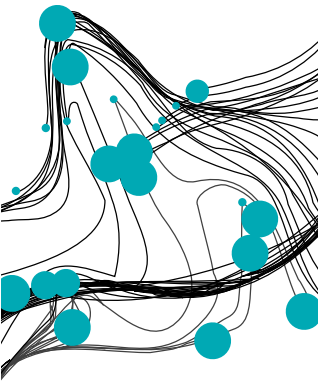
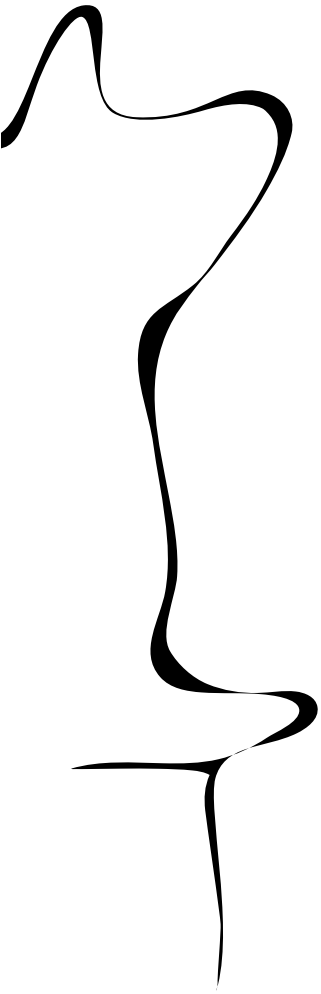


DMB

DATABASE MANAGEMENT
AND
BIOMETRICS

.16277



INVESTIGATING VISION TRANSFORMERS FOR HUMAN ACTIVITY RECOGNITION FROM SKELETAL DATA

Ajay Mathew Joseph

MSC. FINAL PROJECT ASSIGNMENT

Committee:
E. Talavera Martínez, PHD (1st supervisor)
dr.ir. L. Spreeuwers
dr. J. Reenalda

January, 2023

2023DMB0001
Data Management and Biometrics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Investigating Vision Transformers for Human Activity Recognition from Skeletal Data

Ajay Mathew Joseph
Faculty of EEMCS
University of Twente
Enschede, The Netherlands

Estefanía Talavera
Faculty of EEMCS
University of Twente
Groningen/Enschede, The Netherlands

***Abstract*—Transformers are increasingly being used for different kinds of applications these days. Recent works show that vision transformers can also demonstrate great capacity in solving Human Activity Recognition tasks based on skeletal trajectories. However, there are still certain aspects of them that are left unexplored, with respect to the input representation as well as the model architecture. We investigate two aspects of the problem: first, we use skeletal keypoint trajectories as inputs which are decomposed locally as well as globally. Secondly, we introduce convolutional learning in to transformers by using tubelet embeddings which we assume could extract better spatio-temporal information. We inspect our model on two different datasets, NTURGB+D 120 and HR-Crime. We observe that decomposing the keypoints globally and locally does not improve the performance. We also observe that incorporating a tubelet embedder to a simple transformer architecture gives similar results as the baseline results with significantly lesser computational costs. We also discuss the limitations of our work and what could be done to improve it.**

***Index Terms*—Human activity recognition, Transformers, Skeleton based activity classification, Tubelet embeddings, Keypoint decomposition**

I. Introduction

In the past decade, there have been great advancements in the technological field in our world. Humans now live in a digital ecosystem. We are surrounded by motion sensors and cameras almost everywhere we go. In public places, governments and businesses have setup closed-circuit television cameras (CCTV) to monitor various activities like traffic, crowds and even to detect anomalous activities [1]. Even though humans have advanced a lot technologically, and complex surveillance systems are in place, we still observe a lot of criminal activities happening around us in plain sight. However, it's also a humongous task to manually inspect each video [2, 3].

In the recent years, price of computing has gone down and the usage of digital systems have gone up [4, 5, 6] and this brings us an opportunity to make use of Deep Learning applications for automating tasks like these. In the Deep Learning world, these tasks come under the field of Human Activity Recognition (HAR). HAR aims to understand and recognize human behaviour from data like videos, skeleton trajectories [7, 8, 9, 10], depth maps [11], infrared data [12], audio data [13] etc. It has a wide range of applications like surveillance systems [1], virtual/augmented reality [14], human monitoring [15], etc. For example, identifying a criminal activity from a live CCTV video.

Most of the previous HAR works had focused on visual features extracted from videos like Spatio-Temporal Interesting Points (STIP) [16, 17], and recently, experiments with other kinds of data like those mentioned before have picked up. Deep learning methods also primarily focused on video data, however, they are computation intensive. Also, video features contain other kinds of information like background noises, lights, clothing etc. which could influence the results. Other features mentioned above have an advantage that they could be more descriptive compared to videos, like human skeletons which are compact, strongly structured and semantically rich. However, obtaining the skeletal keypoints is a different task and it comes under the field of pose estimation.

These days, transformers are increasingly used for video analysis and have obtained state of the art performances. Since the domains of skeletal and video activity recognition are similar, recent advancements related to vision transformers (used for video related tasks) can also be explored with respect to skeletal trajectories. Even though transformer based methods demonstrated great capacity in solving HAR tasks, there are still areas left to be explored, which could help us understand better about how transformers work for HAR tasks. The input representation of the skeletal trajectories and the usage of different types of embeddings are areas which are less explored in this task. We explore those areas in this work.

Video Vision Transformer (ViViT) [18] introduced by the Google Research team used tubelet embeddings which corresponds to 3D convolution capturing temporal and spatial information at once from videos. This introduced convolution to transformers. Typically, transformers are pretrained on large datasets since they lack inductive bias. However, convolutional networks have inductive bias and we investigate whether tubelet embeddings could help overcome the lack of inductive bias for transformers. We propose an architecture which incorporates the tubelet embedding. For each body part, we pass the keypoints through a 3D convolution layer whose outputs are passed to a transformer encoder. This tubelet embedder is explored as an alternative for the patch embedding layer in general transformers.

We also investigate a different input representation, with the keypoints decomposed locally and globally. We build our

experiments on the architecture proposed by Zheng et.al. [19] (originally proposed for pose estimation), which gave state of the art performance on HRC dataset [7]. We explore the following research questions:

1) Can skeletal keypoints which are decomposed locally and globally better represent the skeletal trajectories for HAR tasks with transformers compared to conventional image coordinates?
2) Can tubelet embeddings replace the patch embeddings in transformers for HAR tasks to encode information such as movement of body parts and reduce the computational complexity?

Our major contributions are:

1) We estimate the performance of NTU-RGB+D 120 dataset (see section IV) on the transformer models used in [7].
2) We demonstrate that keypoints which are decomposed locally and globally cannot benefit HAR tasks with a simple transformer architecture.
3) We incorporate a Tubelet Embedder to different transformer architectures and demonstrate that they can help match the performance of the baseline architecture with much lesser computational costs.
4) While obtaining similar results as to the baseline architectures, we improve the interpretability of the results by analyzing them visually using attention heatmaps and T-SNE plots.
5) We compare two different input representations for the Tubelet Embedder and discuss their performances.

The rest of the paper is divided as follows. In Section II, we will discuss about the scientific background of this research. In Section III, the methodology of this project will be discussed and then in Section IV, the experimental setup will be described. Section V describes the results and Section VI discusses the results obtained. Section VII will finally conclude the research work.

II. Scientific Background

In this section, we would first discuss about the technical background for this work and then discuss the related works in this field.

A. *Technical background*

1) ***Transformers****:* Transformers were introduced by Vaswani et.al. in 2017 [20] with the purpose of using in Natural Language Processing (NLP) tasks. As per the authors, "the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution" [20]. This helps to extract a better meaning for each word and also to build better relations with other words. Transformers basically use self-attention mechanism to extract features for each word by figuring out how important all the other words in the sentence are, with respect to the given word.

Before Transformers, Recurrent Neural Networks (RNNs) dominated the NLP space, especially for dealing with sequence-to-sequence problems. However, they had few serious limitations. They could not retain information while dealing with long sentences. The information from the initial words were lost by the time the model reaches the later words. This was because at any time, the decoder dealt only with the latest hidden state. In the case of CNNs, the receptive field is limited by the size of the filters and the number of convolutional layers used. Increasing these hyperparameters increases the complexity of the model and can lead to problems like vanishing gradients. It was for these reasons, the concept of attention mechanism was introduced. With attention mechanism, the model related every word with every other word. Hence, no information could get lost.

Discussing about the structure of a vanilla transformer, it's inputs are in the form of tokens and these tokens are represented using embeddings. For example, each word in a sentence could be a token.

The next step is to add positional embeddings. In the case of RNNs, the model could understand the order of the words since they were processed sequentially but here in the case of transformers, it processes all words at once, hence there is no way to retain the information about the token's positions. For this reason, positional embeddings were introduced. These are embeddings which could be added to the embeddings of the words. There are multiple ways to use these embeddings. They could be fixed or learnable. The idea is that as training progresses, the model would learn to understand the information related to the positions of the words.

These embeddings would be the inputs to the core part of the transformer, the encoders and decoders. Also, these embeddings would maintain the same shape and structure until the very final layer. The transformer has a series of encoders followed by a series of decoders. Each encoder has a self attention layer which is followed by a feedforward neural network. The decoder also has a similar structure, with a vanilla attention layer between the final encoder and the corresponding decoder. This can be observed in Figure 1.

The output of the final decoder can be passed through linear and softmax layers to get the final output.

However, transformers assume minimal prior knowledge about the structure of the problem as compared to their convolutional and recurrent counterparts, and hence, they are typically pre-trained using pretext tasks on large-scale (unlabelled) datasets. The learned representations are then fine-tuned on the downstream tasks in a supervised manner to obtain favorable results.

2) ***Self attention****:* Before discussing self attention, let's have a look at vanilla attention. Attention mechanism is the most important component of the transformer architecture. Let's look at attention from the context of Natural Language Processing. Consider a model which could translate a sentence from one language to another. Here, the attention mechanism helps the model look at every other words while making the decision. So the model could 'attend' to every other word with

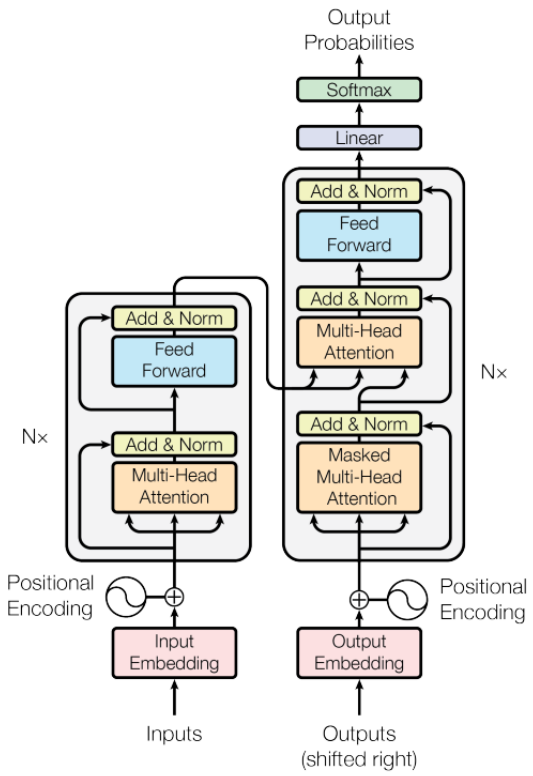


Fig. 1: Architecture of a transformer [20].

respect to a certain word before making a decision. Hence, every word could contribute to the decision making process. Vanilla attention mechanism is illustrated in Figure 2a.

Self attention is a mechanism in which the words in a sentence could attend to other words in the same sentence. This is helpful when we have ambiguous words whose meaning changes depending on the context. In that case, depending on the attention from the other words, the model could take a decision. Self attention mechanism is illustrated in Figure 2b.

As per [20], "An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key". To calculate the attention, as seen in equation 1, we initially do a matrix multiplication of the Query and Key matrices, divide it by the square root of the dimension of the embeddings and apply a softmax function before matrix multiplying it by the Value matrix. Further explanation of the attention mechanism is out of the scope of this section.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

B. *Related Works in the field of HAR*

In this section, we summarize some of the related works in the fields related to HAR.

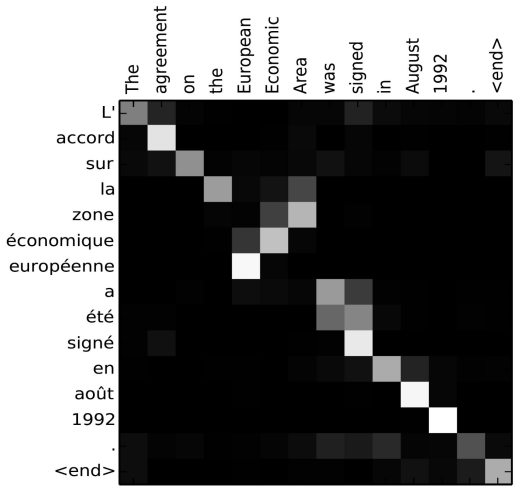
*1) **Human Activity Recognition**:* HAR on video data involves mainly two approaches: using handcrafted features and using Deep Learning models. While methods using handcrafted features used features like Spatio-Temporal Interesting Points (STIP) [16, 17] and skeleton trajectories [23, 24], deep learning methods [25, 26] proposed architectures based on convolutional networks making using of intermediate representations like Video Spatio-Temporal Map (VSTM).

Many works have tried combining both approaches, using deep learning with handcrafted features. They used different architectures like Recurrent Neural Networks (RNN) [27, 28, 29], Convolutional Neural Networks (CNN) [30, 31, 32] and Graph Convolutional Network (GCN) [33, 34, 35]. However, they were affected by their limitations [36, 37, 38]. CNN methods considered only correlation between neighbouring joints while RNN could attend only to short term contexts. GCN demonstrated difficulty in capturing relationship between joints which are far away. Until transformers became popular, GCNs dominated the state of the art works in skeletal HAR.

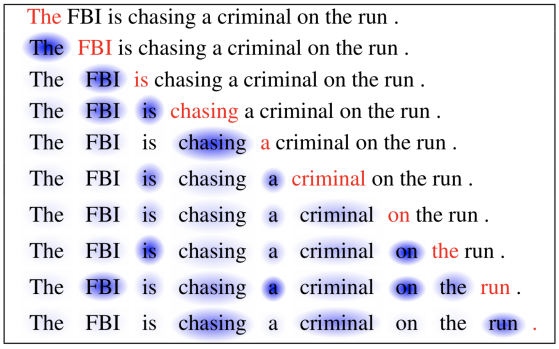
Most of the state of the art works in skeletal activity recognition has been done using Graph Convolution Networks as seen in Table I. Nowadays transformers exhibit great ability in solving sequential tasks and have lately replaced architectures like Recurrent Neural Networks (RNN). Thus, transformers are also used in HAR tasks these days [8, 7, 39, 40, 41, 42]. We use transformers in this work and they will be discussed in detail in section II.

Discussing about the usage of transformers for activity recognition, Gavrilyuk et.al. recently used transformers for group activity recognition from videos with the help of a 2D pose network and 3D CNN [61]. Also, for group activity recognition, Li et.al. introduced GroupFormer [62], which used a Clustered Spatial-Temporal Transformer which grouped individuals and made use of inter-group and intra-group relations to capture global activity. In [63], Li et.al. used the reflection and refraction of WiFi signals from humans as input to a transformer to detect activities with the help of a transformer.

Skeleton trajectories were used for the first time by Morais et.al. [29]. However, they used RNNs for the task. In another work, Morais et.al. [9] demonstrated that using keypoints which are locally and globally decomposed could benefit HAR tasks. They factored out the the human behavioral irregularity into factors like location, velocity, direction, etc. For this, they decomposed the skeletal keypoints to local and global components, where the global component tracked the whole body, while the local component tracked the skeleton in the canonical coordinate frame of the body's bounding box. The global component carries information about the shape, size and rigid movement of the human bounding box. The local component models the internal deformation of the skeleton and ignores the skeleton's absolute position in relation to the environment. We investigate whether this could improve the performance while being used with transformers. We explore two kinds of input representations for the global and local decomposed keypoints. In the first representation, we keep the



(a) Attention mechanism: Words from one sentence attending to words from another sentence [21]. Considering the image as a heatmap, the intensity shows how much the word attends to the other word.



(b) Self Attention mechanism: Words from one sentence attending to words from the same sentence [22]. The red word is the word which is being attended and the intensity of blueness indicates the attention strength.

Fig. 2: Two kinds of attention mechanisms

Method	NTU-RGB+D 60	NTU-RGB+D 120	Year
LSTM + Convolution			
Va-Fusion [43]	89.1	-	2018
AGC LSTM [44]	89.2	-	2019
GCN			
AGCN [45]	88.5	-	2019
DGNN [46]	89.9	-	2019
DDGCN [47]	91.1	-	2020
ST-GCN [48]	81.5	70.7	2018
AS-GCN [49]	86.8	78.3	2019
SGN [50]	89.0	79.2	2019
RCGCN [51]	87.3	81.1	2021
2S AGCN [52]	88.5	82.5	2019
FGCN [53]	90.2	85.4	2020
Shift-GCN [54]	90.7	85.9	2020
DCGCNADG [55]	90.8	86.5	2020
MS-G3D [56]	91.5	86.9	2020
MSTGCN [57]	91.5	87.5	2022
EGCN [58]	91.7	88.3	2022
CTR-GCN [59]	92.4	88.9	2021
STF [60]	92.5	88.9	2022
Transformer			
STST [42]	91.9	-	2021
ST-TR [8]	88.7	81.9	2021
DSTA-Net [39]	91.5	86.6	2020
IIP-Transformer [40]	92.3	88.4	2021
FG-STFormer [41]	92.6	89.0	2022

TABLE I: State of the art top-1% accuracy for skeleton activity recognition tasks for two datasets, NTU-RGB+D 60 and NTU-RGB+D 120 (sorted in the order of performance for NTU120 dataset).

global keypoint and then the local keypoints of each joint. In the second representation, we couple both the data: with every local keypoint of a joint, we also keep the global keypoint of the frame.

Franco et.al. [64] used a multi-modal approach combining information from Histogram of Oriented Gradients (HOG) descriptors as well as skeletal information like the joints' 3D position and also it's orientation with respect to the sensor coordinate system. In other works, Mazi et.al.[65] and Cippitelli et.al. [66] used Support Vector Machine (SVM) techniques to dynamically cluster skeleton data.

Combining both transformers and skeletal activity recognition, Zhang et.al. introduced Zoom Transformer [67] for group activity recognition which exploited both single person motion as well as multi-person motion. Zhai et.al. [68] used RGB input as well as skeletal information to form a group activity recognizer. However, they used the transformer to process the RGB input while the skeletal data was processed by a GCN. One of the very first works making use of pure attention networks was by Shi et.al. [39]. Zhang et.al. proposed a multi-task self-supervised learning method [42] using transformers which used confusing samples in different situations to improve the robustness of the model. Plizzari et al. [8] introduced transformer self attention in skeleton activity recognition instead of graph convolution and also proposed Spatial Self-Attention module (SSA), Temporal Self-Attention module (TSA) and Spatial-Temporal Transformer network (ST-TR). While SSA was used to deal with joint relationships within a frame, TSA was used to deal with long range dependencies across the skeleton trajectory. Combining both, the ST-TR architecture was able to produce state of the art performance on various datasets. However, it also made use of convolutional modules. In [40], Wang et.al.

captured intra-body part and inter-body part dependencies and made use of partition encoding and intra-inter-part transformer. In a much recent work, Snoun et. al. [69] used CNNs to process 2D skeletal data while 3D skeletal data were dealt with transformers and proved that transformers are superior in performance. In the most recent work [41], Gao et.al. proposed a focal joints and global parts coupling spatial transformer to model the correlations of adaptively selected focal joints and that of human body parts and obtained the state of the art performance.

In her master thesis [7], Boekhoudt analyzed several transformer architectures and explored different representations for human body movement. It was built on the works of Zheng et.al. [19]. While the architecture proposed in [19] lifted a 2D pose to a 3D pose by analyzing a sequence of skeletal trajectories, Boekhoudt [7] modified the same architecture and used it to classify the skeletal trajectories by replacing the regression head with a classification head. Boekhoudt's work demonstrates that model architectures designed for pose estimation could also be modified for HAR tasks. We would be using this architecture as our baseline model.

Eventhough transformers perform greatly, they have certain limitations as well. One of them is that they have quadratic complexity since they attend to all of the input tokens. Another limitation is that they lack inductive bias. Convolutional Neural Networks (CNN) have inductive bias because we assume that closer pixels are related, and hence, we use kernels which walk over the image picking up patterns. Transformers do not have this functionality and they just attend to different tokens of the input. In ViT[70], Dosovitskiy et.al. used a hybrid model, which uses a CNN before the transformer and the patches for the transformer are extracted from a CNN feature map. Another way to overcome the lack of inductive bias is to use a large dataset and this is one reason we use the NTURGB+D 120 dataset which is large in size (see Table II).

In other experiments, Saini et.al. [71] used genetic algorithms using features like velocity and acceleration of trajectories, movement direction, radial distance, circle structure, etc. to classify the trajectories and obtained great results. Matei et.al. [72] tested different methods and showed the negative impact of dataset imbalance and also demonstrated how data augmentation methods can boost the performance in HAR tasks. In [73], Gupta et.al. introduced three different datasets, each varying on the context in which they were created. They also investigate the strength and limitations of datasets which are created in a lab environment and those which are obtained from real-life situations.

*2) **Human Pose Estimation**:* Human Pose estimation (HPE) is an important step in HAR applications and it aims to estimate human poses based on their skeleton joints from images. Human pose estimation could be done either in 2D or 3D. 3D pose estimation is majorly done in two ways: Direct estimation approaches which infers the pose from 2D images; and 2D to 3D lifting approaches, which infer 3D poses from 2D poses detected from images.

Research in this field started from extracting 3D poses from a single 2D pose of an image [74]. It then progressed to estimating poses from videos, making use of the temporal information [75, 76, 77, 78, 79, 80, 81]. Some works [82] made use of LSTM cells while some other works [76, 79, 83] made use of other parameters like bone-length and spatial-temporal relationships. In [84], Liu et.al. made use of attention mechanism to identify significant frames. Since majority of these works made use of dilated temporal convolutions to capture the global dependencies, they were limited in temporal connectivity.

Following the works of [74, 77, 84, 85], Zheng et al. [19] presented PoseFormer, using a purely transformer based architecture without involving a convolutional architecture. It used the human joint relations within one frame and also the temporal correlation across multiple frames to display a 3D human pose of the center frame.

In [86], Xu et al. proposed a simple vision transformer baseline, ViTPose for pose estimation from images. ViTPose takes as input a single image and outputs the 2D pose of the person(s) in that image. It used a simple architecture, with a ViT backbone and simple decoder for estimating the poses. With the biggest backbone, it was able to obtain state of the art performance on the MS COCO keypoint dataset [87].

While most of the works used skeleton keypoints obtained with respect to the frame boundaries, Morais et.al. [9] demonstrated that a combination of global and local decomposition of the skeletons would work in a better way. The global decomposition can track the whole body in the scene while the local decomposition can describe the skeleton within the frame of the body's bounding box (see Figure 3).

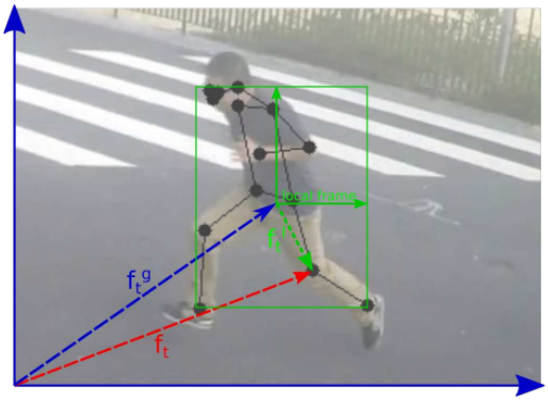


Fig. 3: Local & Global decomposition of keypoints [9].

*3) **Vision Transformers**:* In recent years, transformers are being increasingly used in the field of Computer Vision. Even though transformers were built for the purpose of dealing with Natural Language Processing tasks, researchers have extended it to the field of Computer Vision. Dosovitskiy et.al. introduced Vision Transformers (ViT) [70] and proved that a pure transformer applied directly to sequences of image patches can perform well on image classification tasks. ViT with its simple architecture has outperformed state of the art convolution based models. Thanks to the self-attention

mechanism of the transformer, global correlations across long input sequences can be distinctly captured. Earlier, in the case of using attention mechanism in the field of computer vision, attention was applied in conjunction with convolutional networks. However, ViT proved that this was not necessary.

The major modification was that the image was divided into small patches and the embeddings of these patches were passed as an input to the transformer. These patches act just like the word tokens in the case of NLP tasks. We also prepend a CLS token, which is a learnable embedding at the beginning of the sequence of embeddings which can be passed to a classification head to do the classification. It also does away with the decoder part since it does not need to output a sequence. This can be observed in Figure 4.

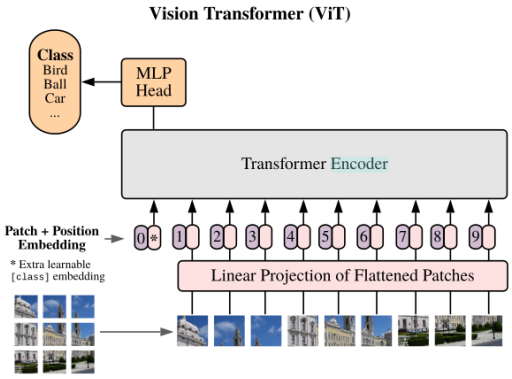


Fig. 4: Architecture of ViT [70].

Dosovitskiy et.al. [70] proves that this strategy when coupled with pre-training on large datasets matches or exceeds the state of the art on many image classification datasets which is a promising result.

Multiscale Vision Transformers (MViT) [88] uses different channel-resolution scale stages to create a multiscale pyramid of features for video classification. These layers in the early stages operate at high spatial resolution and in the later stages at spatially coarse features. MViTv2 [89] improved MViT with the help of decomposed relative position embeddings and residual pooling connections.

Transformers typically use two types of position encodings and these are absolute and relative position encodings. Vision transformers mostly used absolute encodings, however, Wu et.al. [90] experimented with relative position encodings for vision transformers and demonstrated that relative position encodings are beneficial for vision tasks.

Arnab et. al. in ViViT [18] experimented with four different pure-transformer architectures for video classification. These include Spatio-temporal attention (STA), Factorised encoder (FE), Factorised self attention (FSA), Factorised dot product attention (FDPA). STA simply passes the tokens through the transformer encoder. FE passes the tokens initially through a spatial transformer and then passes them through a temporal transformer. FSA does Multi Head Self Attention (MHSA) on spatial features first and then passes it to an MHSA layer for temporal extraction before passing it to the feed forward network. Finally, FDPA computes attention weights for each token separately over the spatial and temporal dimensions using different heads.

One interesting feature of ViViT is that they used tubelet embeddings which is an extension of ViT's embedding to 3D and it corresponds to 3D convolution (see Figure 5). It is essentially a 3D patch over a few frames extracting spatio-temporal tokens which is then linearly projected to a d-dimensional embedding. We implement this feature in our work and we hypothesize that tubelet embeddings could capture better spatio-temporal features compared to 2D patch embeddings.

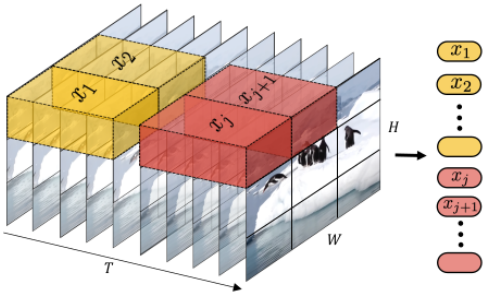


Fig. 5: Tubelet embeddings, as used in ViViT [18]. ViViT uses a 3D convolution layer over multiple video frames. For example, the yellow cuboid labelled x_1 represents a 3D kernel which extracts the embedding x_1.

ViTPose [86] is a recent work and it has been used by Yang et.al. in [91]. Yang et.al. benchmarked ViTPose on it's validation dataset which is manually annotated and performed better than other pose estimation methods. Furthermore, an object tracker ViTTrack was designed which is a siamese structure with a shared backbone encoder, ViT for feature extraction.

In this work, we explore two different aspects which are missing in the literature:

- We explore training transformers for HAR tasks with locally and globally decomposed trajectories (as proposed in [9] for the task of anomaly detection), to investigate whether they could better represent the skeletal trajectories for HAR tasks using transformers.
- We study how tubelet embeddings can be incorporated to transformer architectures for HAR tasks to investigate whether they could help encode information such as movement of different body parts. We also investigate if we can overcome the lack of inductive bias in transformers with the help of tubelet embeddings.

To the best of our knowledge, no other work available today has explored these research lines.

III. Methodology

In this section, we will discuss the model architecture as well as the input representation for both the research questions.

A. *Global and local decomposition of keypoints*

1) Model Architecture: We use the Temporal Transformer (T-Tran) architecture from [7] to investigate whether using globally and locally decomposed keypoints could improve the performance compared to the baseline results. The architecture of this transformer can be seen in Figure 6. It takes as input a sequence of skeleton poses, each representing a frame. The general input representation is of the from $X \in \mathbb{R}^{f \times (J.2)}$ where f is the number of frames and J is the number of keypoints. Here, 2 indicates that every keypoint is 2 dimensional. Each of the frames acts as a patch and is embedded to a higher dimension C using the patch embedding layer. Positional embeddings are also added to the patch embeddings and the class embedding is also concatenated at this stage. These are then passed to the Transformer Encoder layer. The final embedding of the class token is passed through a linear layer to get the classification of the sequence of frames.

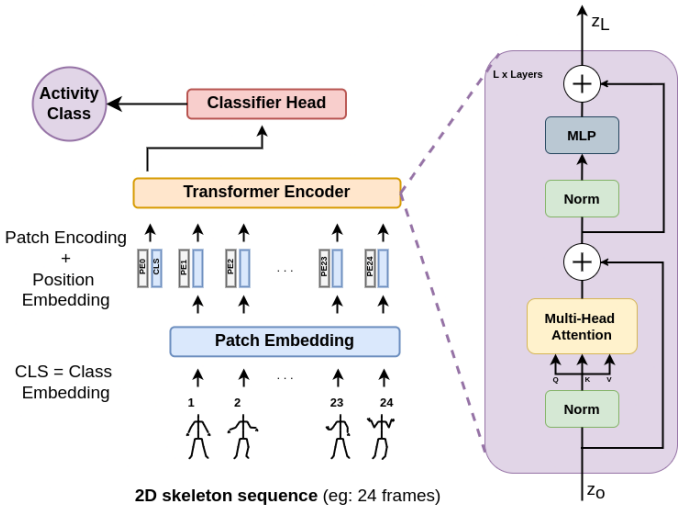


Fig. 6: Architecture of Temporal Transformer.

2) Input representation: In the baseline model, a plain list of n keypoints were used, as seen in Figure 7.

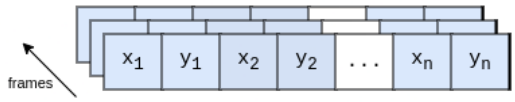


Fig. 7: Baseline input representation. (x_i, y_i) represents the x and y coordinates of the i^{th} skeletal keypoint in a frame.

For this experiment, we decompose the keypoints globally as well as locally. The global keypoint of a frame is taken as the mid-point of the bounding box defined by the keypoints of the skeletal body. This can be observed as the red point (x_g, y_g) in Figure 8.

Now, all the keypoints can be decomposed as the sum of the global keypoint (green vector) and a local keypoint (blue vector). This way, the relative position of the joints can be

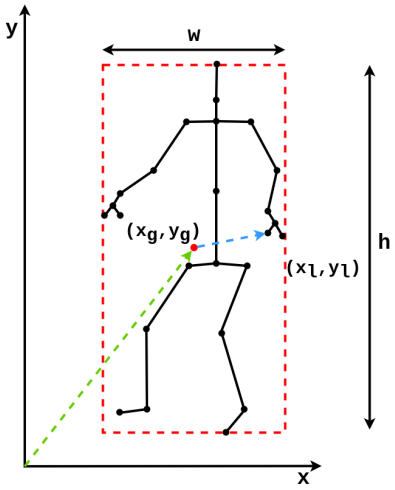


Fig. 8: Globally and locally decomposed keypoints. The red box is the bounding box formed by the skeletal keypoints and it's midpoint is the global keypoint. The local keypoint is the blue vector from the global keypoint to the keypoint.

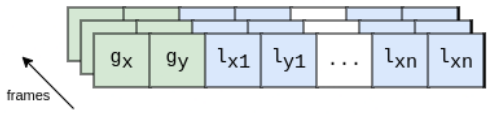
represented with respect to the bounding box. As discussed in [9], if the xy coordinates are used alone, the depth information would be missing. Hence we find the width and height of the bounding box (w, h) using Equation 2 and then use them to normalize the local components (l) using Equation 3. In Equation 2, we find the width and height of the bounding box by taking the difference of the largest and smallest x and y coordinates respectively. In Equation 3, we normalize the local components by dividing them by the width and height of the bounding box.

$$\begin{aligned} w &= max(x_i) - min(x_i) \\ h &= max(y_i) - min(y_i) \end{aligned} \tag{2}$$

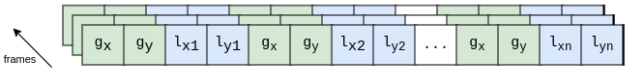
$$\begin{aligned} l_{xi} &= \frac{x_i - x_g}{w} \\ l_{yi} &= \frac{y_i - y_g}{h} \end{aligned} \tag{3}$$

We know that, while the global component tracks the movement of the human bounding box, the local component tracks the internal movement of the skeleton within the bounding box. Now we explore two different input representations, with different levels of importance for the global component :

- **Global Single (GS)**: In this setting, we keep the global keypoint at the beginning of the list of keypoints alongwith other local keypoints. This can be observed in Figure 9a. The input is of the form $X \in \mathbb{R}^{f \times ((J+1).2)}$ since we have J local keypoints and one global keypoint.
- **Global Repeated (GR)**: In the global repeated setting, we keep the same global keypoint along with all local keypoints, as seen in Figure 9b. It is of the form $X \in \mathbb{R}^{f \times (2.J.2)}$ since for each local keypoint, we also attach with it, the global keypoint.



(a) Global single input representation.



(b) Global repeated input representation.

Fig. 9: Different input representations with local (l) and global (g) decomposition of keypoints. (g_x, g_y) represents the global component while (l_{xi}, l_{yi}) represents the local component of the i^{th} skeletal keypoint in the frame.

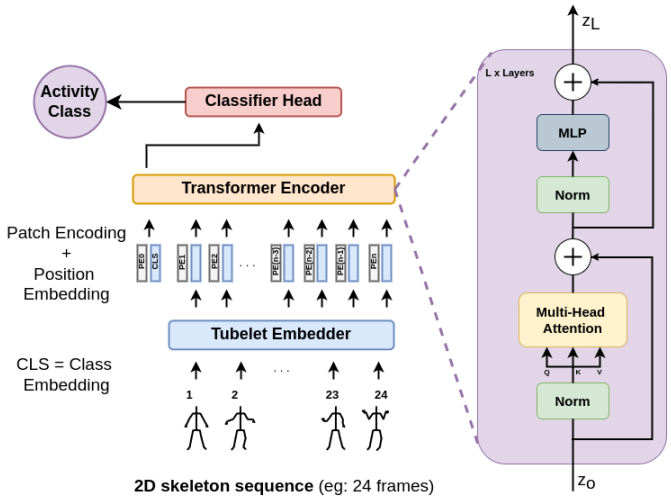


Fig. 10: Temporal Tubelet Transformer (TTubeFormer). The patch embedding in the Temporal transformer (T-Tran) has been replaced with a Tubelet embedder which embeds the whole skeleton to different embeddings.

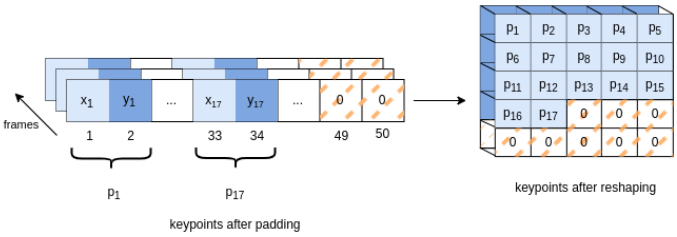


Fig. 11: Input representation for the Temporal Tubelet Transformer (TTubeFormer). In this example, we have considered a dataset with 17 keypoints. Since we have 17 keypoints, we cannot make it a square matrix. Hence, we append 16 zeroes and reshape it to a $5 \times 5 \times 2$ matrix. For a dataset with a square number of keypoints (for example, 25), it can be reshaped without padding any zeros.

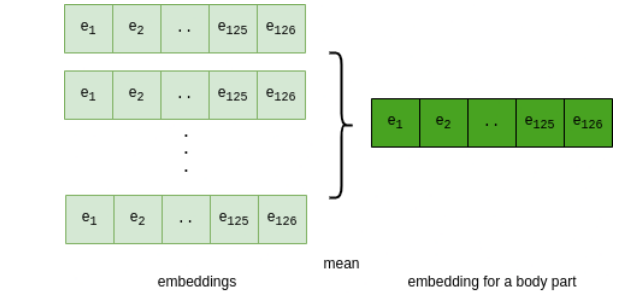
B. Exploring Tubelet Embeddings

To explore tubelet embeddings, we approach the problem as follows. Given a skeletal trajectory extracted from a video, we divide it into different segments based on the user defined segment length and assign the action label of the video to all the segments. We reshape the keypoints to a matrix and feed them into a Tubelet Embedder to obtain different embeddings as output. These would replace the patch embeddings used in the original transformers. The model architecture and input representations would be discussed detailed in the following subsections.

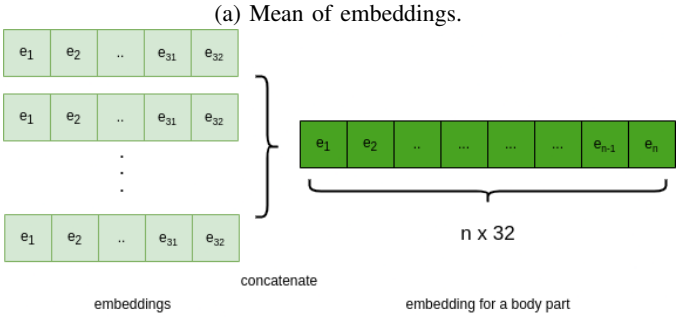
1) Model architecture: Initially, we use the same Temporal Transformer (T-Tran) used in the previous experiment as a baseline model. We replace the patch embedding layer with a Tubelet Embedder layer, as seen in Figure 10. We call this architecture, the Temporal Tubelet Transformer (TTubeFormer). The Tubelet Embedder is essentially a 3D convolution layer which takes as input, keypoints rearranged in the shape of a matrix and does 3D convolution over them using a 3D kernel. The representation of the input in the shape of a square matrix can be seen in Figure 11.

We investigate this aspect in detail by exploring two different architectures:

- **Body Part Transformer (BPFormer)**: For dealing with Tubelet embeddings, we modify the Temporal Transformer architecture which was used in the experiments for Global and local decomposed keypoints. As seen in Figure 14, we introduce a Body Part Embedder which takes as input the sequence of keypoints of a body part over multiple frames. The Body Part Embedder then embeds it to multiple embeddings using a Tubelet Embedder (3D convolutional layer). We use five different Tubelet Embedders, one for each body part (Torso, Elbows, Wrists, Knees, Ankles). Now these embeddings will be encoded either by taking their mean, or by concatenating them as shown in Figure 12a and Figure 12b.
  The Body Part Embedder thus outputs 5 different encodings, one for each body part. These are then fed to a Body Part Transformer Encoder which captures the relationship between different body parts. The class token is then passed through a linear layer to make the final classification.
- **Body Part Tubelet Transformer (BPTubeFormer)**: Here we modify the architecture to include another Transformer Encoder layer. The output embeddings of the Tubelet embedder (3D convolutional layer) are passed to a Transformer encoder to capture the temporal and spatial relationship between the different keypoints within a body part. Here we use five different Transformer Encoders, one for each body part. Thus, each encoder can exclusively learn about a body part. The output features are either concatenated or their mean is taken as in the previous architecture. The rest of the architecture is the same as the previous architecture. With the help of this architecture, we learn and interpret the movements



(a) Mean of embeddings.



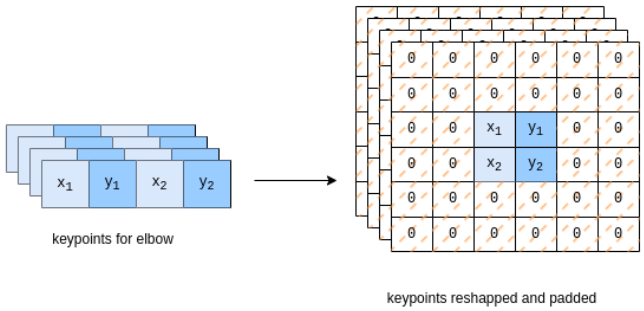
(b) Concatenation of embeddings.

Fig. 12: Embedding encoding methods. In (a), we take the mean of all embeddings. In (b), we concatenate all the embeddings to obtain a long embedding.

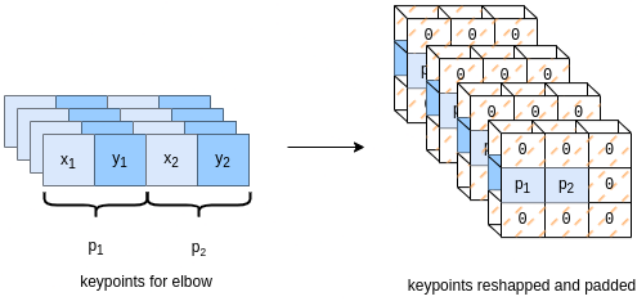
belonging to a body part.

2) Input representation: For the experiments related to tubelet embedding, we make use of two kinds of input representations. The input we have is a sequence of 1D keypoints while for dealing with 3D convolution, we need inputs in the shape of a square matrix. Hence, the difference comes in the way we transform the 1D input (list of keypoints) to a 2D or 3D representation. In both the cases, if we are concatenating the output of the Tubelet Transformer, we take the body part with most number of keypoints as the standard and pad the keypoint data of all other body points to meet the shape of that body part representation. For example, consider a dataset which has a body part containing 9 body points. This means we have 18 values to represent that body part. We can rearrange them in the shape of 3×6. To transform this to a square matrix shape, we need to pad 3 rows to this matrix, so that the whole matrix shape becomes 6×6. For a smaller body part which has only 2 keypoints, we can reshape the 4 values to a 2×2 matrix but we also need to pad 4 rows and 4 columns, as seen in Figure 13a, to match the matrix shape of the torso body part. This is necessary because different shapes would mean different number of embeddings as the result for 3D convolution and it would be impossible to concatenate them to the same dimension. The two types are:

1) 1-Channel: In the 1 Channel representation, we rearrange the 1D input to a 2D matrix and then pad the non-matching dimensions with zeros, as seen in Figure 13a.
2) 2-Channel: In the 2 Channel representation, we rearrange the 1D input to a 3D matrix with the third dimension specifying the xy coordinates and then pad the non-matching dimensions with zeros, as seen in Figure 13b.



(a) 1-Channel representation.



(b) 2-Channel representation.

Fig. 13: Different input representations for the keypoints to explore Tubelet Embeddings. The orange dashed cells labelled 0 represent the padded values.

IV. Experimental Setup

A. *Datasets & Preprocessing*

In this section, we analyse the two datasets which we have used in this research work. A brief summary of them can be seen in Table II and sample images can be seen in Figure 17.

The train and test datasets are obtained based on a 80:20 train:test split. For this, the dataset is divided into four quartiles based on the trajectory lengths. Each of these quartiles are then divided based on a 80:20 split so that trajectories of all lengths are present equally after the split. The training dataset is again divided to obtain a validation dataset while doing cross-validation. During cross-validation, the training dataset is divided into three folds.

During preprocessing, all trajectories which are shorter than a specified segment length are removed. All trajectories are then divided in to segments based on that segment length. These segments are all then labelled with the activity class label of the parent trajectory from which they were segmented.

The two different datasets that we analyse are:

*1) **HR-Crime**:* The HR-Crime dataset [94] is a subset of UCF-Crime dataset [95] which is a surveillance video dataset

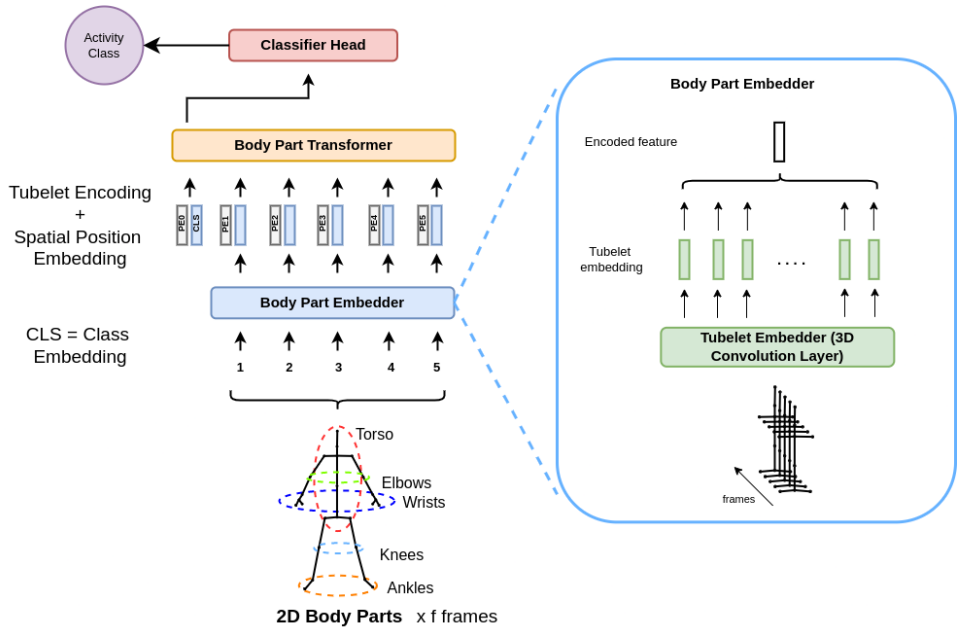


Fig. 14: Body Part Transformer (BPFormer). A body part embedder is used for each body part to obtain the tubelet embeddings (3D convolution over the reshaped body part keypoints over time). In the next step, the mean of these embeddings are obtained or they are concatenated. This leaves us with five different embeddings, one for each body part. The class token embedding is concatenated to them and then passed to the Body Part Transformer along with the position encodings.

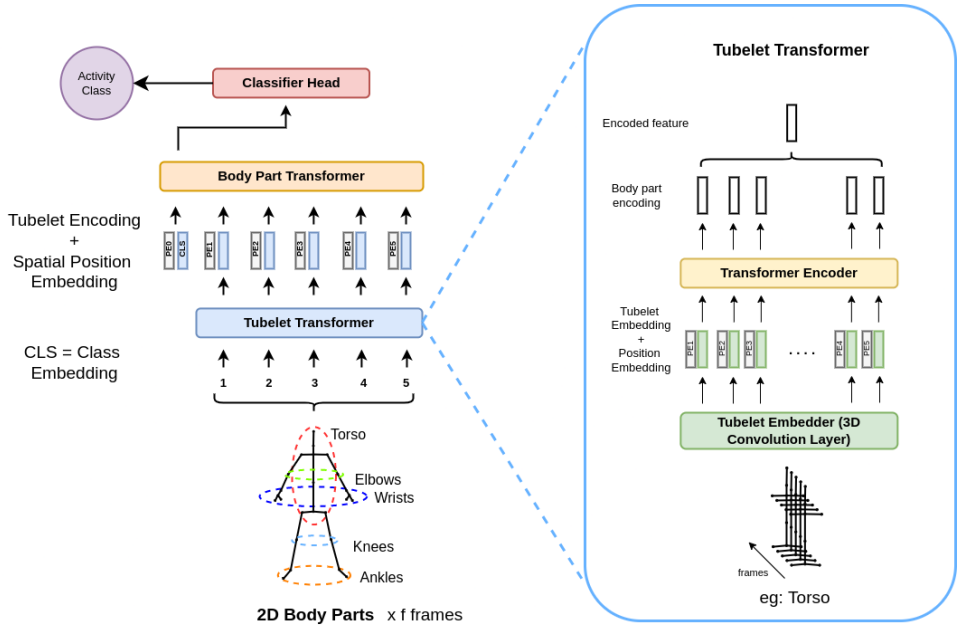


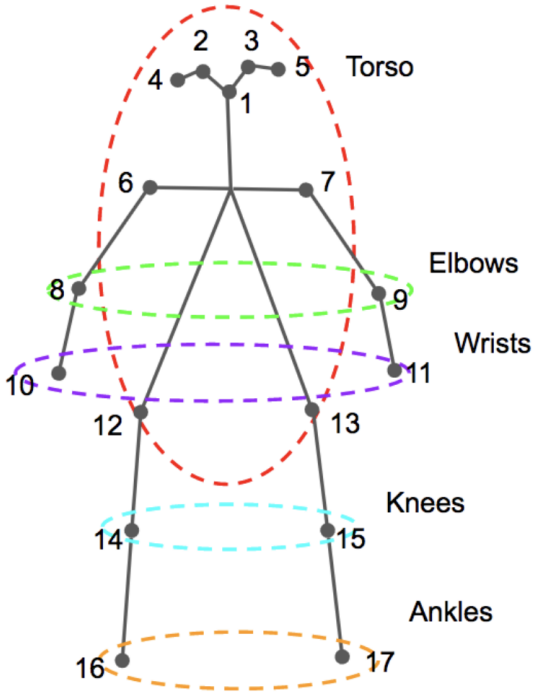
Fig. 15: Body Part Tubelet Transformer (BPTubeFormer). We use 6 different transformer modules in this architecture. One for each body part and finally the Body Part Transformer to extract information between each body part. The Tubelet embedder gives tubelet embeddings (3D convolution over the reshaped body part keypoints over time) for each body part. These embeddings are then fed to a Transformer Encoder alongwith positional encodings. The encoder outputs different embeddings. The mean of these embeddings are taken, or they are concatenated. This leaves us with five different embeddings, one for each body part. The class token embedding is concatenated to them and then passed to the Body Part Transformer along with the position encodings.

(a) Fighting (b) Explosion (c) Shoplifting

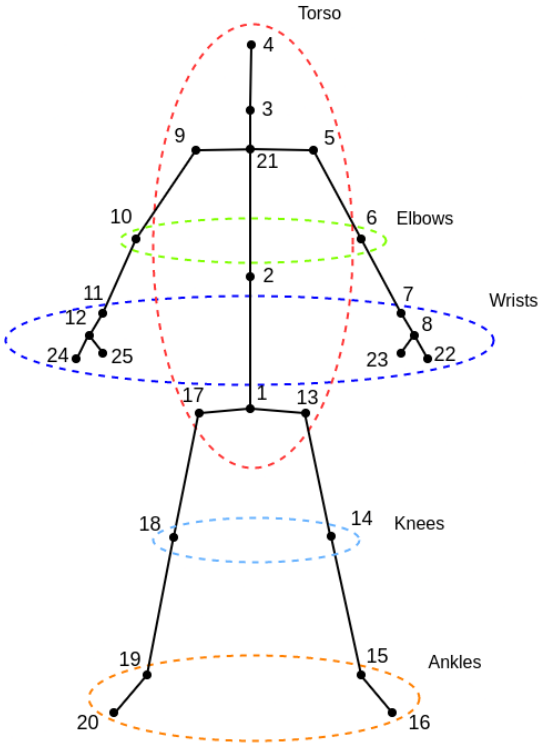
Fig. 16: HR-Crime dataset samples from three activity classes.

(a) Kicking (b) Shaking hands (c) Typing

Fig. 17: NTU120 dataset samples from three activity classes.



(a) Skeleton keypoints for the HRC [92] dataset. It has 17 keypoints in total. The oval shaped groupings represent different body parts of the skeleton.



(b) Skeleton keypoints for the NTU120 [93] dataset. It has 25 keypoints in total. The oval shaped groupings represent different body parts of the skeleton.

Fig. 18: Skeleton keypoints in HRC [92] and NTU120 [93] datasets.

	HRC	NTU120
Keypoints	17	25
Videos	789	114,480
Classes	13	120
Train segments (millions)	3.5	5.2
Test segments (millions)	0.86	1.3
Mean frame length	315	69

TABLE II: Statistics of the HRC [92] and NTU120 [93] datasets. For Train and test segments, those with atleast 24 frames are considered.

consisting of 789 human-related anomaly videos and 782 human-related normal videos. It consists of anomaly videos from 13 categories: Abuse, Arrest, Arson, Assault, Burglary, Explosion, Fighting, Road Accidents, Robbery, Shooting, Shoplifting, Stealing and Vandalism [92]. A box plot of the number of trajectories per video in HRC dataset could be seen in Figure 19. As can be seen, most of the videos have around 30 trajectories while there are few videos which have more than 200 trajectories with one video even having more than 600 trajectories.

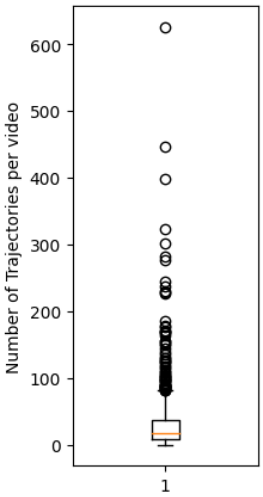


Fig. 19: Box plot of number of trajectories per video in HRC [92] dataset.

The skeletal data was given and they were extracted in three steps: first, the human bodies were detected using YOLOv3-spp [96] and then AlphaPose [97] was used to detect the body skeletons and finally, the skeletons were tracked using PoseFlow [98]. This also means that the trajectories extracted are not perfect. The skeletons thus extracted had 17 keypoints and a sample skeleton can be seen in Figure 18a.

One limitation of the HR-Crime dataset is that it has annotations at the video level and not at the trajectory level [72]. Hence, one video might contain the trajectories of many people of which only one might be involved in the criminal activity. One possible way to overcome this limitation is to assign an anomaly score to each trajectory and group them in to two clusters.

*2) **NTU RGB+D**:* The NTU RGB+D [99] is a human action recognition dataset which is large scale and collected from a lab environment. It has two versions, NTU RGB+D 60 and NTU RGB+D 120. NTU RGB+D 60 consists of 56,880 action samples from 60 action classes collected from 40 different human subjects while NTU RGB+D 120 consists of 114,480 action samples from 120 action classes. For both the datasets, the activity classes can be seen in Appendix:Table XVII and Appendix:Table XVIII. It has 4 different modalities of data, namely, RGB videos, depth map sequences, 3D skeletal data and infrared videos. The action classes consists mostly of daily actions and also few health-related actions and mutual actions. The skeletal data consists of information of 25 human body joints as seen in Figure 18b. Some statistics about the dataset with the number of videos with different number of trajectories can be seen in Table III.

No. of trajectories	No. of videos
1	85834
2	27014
3	1088
4	9

TABLE III: Number of videos with different numbers of trajectories in the NTU120[93] dataset.

The advantage of this dataset is that it contains high definition videos while it also means that the videos do not have a real life nature. Also, since the skeletal trajectories are extracted using sensors, they are the ground truth, contrary to HRC dataset. In this work, we use the NTU RGB+D 120 (NTU120) version of this dataset.

B. Baseline Approaches

As a baseline, we take the results obtained by the HRC and NTU120 dataset on the architectures used by [7]. The results can be seen in Table IV and Table V. It is interesting to note that while a spatial-temporal model gave the best results for the HRC dataset, a temporal model performed better than a spatial-temporal in the case of the NTU120 dataset.

Model	C	f	Balanced Accuracy
T-Tran-1 [7]	256	24	0.4760 ± 0.0090
T-Tran-2 [7]	128	12	0.4325 ± 0.0057
T-Tran-3 [7]	256	12	0.4071 ± 0.0013
T-Tran-4 [7]	128	12	0.3887 ± 0.0013
ST-Tran [7]	32	60	**0.4926 ± 0.0043**
SBPT-Tran [7]	16	60	0.4876 ± 0.0110

TABLE IV: Results for HRC [92] dataset on the models used in [7]. C is the embedding dimension and f is the number of frames. T-Tran [7] architecture can be seen in Figure 6, ST-Tran [7] in Appendix:Figure 27 and SBPT-Tran [7] in Appendix:Figure 28.

C. Implementation Details

The experiments in this research work are carried out using the PyTorch machine learning framework which is based on

Model	C	f	Balanced Accuracy
T-Tran-1 [7]	256	24	0.4669 ± 0.0059
T-Tran-2 [7]	128	12	**0.4830 ± 0.0030**
T-Tran-3 [7]	256	12	0.2425 ± 0.0033
T-Tran-4 [7]	128	12	0.3264 ± 0.0013
ST-Tran [7]	32	60	0.3812 ± 0.0116
SBPT-Tran [7]	16	60	0.4370 ± 0.0012

TABLE V: Results for NTU-120 [93] dataset on the models used in [7]. C is the embedding dimension and f is the number of frames. T-Tran [7] architecture can be seen in Figure 6, ST-Tran [7] in Appendix:Figure 27 and SBPT-Tran [7] in Appendix:Figure 28.

the Torch library. The experiments were run on the EEMCS-HPC Cluster which is part of the many clusters in University of Twente. The cluster consists of many nodes equipped with GPUs for running GPU intensive jobs. The SLURM HPC scheduler is used to manage the jobs submitted to the cluster.

The models are trained with a learning rate of 0.001. We also use an early stopping criteria which would stop training if there is no improvement in the validation loss for 3 epochs. The batch size was set to be 1000, however, few of the models took more memory and hence the batch size had to be reduced to 500 or 100. The weights were updated using the Adam optimizer and the loss calculated was cross entropy loss.

We designed the following experiments in this work:

1) Experiment 1: Global and Local decomposition of key-points:
   a) Experiment 1.1: T-Tran [7] with Global Single input representation with HRC and NTU120 datasets.
   b) Experiment 1.2: T-Tran [7] with Global Repeated input representation with HRC and NTU120 datasets.
2) Experiment 2: Exploring Tubelet Embeddings:
   a) Experiment 2.1: TTubeFormer with whole body input representation with HRC and NTU120 datasets.
   b) Experiment 2.2: BPFormer with 1-Channel and 2-Channel input representation with mean and concatenated embeddings with HRC and NTU120 datasets.
   c) Experiment 2.3: BPTubeFormer with the 2-Channel input representation and concatenated embeddings with HRC and NTU120 datasets.
   d) Experiment 2.4: BPTubeFormer with the 2-Channel input representation and concatenated embeddings with HRC and NTU120 datasets with replication padding for the inputs.
   e) Experiment 2.5: BPTubeFormer with the 2-Channel input representation and concatenated embeddings with HRC and NTU120 datasets with different strides for the tubelet embedder.
   f) Experiment 2.6: BPTubeFormer with the 2-Channel input representation and concatenated embeddings with HRC and NTU120 datasets with different segment lengths for the tubelet embedder.

D. Validation

We evaluate the models in two ways, qualitatively and quantitatively. Quantitative evaluation would help us determine which model/architecture performs the best while qualitative evaluation would help us understand how the transformer learns and interprets the problem.

Quantitative evaluation: To evaluate the models quantitatively, we use various metrics. We report the metrics with their mean and standard deviation, since we do 3 fold cross validation. Since we are dealing with multiclass classification with imbalanced classes, we use weighted metrics to get an estimate of the results which accounts for the class imbalance. These include Balanced accuracy, which is the average of the recall obtained for each class, and also the weighted F1 score.

In the end, we summarize the results using a confusion matrix for the best performing models. To analyze the complexity of the models, we also compare the number of multiply–accumulate (MAC) operations as well as the total number of trainable parameters in the model. The MAC operation computes the product of two numbers and adds that product to an accumulator ($a \leftarrow a + (b \times c)$). It is the most common operation used in machine learning models due to its usage in matrix operations. These are calculated using the THOP[1] library.

Qualitative evalutaion: For qualitative evaluation, we used the trained models from our experiments. The models would be used to predict the activity of the class. We can then generate heatmaps using the attentions scores of the Body Part Transformer and the Tubelet Transformer to explore which parts of the input the different heads attend to. In the body part transformer, we would consider the attention scores between different body parts, and in the tubelet transformer, we would consider the attention scores between different tubelet embeddings for each body part. The proper classifications and misclassifications would also be analysed.

We also plot the t-distributed Stochastic Neighbor Embedding (T-SNE) plots [100] and Silhouette plots [101] of the class token embeddings generated by the BPTubeFormer models for the test sets from both the datasets. T-SNE essentially helps us to understand high dimensional data and project it to lower dimensional space. It uses a similarity measure between pairs of instances in the higher dimensional and lower dimensional space. This measure is used to project the embeddings to lower dimensional space. T-SNE plots are generated using the Tensorboard[2] toolkit.

Silhouette plots demonstrate how well objects are classified as clusters of data. It is a measure of how similar is an object to it's own cluster and how different it is from other clusters. The similarity is called cohesion and the difference is called separation. Silhouette values range from -1 to +1 where +1 shows proper matching while -1 shows poor matching. Silhouette plots are generated using the Scikit-learn library [102].

[1]THOP : https://github.com/Lyken17/pytorch-OpCounter
[2]Tensorboard : https://www.tensorflow.org/tensorboard

V. Results

A. *Quantitative Evaluation*

Experiments 1.1 and 1.2: Local and Global decomposition of keypoints

The results for experiments with locally and globally decomposed keypoints can be observed in Table VI. We experimented with two different input representations, namely, Global Single and Global Repeated. In Global Single representation, we keep a single copy of the global keypoint while in the Global Repeated representation, we keep a copy of the global keypoint alongwith all local keypoints. We have experimented these representations on the Temporal Transformer architecture with an embedding dimension of 256 and segment length of 24 since these parameters gave the best performance on the HRC dataset. The results show that for the HRC dataset, the Global Single representation gave a balanced accuracy of 0.4122 ± 0.0004 while the Global Repeated representation gave 0.3919 ± 0.0037. For NTU120 dataset, the Global Single representation gave 0.4473 ± 0.0016 while Global Repeated representation gave 0.3959 ± 0.0102. For both the cases, the F1 score also follows a similar pattern. Even though the difference is small, we observe that keeping a copy of the global keypoint with all local keypoints do not improve the performance. Also, altogether, using globally and locally decomposed keypoints do not improve the performance when compared to the baseline input represention shown in Figure 7.

Dataset	Input Type	Balanced Accuracy	F1 score
HRC [92]	GS	**0.4122 ± 0.0004**	0.4500 ± 0.0021
	GR	0.3919 ± 0.0037	0.4309 ± 0.0030
NTU120 [93]	GS	**0.4473 ± 0.0016**	0.4406 ± 0.0005
	GR	0.3959 ± 0.0102	0.3929 ± 0.0103

TABLE VI: Results of Experiments 1.1 and 1.2 with locally and globally decomposed keypoints implemented on a Temporal Transformer with embedding dimension 256 and segment length 24. GS indicates the Global Single representation and GR indicates the Global Repeated representation.

Experiment 2.1: TTubeformer with whole body inputs

The results for Experiment 2.1 can be seen in Table VII. We used the TTubeformer which is the Temporal Transformer with the patch embbeding layer replaced with the Tubelet Embedder. We observe that for the HRC dataset, we obtained a balanced accuracy of 0.4871 ± 0.0028 while for the NTU120 dataset, it came out as 0.3904 ± 0.0010. Comparing it with the baseline results, the accuracy for the HRC dataset is very close to that of the ST-Tran [7] architecture, while for the NTU120 dataset, there is a big difference of nearly -10%. This suggests that, for the HRC dataset, TTubeFormer architecture can match the performance of the baseline model with the Tubelet Embedder.

Experiment 2.2: BPFormer with 1-Channel and 2-Channel input representation with mean and concatenated embeddings

Dataset	Balanced Accuracy	F1 Score
HRC [92]	0.4871 ± 0.0028	0.5234 ± 0.0036
NTU120 [93]	0.3904 ± 0.0010	0.3887 ± 0.0013

TABLE VII: Results of Experiment 2.1 with the TTubeformer and whole body inputs ($C = 256$, $f = 24$).

Table VIII shows the results for the Experiment 2.2 with different input representations and embedding combination techniques for BPFormer.

We investigate two input representations here, namely, 1-Channel and 2-Channel representation. With respect to embedding combination technqiues, we investigate taking the mean of the embeddings as well as concatenating the embeddings. We do experiments with all combinations of these techniques.

As it can be seen, concatenating the embeddings constantly outperformed taking the mean of the embeddings. With the 1-Channel representation, for HRC dataset, concatenation gave a balanced accuracy of 0.4617, 5% higher than taking the mean. For NTU120, difference was much higher, +17%.

With the 2-Channel representation, for the HRC dataset, the mean method and the concatenation method produced very similar accuracies, around 0.4450. However, for NTU120, concatenation method gave a balanced accuracy of 0.4560, nearly 15% higher compared to the mean method.

Now comparing the two different input representations, for NTU120, the 2-Channel representation clearly outperformed the 1-Channel representation by nearly 7%. However, in the case of HRC dataset, the 1-Channel representation slightly outperformed the 2-Channel representation. While the 1-Channel representation with concatenated embeddings gave a balanced accuracy of 0.4617, the 2-Channel representation with the same embedding combination method gave 0.4472. Since the difference is negligible and the 2-Channel representation takes less number of operations, we take the 2-Channel representation to be performing better than the 1-Channel representation.

Experiment 2.3: BPTubeFormer with the 2-Channel input representation and concatenated embeddings with HRC and NTU120 datasets

Here we analyze the performance of the BPTubeFormer which uses a Tubelet Transformer alongwith the Body Part Transformer, on the two datasets with different kernel sizes for the Tubelet Embedder. Since all body parts are reshaped to (2, 3, 3), we start with a kernel size of (1, 3, 3) and only change the temporal dimension. Following the ViViT [18] approach, we keep the stride same as the kernel size. We only change the temporal dimension since changing the other dimensions would not be useful because the stride is same as the kernel size. The segment length is set as 24 while the embedding dimension is 32.

We observe in Table IX that, for the HRC dataset we were able to achieve a balanced accuracy of 0.4982 with a kernel size of (2, 3, 3) while the least performance was obtained with a kernel size of (8, 3, 3) which gave a balanced accuracy of 0.4686, nearly 3% lesser compared to the best performance.

Dataset	Model Type	Channels	Embedding	C	f	Balanced Accuracy	F1 Score
HRC [92]	BPFormer-V1	1	Mean	256	24	0.4129 ± 0.0022	0.4480 ± 0.0026
	BPFormer-V2	1	Concatenate	32	24	**0.4617 ± 0.0048**	0.4990 ± 0.0034
	BPFormer-V3	2	Mean	256	24	0.4431 ± 0.0018	0.4811 ± 0.0007
	BPFormer-V4	2	Concatenate	32	24	0.4472 ± 0.0090	0.4857 ± 0.0075
NTU120 [93]	BPFormer-V1	1	Mean	256	24	0.2129 ± 0.0034	0.2192 ± 0.0030
	BPFormer-V2	1	Concatenate	32	24	0.3797 ± 0.0060	0.3748 ± 0.0050
	BPFormer-V3	2	Mean	256	24	0.2952 ± 0.0324	0.3060 ± 0.0345
	BPFormer-V4	2	Concatenate	32	24	**0.4560 ± 0.0033**	0.4559 ± 0.0014

TABLE VIII: Results of Experiment 2.2 with BPFormer. While concatenating embeddings, the intermediate embedding dimension of the Tubelet embedder is taken as 32, while during mean calculation, it is taken as 256. The segment length is taken as 24. For the 1-Channel representation, the kernel size was taken to be (5, 2, 2) while for the 2-Channel representation, the kernel size was taken to be (5, 3, 3).

Dataset	Model Type	Kernel	Balanced Accuracy	Top-3 Accuracy	Top-5 Accuracy	F1 Score
HRC [92]	BPTubeFormer-1	(1, 3, 3)	0.4797 ± 0.0065	0.7384	0.8453	0.5134 ± 0.0071
	BPTubeFormer-2	(2, 3, 3)	**0.4982 ± 0.0004**	0.7485	0.8508	0.5309 ± 0.0009
	BPTubeFormer-3	(4, 3, 3)	0.4873 ± 0.0048	0.7457	0.8495	0.5230 ± 0.0039
	BPTubeFormer-4	(5, 3, 3)	0.4768 ± 0.0017	0.7351	0.8456	0.5123 ± 0.0011
	BPTubeFormer-5	(6, 3, 3)	0.4747 ± 0.0045	0.7352	0.8457	0.5115 ± 0.0045
	BPTubeFormer-6	(8, 3, 3)	0.4686 ± 0.0022	0.7349	0.8458	0.5024 ± 0.0023
NTU120 [93]	BPTubeFormer-1	(1, 3, 3)	0.3957 ± 0.0083	0.6077	0.7000	0.3895 ± 0.0087
	BPTubeFormer-2	(2, 3, 3)	0.4229 ± 0.0119	0.6363	0.7257	0.4162 ± 0.0091
	BPTubeFormer-3	(4, 3, 3)	0.4468 ± 0.0053	0.6702	0.7586	0.4449 ± 0.0050
	BPTubeFormer-4	(5, 3, 3)	0.4217 ± 0.0128	0.6520	0.7439	0.4245 ± 0.0099
	BPTubeFormer-5	(6, 3, 3)	0.4617 ± 0.0130	0.6820	0.7680	0.4586 ± 0.0131
	BPTubeFormer-6	(8, 3, 3)	**0.4704 ± 0.0071**	0.7008	0.7865	0.4713 ± 0.0055
	BPTubeFormer-7	(12, 3, 3)	0.4681 ± 0.0037	0.7073	0.7931	0.4722 ± 0.0027

TABLE IX: Results of Experiment 2.3 with BPTubeFormer for HRC[92] and NTU120[93] datsets. The embedding dimension, C is 32. The segment length, f is 24 and the stride is same as the kernel size. Kernels are square matrix filters that are used to extract features from other square matrices (here, keypoints represented in square matrix form) with the help of convolution.

For the NTU120 dataset, however, the best performance was obtained with a kernel size of (8, 3, 3) while the performance decreased consistently as the kernel size was reduced, reaching the lowest with a kernel size of (1, 3, 3).

It is interesting to note that for the HRC dataset, the highest accuracy was obtained by taking the convolution of just two frames while for the NTU120 dataset, the convolution of 8 frames had to be taken to obtain the highest accuracy. Also, for both the datasets, the Top-3 Accuracy is around 70% and the Top-5 Accuracy is around 85% for the HRC dataset and 78% for the NTU120 dataset.

Experiments 2.4, 2.5 and 2.6: BPTubeFormer with the 2-Channel input representation and concatenated embeddings with HRC and NTU120 datasets with replication padding, different strides and different segment lengths for the tubelet embedder

In previous experiments, for padding the keypoints for reshaping them, we used the constant padding mode, (with zeros) as shown in Figure 13b. In Experiment 2.4, we use the 'replicate' padding mode provided by PyTorch, which pads the matrix elements with the outer elements, instead of zeros. We observe in Table XI that for the HRC dataset, replicate padding decreased the performance by 2% while for the NTU120 dataset the performance improved by a small margin, 0.8%. Since the increase in performance for the NTU120 dataset with the replicate padding mode is very small, this might indicate that the replicate padding mode cannot give no better results than padding with zeros, or that padding with non-zero values results in noise.

We also experimented with a smaller kernel and a smaller stride value which is not equal to the kernel size. Thus the same elements would be convolved over by the same kernel more than once. Also, this would give more number of embeddings. It is to be noted that this is in contrast to the original proposed method in ViViT [18], where the kernel size was same as the stride. We used a kernel of dimension (2,2,2) with two different stride values, (2,1,1) and (1,1,1). The results can be seen in Table X. For the HRC dataset, it was observed that the balanced accuracies decreased by a very small amount. For the NTU120 dataset, the accuracies decreased by a significant amount. This helps us to infer that small changes in the kernel size or the stride value does not bring any improvement in HAR tasks. It also proves that the original implementation of keeping the same value for both the kernel and stride provides the best performance.

Further experiments were carried out with a longer segment length. For this, we considered segments which are 60 frames long compared to 24 in the previous experiments. We observe in Table XII that the balanced accuracies remained similar for HRC dataset while for the NTU120 dataset, the accuracy

Dataset	Model Type	Kernel	Stride	Balanced Accuracy	F1 Score
HRC [92]	BPTubeFormer-2-V1	(2, 3, 3)	(2, 3, 3)	0.4982 ± 0.0004	0.5309 ± 0.0009
	BPTubeFormer-8-V1	(2, 2, 2)	(2, 1, 1)	0.4890 ± 0.0025	0.5246 ± 0.0035
	BPTubeFormer-8-V2	(2, 2, 2)	(1, 1, 1)	0.4917 ± 0.0022	0.5265 ± 0.0010
NTU120 [93]	BPTubeFormer-6-V1	(8, 3, 3)	(8, 3, 3)	0.4704 ± 0.0071	0.4713 ± 0.0055
	BPTubeFormer-8-V1	(8, 2, 2)	(8, 1, 1)	0.4469 ± 0.0052	0.4408 ± 0.0044
	BPTubeFormer-8-V2	(8, 2, 2)	(4, 1, 1)	0.3888 ± 0.0071	0.3823 ± 0.0053

TABLE X: Results of Experiment 2.5 with the BPTubeFormer (C = 32, f = 24) with different strides. The kernel size was taken to be (2, 2, 2) for the HRC dataset and (8,2,2) for the NTU120 dataset.

Dataset	Model Type	Padding Mode	Balanced Accuracy
HRC [92]	BPTubeFormer-2-V1	constant	0.4982 ± 0.0004
	BPTubeFormer-2-V2	replicate	0.4722 ± 0.0034
NTU120 [93]	BPTubeFormer-6-V1	constant	0.4704 ± 0.0071
	BPTubeFormer-6-V2	replicate	0.4784 ± 0.0044

TABLE XI: Results of Experiment 2.4 with the BPTubeFormer (C = 32, f = 24) with replicate padding.

dropped by 5%. This suggests that for both the datasets, longer segment lengths do not provide any benefits. However, the decrease in performance can also be because when the segment length is fixed as 60, it removes any trajectories which have shorter lengths.

Dataset	Model	Segment Length	Balanced Accuracy
HRC [92]	BPTubeFormer-2-V1	24	0.4982 ± 0.0004
	BPTubeFormer-2-V4	60	0.4899 ± 0.0023
NTU120 [93]	BPTubeFormer-6-V1	24	0.4704 ± 0.0071
	BPTubeFormer-6-V4	60	0.4201 ± 0.0165

TABLE XII: Results of Experiment 2.6 with the BPTubeFormer (C = 32) with a longer segment length f=60.

Best performing models for Experiment 2.3: In this section we analyze the performance of our models used in Experiment 2.3 with the HRC [92] as well as the NTU120 [93] dataset.

HRC dataset: We use a confusion matrix to analyze the classification performance of BPTubeFormer-2 which gave the best performance on the HRC dataset. This can be seen in Figure 20. It can be seen that all models perform better than random guessing ($\frac{1}{13} \approx 0.0769$). We observe that out of the 13 classes, 10 classes exhibit an accuracy above 50% (*Robbery* has nearly 50% accuracy). The least performance was seen for the class *Arson*, 22%. The best performance was seen for the class *Shoplifting*, similar to the finding in [7]. However, the accuracy for the class Vandalism saw great improvement, rising to 50% from 0.04% in the baseline model [7]. Accuracies of some other classes also improved, namely, Shooting, Fighting and Explosion. Accuracy for the class Arson got reduced by 3 times.

The high performance for *Shoplifting* videos is justifiable because such videos mostly contain actions which are seemingly normal, while other crime videos contain eccentric activities like *Fighting, Vandalism, Shooting* etc. Among all the 13 classes, *Shoplifting* is the only activity with such a property. This might be the reason why it has high accuracy.

It was interesting to note that the accuracy of *Vandalism* improved from 0.04 from Boekhoudt's work [7] to 0.50 in our work. It was mostly misclassified as either *Fighting* or *Road Accidents*. The misclassification as *Road Accidents* is understandable since most of such activities happen in the public space where there are roads and peoples' behaviour and walking could be similar in both the situations. However, it should also be noted that *Vandalism* is one of the activity class with very few video samples, 1388.

The most misclassified classes were Shooting, Robbery and Arson. Shooting was misclassified mostly as Fighting and Arrest while Arson was misclassified as Robbery and Fighting many times. While observing Shooting videos, we saw that many of them had people trying to fight with the shooter to stop him. Shooting videos also resembled police officers trying to arrest people since the attacked people were trying to catch the shooter's hands. The class Arson is also easily misclassified because at times, the videos become blocked with fire occluding the camera and interrupting the skeletal trajectories.

It was observed that the activity class Fighting was predicted many times while the actual classes were Arson, Assault, Explosion, Robbery, Shooting, Stealing, etc. This is because Fighting is an activity which can happen during any of the mentioned activities. People will fight the attackers when activities like Stealing, Assault or Robbery happen.

It was also interesting to note that Robbery was also misclassified as Shoplifting. While observing the videos belonging to the class Robbery, we noticed that many of them happened inside shops and closely resembled shoplifting. This might be an indication that our model performs well in understanding similar activities.

Similar to the activity class Arson, Explosion also demonstrated a low accuracy of 0.33. Explosion videos were classified as almost all of the other activity classes except Arson and Shoplifting. This is reasonable because in most of the explosion videos, the camera gets occluded with fire and smoke and interrupts the trajectories and hence affects the learning of the model. This could be the reason why it results in a low accuracy like that of Arson.

NTU120 dataset: The confusion matrix for NTU120

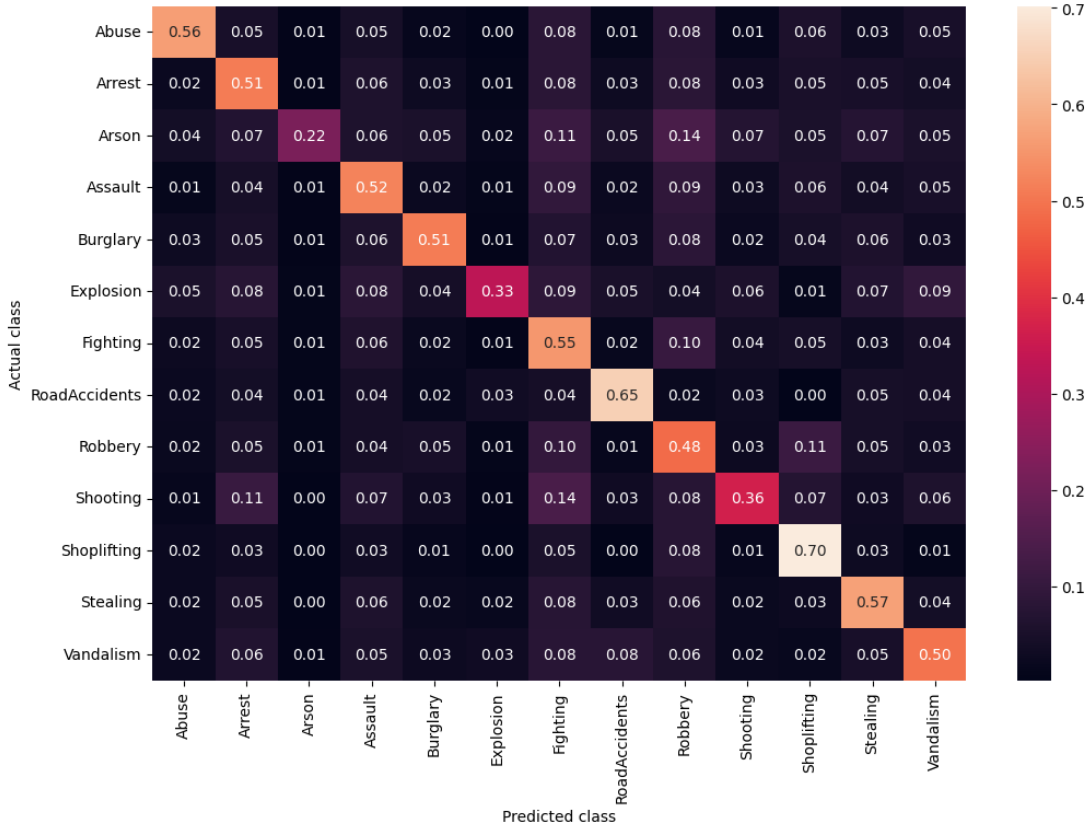


Fig. 20: Confusion matrix for BPTubeFormer-2 model using HRC dataset.

dataset on the BPTubeFormer-6-V2 model can be seen in Appendix:Figure 30. The best performance a class exhibited is 80%, which was obtained by A9, A22, A26, A27, A59, A97, A98 which corresponds to the activities standing up, cheer up, hopping, jump up, walking towards each other, arm circles and arm swings. The least performance exhibited was 10% obtained by the activities A72, A107 which corresponds to make victory sign and wield knife towards other person. Since there are many classes, a lot of misclassifications were also observed. This can be observed in Table XIII.

Observing the activity classes with high classification accuracy, one thing to be noted is that all of them are activities with big movements of the body. For example, standing up, hopping and jumping up are activities in which the torso or the other body parts exhibit large movements. While standing up, the upper body parts move and while jumping, all the body parts move at the same time. This is also the case while walking towards each other. Also, activities like arm circles and arm swings happen by moving the arm a lot. The transformer is able to learn a lot with the help of these large movements.

On the other hand, the most misclassified activities with less accuracy are making victory sign and wielding knife towards another person. One problem with these activities is that they closely resemble other similar activities like A71 (make ok sign) and A69 (thumb up). These are quick and subtle activities and also, at the same time, most of the body parts remain constant during all these activities. For example, while making victory sign and ok sign, only the wrists part change while all other parts including elbows, torso, knees and ankles remain motionless.

The most misclassified activity is A17 (taking off a shoe). It is misclassified as A16 (wearing a shoe). In both these activities, what changes is the order of movement of the leg. Another example just like A17 and A16 is A89 (put something into a bag) and A90 (take something out of a bag) where only the order changes. All other aspects remain the same during both the activities. Another example is A24 (kicking something) and A51 (kicking other person). More such activities can be seen in Table XIII.

A30 (typing on a keyboard) is an activity which is misclassified as A11 (reading) and A12 (writing). In all three activities, the movements are similar, where the person is sitting and the hands are moving.

Complexity of the models in terms of MACS and number of parameters. The comparison of the models based on the complexity is given in Table XIV. From the table, it can be seen that the complexity varies over a wide range of values. The most complex model in terms of MACS operations is the ST-Tran [7] while in terms of the number of trainable parameters, it is BPFormer-2. The least complex model in terms of MACS operations is BPFormer-4 while in terms of the number of trainable parameters, it is T-Tran-2 [7] and T-

Actual Class	Predicted Class	Misclassification Rate
A1 (drink water)	A3 (brushing teeth)	0.2
A11 (reading)	A12 (writing)	0.2
A17 (take off a shoe)	A16 (wear a shoe)	0.4
A24 (kicking something)	A51 (kicking other person)	0.2
A31 (pointing to something with finger)	A32 (taking a selfie)	0.2
A53 (pat on back of other person)	A57 (touch other person's pocket)	0.2
A58 (handshaking)	A56 (giving something to other person)	0.2
A72 (make victory sign)	A71 (make ok sign)	0.3
A73 (staple book)	A76 (cutting paper (using scissors))	0.2
A84 (play magic cube)	A74 (counting money)	0.2
A83 (ball up paper)	A82 (fold paper)	0.2
A89 (put something into a bag)	A90 (take something out of a bag)	0.3
A92 (move heavy objects)	A114 (carry something with other person)	0.2

TABLE XIII: Misclassification rates between actual and predicted classes for NTU120 dataset with the BPTubeFormer-6 model.

Model	MACS ($\times 10^9$)	# Params (Millions)
T-Tran-1 [7]	52.9	2.1
T-Tran-2 [7]	**18.6**	**0.53**
T-Tran-3 [7]	27.4	2.1
T-Tran-4 [7]	5.8	0.53
ST-Tran [7]	**613.5**	**9.5**
SBPT-Tran [7]	154	2.4
BPFormer-1	12.9	2.1
BPFormer-2	255.1	42.5
BPFormer-3	13.1	2.2
BPFormer-4	3.2	0.55
BPTubeFormer-2	**30.5**	**4.9**
BPTubeFormer-6	**3.9**	**0.7**

TABLE XIV: Comparison of the computational complexity of different models in terms of MACS and the number of trainable parameters. MACS is expressed in the order of 10^9 and the number of parameters in millions. The best performing models from the baseline and our proposed methods are shown in bold letters.

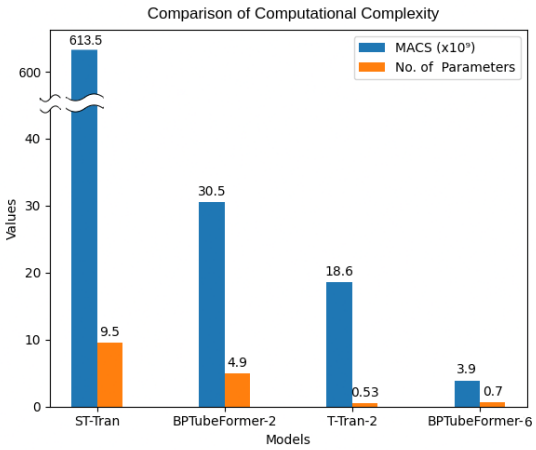


Fig. 21: Bar plot comparing the MACS and number of parameters for different models. Note that the blue bar for ST-Tran has been broken between 40 and 600. For HRC dataset, ST-Tran performed should be compared with BPTubeFormer-2 and for NTU120 dataset, T-Tran-2 should be compared with BPTubeFormer-6.

Tran-4 [7].

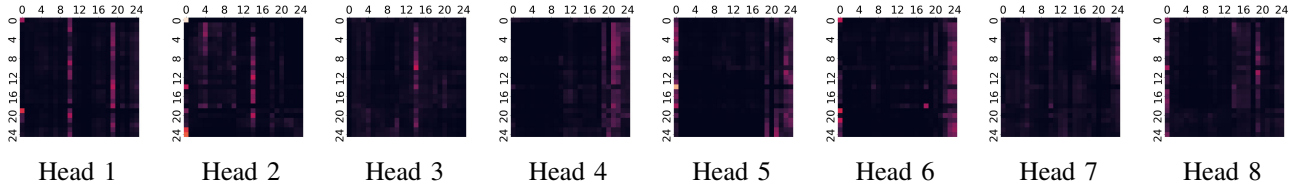
For the HRC dataset, the best baseline performance was given by ST-Tran [7]. Using the Tubelet Embedder, the best performance was given by BPTubeFormer-2. From the table, we can observe that the MACS operations came down from 613.5×10^9 to 30.5×10^9 (about 20 times less) and the number of parameters from 9.5 million to 4.9 million (about 2 times less). The high number of MACS operations for ST-Tran model is because it takes in a segment length of 60, thereby significantly increasing the number of embeddings. For the NTU120 dataset, the best baseline performance was obtained using T-Tran-2 [7] and using the Tubelet Embedder, it was using BPTubeFormer-6. In terms of complexity, the MACS operations came down from 18.6×10^9 to 3.9×10^9 (about 5 times less). However, it was observed that the number of parameters increased slightly from 0.53 million to 0.7 million.

The above results prove that the proposed models which use Tubelet embeddings can match the performance of the architectures proposed by Boekhoudt et.al. [7] with much lesser computational costs. This observation will be discussed in detail in Section VI.

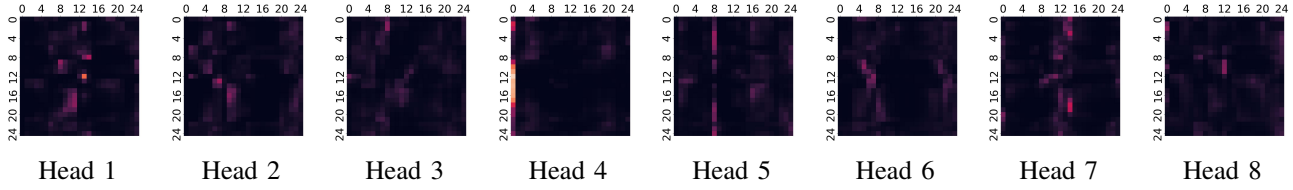
B. *Qualitative Evaluation*

Experiment 1.1. To perform qualitative evaluation, we take a trajectory segment from both the datasets and observe their attention weights of the last layer for all the 8 heads of the model. The attention is between different frames of the segment. This can be observed in Figure 22. For both the datasets, it can be seen that none of the heads focus on any specific frames. For the HRC dataset, we can see that many heads seem to focus on some of the frames, but however, the values are low and the heatmap is not bright enough. In the case of the NTU120 dataset, we do not see any specific pattern at all. There is no specific focus on any of the frames and the attentions are scattered around. This proves that the model could not learn enough from the locally and globally decomposed keypoints.

Experiment 2.3. To perform qualitative evaluation for this experiment, we take a trajectory segment from both the datasets and observe their attention weights of the last layer



(a) Segment of a video from *Abuse* category in HRC dataset.



(b) Segment of a video from *Arm circles* category in NTU120 dataset.

Fig. 22: Heatmaps for attention between frames within a segment for Experiment 1.1 with HRC and NTU120 datasets.

for all the 8 heads of the model. The attention is between the encoded embeddings for each body part. We also visualize the different body parts and observe their attention scores.

For the HRC dataset, we take a segment from the *Abuse* class in which a woman is hitting a child on a running bus. From Figure 23a, it can be observed that the first, fourth and fifth heads focus most on the second column, which corresponds to the Torso body part. The seventh head focuses on the fourth column, which is of the wrists. The second, third and eighth heads focus on the third column, which is of the elbows. The sixth head slightly focuses on many columns, mostly on those for the torso, elbow, wrists. The last two columns which corresponds to knees and ankle are focused on none of the heads. This makes sense since it can be observed in Figure 23e that in all the frames, the lower body part remains constant and only the upper body part is under action. Each blob in the frames correspond to different body parts - blue for torso, green for elbow, red for wrists, purple for knee and yellow for ankle. The radius of the blob indicates the attention score for that body part. The wrists and torso blobs are the biggest and the knee and ankle blobs are the smallest, as expected.

Figures 23b, 23d and 23c show the attention weights between different tubelet embeddings within the Tubelet transformer for the body parts torso, elbows and wrists. The segment length is 24 and the kernel size is (2,3,3). Hence, the number of tubelet embeddings is 12, with each of them representing two frames. This is why the attention matrix is in the shape 12×12. It can be seen that for the tubelet embeddings of the Wrists, Head 1 focuses on the frames 3 and 4. Heads 5 and 7 collectively focuses on frames 3 to 6. Head 3 completely focuses on frames 15 and 16. Head 4 focuses on frames 5 to 8.

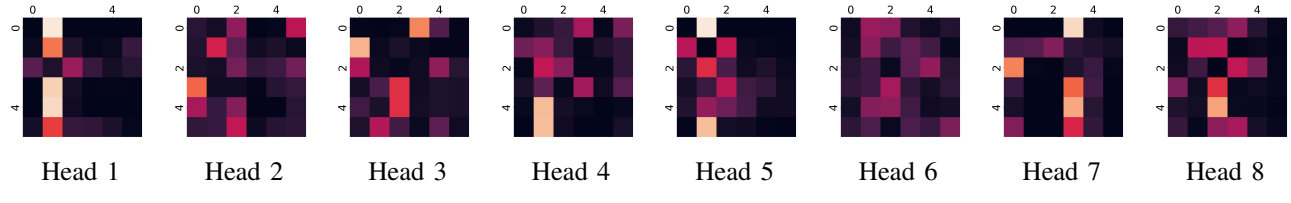
For the Torso, Heads 1 and 9 focus on frames 19 and 20. Also, Head 4 focuses on frames 11, 12, 21, 22. For the Elbows, Head 1 focuses on frames 3 and 4, Head 5 and 7 on frames 23 and 24, Head 6 on frames 13-16. It can be seen that wrists being the most active body part in this activity, it is attending to the frames 3 to 6 where it shows the most movement, as can be seen in Figure 23e.

For the NTU120 dataset, we take a segment from the *Arm circles* class in which a man circles his arms while standing in a place. Figure 24a shows that most of the focus is on the columns for Torso, elbows and wrists while heads 5,6 and 7 also focuses a bit on the knee. While focusing on torso, elbows and wrists is important, it is unsure as to why some heads also focus on the knees. From Figures 24b, 24c and 24d, we do not observe any specific pattern. Here, the segment length is 24 while the kernel size is (8,3,3). Thus, one tubelet embedding represents 8 frames. This is why the attention matrix is in the shape 3×3. Observing the attention matrix of the tubelet embeddings of Wrists, we see that Head 1, 4, 6, and 7 focuses on the frames 17-24. Heads 2 and 5 focus on frames 9 to 16. Head 3 and 8 focus on frames 1-8.

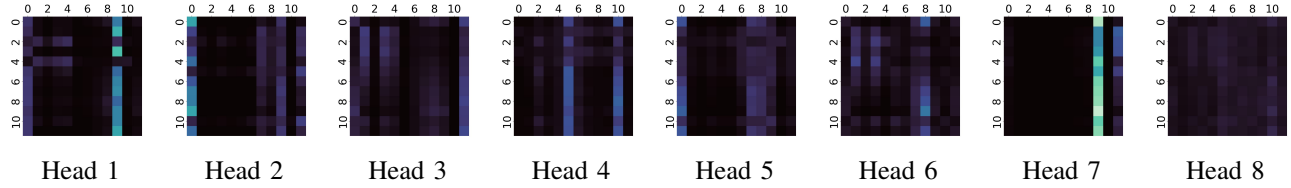
For the attention matrix related to Elbows, Head 1 and 3 focuses on the frames 1-8 and Heads 2, 4, 6 and 7 focus on the frames 17-24.

Contrasting to the HRC dataset, the attention focus is scattered in the NTU120 dataset since the activity occurs not within a timeframe of the segment but mostly throughout all frames. From Figure 24e, we see that the blobs for wrists and elbows are big as expected. The blob for torso is smaller since the torso remains almost constant while doing arm circles, like knees and elbows.

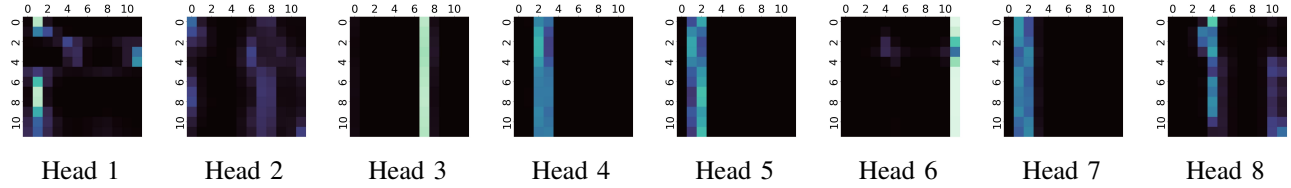
Silhouette plots. Figure 25 and Appendix:Figure 29 shows the silhouette plots for the BPTubeFormer models on the datasets HRC and NTU120 respectively. For the HRC dataset, we observe an average silhouette score of 0.0413. We observe generally low values for the silhouette coefficients for all the classes. Most embeddings in the Abuse, Road Accidents, Stealing and Vandalism classes are cohesive indicated by the long right tail and the height of each silhouettes in Figure 25. Separation value (indicated by the left tail of the silhouettes) is high for the classes Assault, Burglary, Explosion, Fighting, Robbery, Shooting and Shoplifting. This indicates that they have embeddings which are largely separated and overlaps



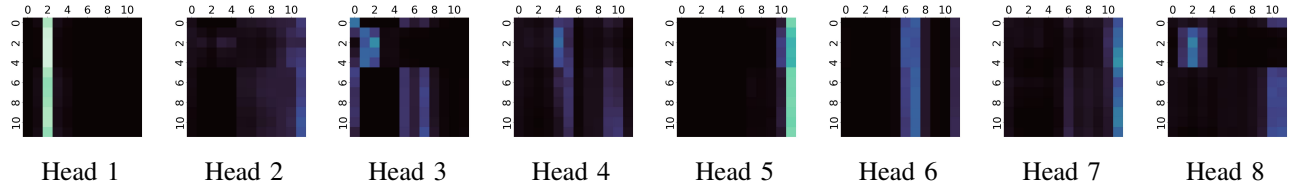
(a) Heatmaps for attention between body parts



(b) Heatmaps for attention between Tubelet embeddings of Torso



(c) Heatmaps for attention between Tubelet embeddings of Wrists



(d) Heatmaps for attention between Tubelet embeddings of Elbows

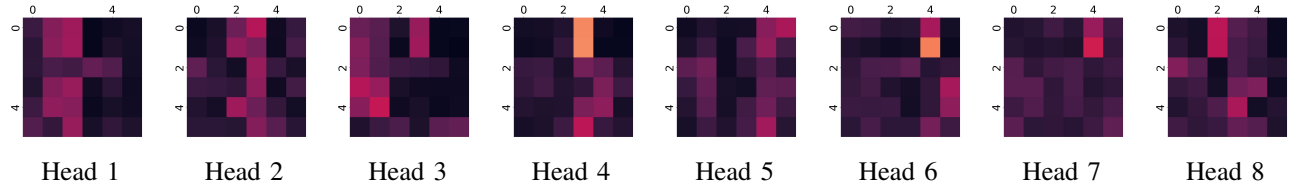
(e) Visualization for the attention scores for each body part

Fig. 23: **Experiment 2.3 for HRC dataset**. Visualization of self-attention in BPTubeFormer on a sequence of 24 frames. **(a, b, c, d) Attention heatmap**: Attention matrix of the last layer for heads 1 to 8. (a) represents the attention weight between the different body parts (index 0) corresponds to the class token. (b), (c) and (d) shows the attention weights between the different tubelet embeddings within the Tubelet transformer for the body parts Torso, wrists and elbows respectively. **(e) Skeleton body parts and attention**: The importance of each body part for the class token is illustrated. Each blob represents a body part - blue for torso, green for elbows, red for wrists, purple for knees, and yellow for ankles. The bigger the blob, the higher the attention score.

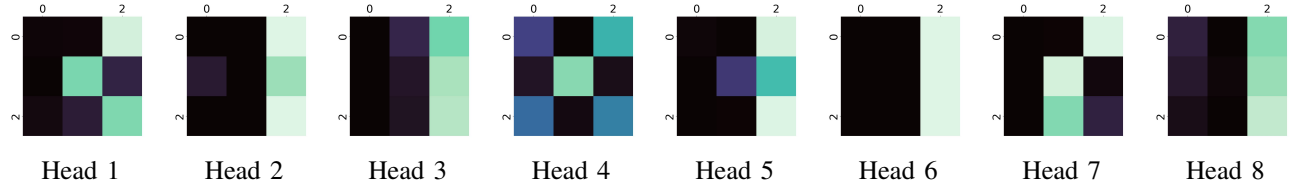
with other clusters. There are also some classes which have most of their embeddings separated. These are Assault, Burglary and Robbery.

For the NTU120 dataset, we observe an average silhouette score of 0.0436. The classes with good cohesion values are A18 (wear on glasses), A72 (make victory sign), A84 (play magic cube), A94 (throw up cap/hat), A113 (cheers and drink), A120 (finger-guessing game (playing rock-paper-scissors)). The classes with most of their embeddings separated are A2 (eat meal/snack), A9 (standing up (from sitting position)), A15 (take off jacket), A28 (make a phone call/answer phone), A36 (shake head), A51 (kicking other person), A60 (walking apart from each other), A83 (ball up paper), A86 (apply cream on hand back), A87 (put on bag), A100 (butt kicks (kick backward)), A112 (high-five), A114 (carry something with other person).

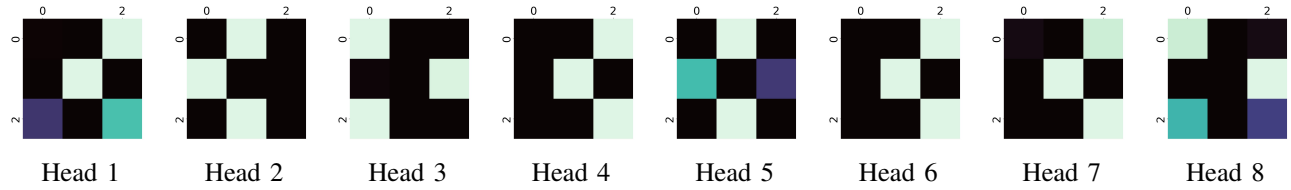
It is interesting to note that most of the activities with high separation values are those which are in Table XIII. This confirms that the model has confusion with distinguishing between activities which are very similar.



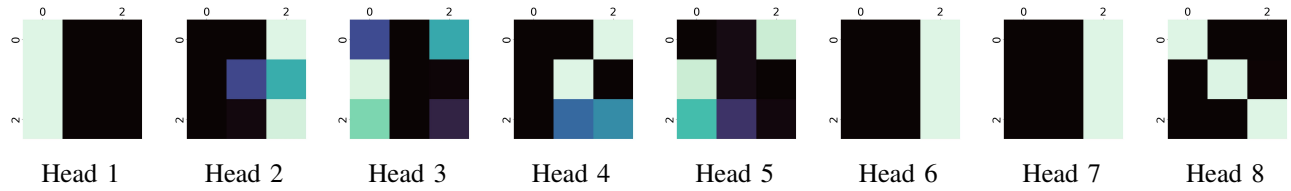
(a) Heatmaps for attention between body parts



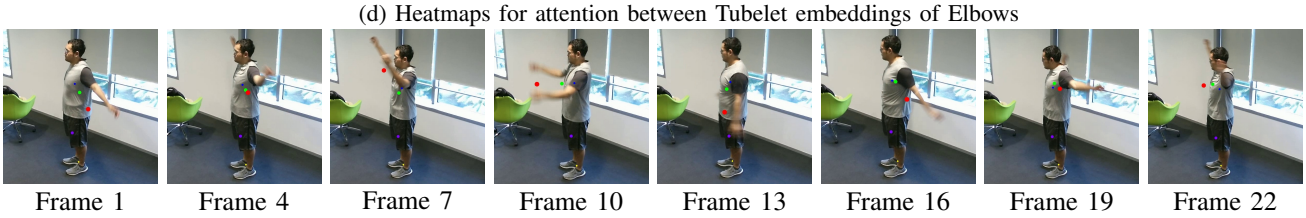
(b) Heatmaps for attention between Tubelet embeddings of Torso



(c) Heatmaps for attention between Tubelet embeddings of Wrists



(d) Heatmaps for attention between Tubelet embeddings of Elbows



(e) Heatmaps for attention between Tubelet embeddings of Elbows

Fig. 24: **Experiment 2.3 for NTU120 dataset**. Visualization of self-attention in BPTubeFormer on a sequence of 24 frames. **(a, b, c, d) Attention heatmap**: Attention matrix of the last layer for heads 1 to 8. (a) represents the attention weight between the different body parts (index 0) corresponds to the class token. (b), (c) and (d) shows the attention weights between the different tubelet embeddings within the Tubelet transformer for the body parts Torso, wrists and elbows respectively. **(e) Skeleton body parts and attention**: The importance of each body part for the class token is illustrated. Each blob represents a body part - blue for torso, green for elbows, red for wrists, purple for knees, and yellow for ankles. The bigger the blob, the higher the attention score.

T-SNE plots. We visualize the T-SNE plots for Experiment 2.3 in Figure 26. The plot for the HRC dataset can be seen in Figure 26a. We can observe that the embeddings are tightly packed and some embeddings are loosely scattered around the edges. Tensorboard displays embeddings with non-unique colours and hence, some colours are shared by two different classes. For example, red is shared by *Assault* and *Vandalism*. We can see that the embeddings for Robbery and Shoplifting are close to each other with some embeddings of Shooting placed in between them. Overall, most of the embeddings are well clustered.

The T-SNE plot for NTU120 dataset can be seen in Figure 26b. Again, Tensorboard supports colouring for only 50 different labels at maximum. Hence, we cannot uniquely colour all the 120 classes of the NTU120 dataset. In Figure 26c and Figure 26d, we illustrate the embeddings for the classes A59 and A107 where A59 is one of the classes with high accuracy and A107 is a class with a low accuracy. We can see that the embeddings for the class A59 is closely packed indicating good classification and the embeddings for the class

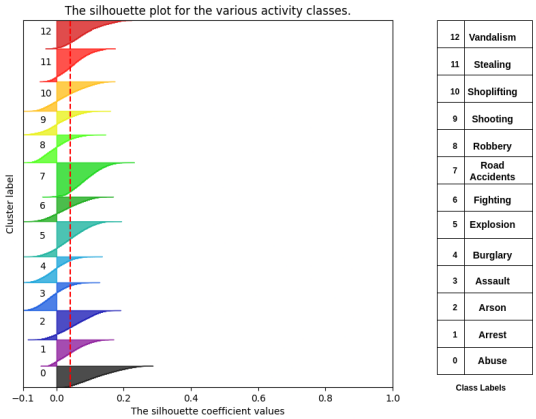
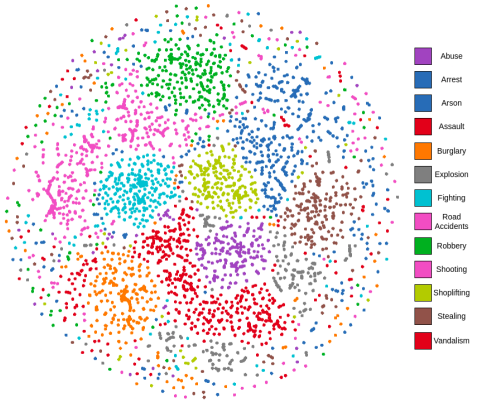
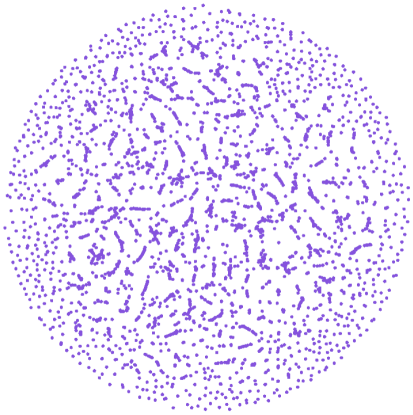


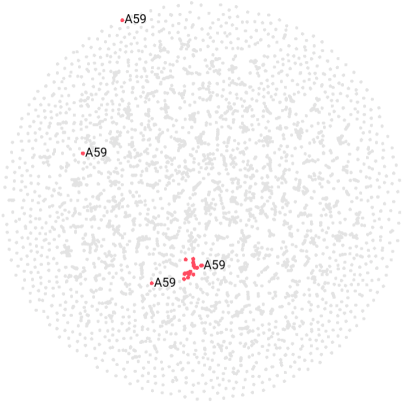
Fig. 25: Silhouette plot for the classifications of the BPTubeFormer model on the HRC dataset. The y axis labels indicate the activity class labels. The red line indicates the average silhouette score for all the classes.



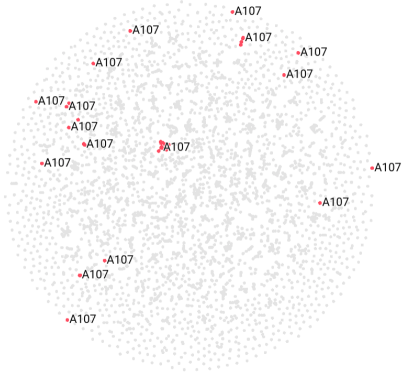
(a) T-SNE plot for HRC dataset.



(b) T-SNE plot for NTU120 dataset.



(c) Embeddings for A59 class within the T-SNE plot.



(d) Embeddings for A107 class within the T-SNE plot.

Fig. 26: T-SNE plots for the BPTubeFormer models on the HRC and NTU120 datasets. Tensorboard does not support colouring for more than 50 labels, hence for the NTU120 dataset, we show the embeddings for the A59 and A107 classes separately in (c) and (d).

A107 are scattered around indicating that there is variability in representing the samples of this class.

VI. Discussion

In this section, we discuss the overall results obtained while investigating the two aspects of this work.

Local and Global decomposition of keypoints: We investigated two different keypoint representations with respect to the Local and Global decomposition of keypoints. However, it was observed that this approach lowered the performance of the models compared to the baseline results. In the original work [9], the authors used Message-Passing Encoder-Decoder Recurrent Neural Network (MPED-RNN) which is a network consisting of two RNN branches dedicated to the global and local frame components. However, we processed the local and global components in a much simpler way. This might be suggesting that we need a more complex architecture to make use of the local and global components of the skeletons. The Global Single representation performed better compared to the Global Repeated representation. This could also mean that the global component is introducing some noise to the input representation. The local component might be more important for the model to learn the dependencies. This makes sense because the local component captures the actual movement of the body joints after factoring out the global component.

As seen in Figure 22, for all the heads, the attention between frames do not follow a pattern and is scattered with low values. This is the case for both the datasets. This suggests that the model cannot learn much with locally and globally decomposed keypoints.

Exploring Tubelet Embeddings: The major part of this work was concentrated on exploring how Tubelet embeddings could benefit the task of HAR. It was observed that Tubelet embeddings helped transformers to nearly match the performance of patch embeddings on both the datasets. However, it was ironic to note that the same architecture helped in matching the baseline performance for both the datasets. A simple TTubeFormer model which replaced the patch embedding layer in T-Tran model [7] with a Tubelet Embedder layer matched the performance of the baseline model on HRC [92] dataset, while it did not, in the case of the NTU120 dataset. For the BPFormer model, taking the concatenation of the embeddings worked the best for both the datasets. The 2-Channel representation gave a boost of 7% in performance for the NTU120 dataset over the 1-Channel representation. However, for the HRC dataset, both the 1-Channel and 2-Channel representations resulted in nearly similar accuracies. Hence, we can infer that we lose some information by taking the mean of embeddings when compared to concatenating the embeddings.

Another contrasting observation was that, using the BP-TubeFormer architecture, a kernel size of (2,3,3) gave the best performance in our work and the worst performance was obtained for the kernel size of (8,3,3). However, for the NTU120 dataset, the best performance was obtained for the kernel size of (8,3,3) and the worst for (1,3,3). Other modifications to the architecture like using replication padding mode, different stride than kernel size and a longer segment length did not result in any significant improvement in the performance. The fact that changing the stride value did not improve the performance establishes that keeping the same value for both the kernel and stride as in the original work [18] gives the best performance. This might be because having disjoint convolutions helps better in encoding the movement of different body parts.

While comparing the attention maps of the Tubelet transformer, it was observed that different heads attended to the tubelet embeddings corresponding to the frames which represented important movements in the activity. For example, the movement of wrists while Abusing. This observation was evident for the HRC dataset. For the NTU120 dataset, we observed that the heads did not focus on any particular tubelet embedding. We understand that this is because the NTU120 dataset is a lab made dataset and mostly contains the activity throughout the given segment length. This should be why the heads do not focus on any particular tubelet embedding. This proves that tubelet embeddings can detect movements of various body parts and helps in better interpretability of the models.

A major observation was related to the misclassifications related to very similar activities. It was evident from the silhouette plots of Figure 29 that the model found difficulty in distinguishing very similar activities in the NTU120 dataset. The model might be too simple to catch the differences between such very similar activities. This could be further investigated with the NTU60 dataset which is a subset of the NTU120 dataset and a more complex model architecture.

The two datasets, HRC [92] and NTU120 [93] did not work in the same way with Tubelet embeddings. One reason could be the nature of both the datasets. While the NTU120 dataset is lab made and skeletal data is obtained using sensors, the HRC dataset is obtained from real world surveillance footages with low quality videos and the skeletal data might not be accurate.

Thus it can be inferred that replacing the patch embeddings with tubelet embeddings will help in encoding the movement of different body parts and using a more complex architecture can significantly improve the performance.

Comparison with state of the art results: While comparing the performance of NTU120 dataset on the baseline architectures used in [7] (T-Tran-2) and the state of the art (SOTA) models, it was observed that the performance of T-Tran-2 was almost half that of the SOTA models. This could be because SOTA models used complex architectures. For example, ST-TR [8] used two-streams, one for Spatial data and the other for Temporal data. For both the streams, convolutional layers were also used. The two streams are trained end-to-end separately and are later fused together. FG-STFormer [41], used much more complex input features. They selected certain informative 'focal' joints and also used body parts. They used self-attention among them as well as cross-attention between the focal joints and the body parts. These might be the reason why the performance of T-Tran-2 did not

match that of the SOTA results. It is also to be noted that all the SOTA works used the cross-subject and cross-setup benchmarks, while we used the whole dataset and split them to train and test datasets based on the trajectory lengths.

Dataset	Model	Accuracy
HRC [92]	ST-Tran [7] (Baseline)	0.4926 ± 0.0043
	BPTubeFormer-2 (Ours)	**0.4982 ± 0.0004**
NTU120 [93]	ST-TR [8]	0.8190
	DSTA-Net [39]	0.8660
	IIP-Transformer [40]	0.8840
	FG-STFormer [41]	**0.8900**
	T-Tran-2 [7] (Baseline)	0.4830 ± 0.0030
	BPTubeFormer-6-V2 (Ours)	0.4784 ± 0.0044

TABLE XV: Comparison of state of the art results with ours.

Computational Complexity of the models: Comparing the computational complexity of the models, we can observe that the number of MACS operations increases a lot with increase in the number of embeddings. For example, with ST-Tran architecture, we take a segment of 60 frames and this exponentially increases the number of embeddings and hence, the number of MACS and parameters.

We explore the number of embeddings the architecture deals with, in the case of HRC dataset. Table XVI illustrates these details. The ST-Tran architecure has a Spatial Transformer which deals with 17 (keypoints) 32-dimensional embeddings and a Temporal Transformer Encoder which deals with 60 (frames) 544-dimensional embeddings. In the case of the BPTubeFormer-2 architecture, it has a Tubelet Transformer which deals with 12 (tubelet embeddings) 32-dimensional embeddings and a Body Part transformer which deals with 6 (body parts) 384-dimensional embeddings. In the case of BPTubeFormer-6, the Body Part Transformer deals with 6 (body parts) 96-dimensional embeddings.

The less computational complexity of the architectures which use Tubelet Embeddings is due to the fact that Tubelet embedders introduce 3D convolutional learning and 3D convolutions can extract spatial and temporal dependencies at the same time, with low computational cost. This avoids the need to have separate Spatial and Temporal encoders with transformers. Convolutional learning also has the advantage that it introduces inductive bias to the architecture which transformers typically lack. This asserts the fact Tubelet Embedders are a promising method to improve the performance of Transformers used in HAR tasks.

A. *Limitations*

Here we discuss some of the limitations with respect to this work. One of the limitation is that we observed that the validation accuracy nears 100% towards the end of training. This could be a reason why we do not have a major improvement in the test accuracy. As a remedy, we also used weight decay regularization to prevent overfitting. However, it was observed that the performance did not improve. Hence, it was decided to not use weight decay techniques.

Model	Encoder	No. of Embeddings	No.of Elements
T-Tran-2 [7]	Transformer	34 × 128-d	4352
ST-Tran [7]	Spatial	17 × 32-d	544
	Temporal	60 × 544-d	32640
BPTubeFormer-2	Tubelet	12 × 32-d	384
	Body Part	6 × 384-d	2304
BPTubeFormer-6	Tubelet	3 × 32-d	96
	Body Part	6 × 96-d	576

TABLE XVI: Comparison of the number of embeddings dealt by different modules of the different architectures.

Another limitation is that in the HRC dataset, we do not have videos from the 'Normal' category which do not have any anomalous events. Hence, if the model sees trajectories belonging to the Shoplifting class, it's more than likely to predict it as a shoplifting activity since those activities resemble normal activities and the shoplifters does this crime without anyone noticing it.

We also found that in the HRC dataset, classes like Arson, Explosion exhibited low accuracies. As mentioned in [7], these activities occlude the camera with smoke and dust and thus skeletal trajectories are not retrieved which makes activity recognition difficult.

In the original paper, Liu et.al. [84] proposed two evaluation benchmarks for the NTU120 dataset, cross-subject and cross-setup, to standardize the research using this dataset. However, these could not be used in this research work. Using them could have standardised the comparison with the state of the art results.

B. *Future Work*

There is a lot of scope for improvement in this work:

1) The models could be evaluated on other datasets like Kinetics-Skeleton [103] and Human 3.6M [104].
2) The models could be checked for overfitting and the models could be regularised using regularization techniques other than weight decaying.
3) NTU provides 3D keypoints for the same dataset. The code already supports 3D keypoints and these could be easily integrated. This couldn't be implemented in this work due to memory limitations and time constraints.
4) A major scope for improvement is to use a hybrid architecture as used in ViT [70]. For this, a ResNet CNN architecture could be used to extract embedding features from the videos and these could be used along with the patch/tubelet embeddings and the class token.
5) For the experiment regarding local and global decomposition of keypoints, the framework could be extended by making it a two stream network in which one stream deals with the global components, while the other deals with the local components.
6) The NTU120 dataset comes with other modalities of data like depth map sequences and infrared data. These data could be used to retrieve more information.

7) The authors of NTU120 work [93] proposed two evaluation criterias for the NTU120 dataset, cross-subject and cross-setup. In the future works, these could be used for direct comparison with state of the art works.

VII. Conclusion

In this work, we explored two different aspects of skeletal activity recognition using transformers with two different datasets. In the first we explored whether using an input representation with the locally and globally decomposed keypoints could better represent the skeletal data. We observed that using locally and globally decomposed keypoints, the models performed badly compared to the baseline models. This indicates that with this architecture, they do not help in extracting enough information for HAR tasks. We also investigated tubelet embeddings with three different architectures. We observe that they matched the existing baseline performance with much less computational cost. We demonstrated that Tubelet Embedders could encode the movement of different body parts with better interpretability which is relevant for HAR tasks. For the NTU120 dataset, many similar activites were misclassified with each other, for example, reading and writing, take off shoe and wear shoe, etc. This suggests that incorporating tubelet embeddings to transformers could benefit HAR tasks and the performance could be further improved with more complex model architectures.

References

[1] Elly Cosgrove. *One billion surveillance cameras will be watching around the world in 2021, a new study says*. Dec. 2019. URL: https://www.cnbc.com/2019/12/06/one-billion-surveillance-cameras-will-be-watching-globally-in-2021.html.

[2] *CCTV camera market size, trends, growth and overview – 2030: MRFR*. Feb. 2020. URL: https://www.marketresearchfuture.com/reports/cctv-camera-market-8160.

[3] *Video surveillance storage market segmentation by enterprise type (large enterprise, and Small amp; Medium Enterprise); by end-user (residential, commercial, defense, industrial, and others); by deployment type (on-premise, and Cloud)-Global Demand Analysis amp; Opportunity Outlook 2031*. Oct. 2022. URL: https://www.kennethresearch.com/report-details/video-surveillance-storage-market/10154361.

[4] John C McCallum. *Historical cost of computer memory and storage*. 2022. URL: https://ourworldindata.org/grapher/historical-cost-of-computer-memory-and-storage?country=~OWID_WRL.

[5] D. Berleant. *Trends in the cost of computing*. June 2022. URL: https://aiimpacts.org/trends-in-the-cost-of-computing/.

[6] *Digital around the world - datareportal – global digital insights*. URL: https://datareportal.com/global-digital-overview.

[7] Kayleigh Boekhoudt. *Learning how to represent human-related crimes when using transformers for action recognition*. 2021.

[8] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. "Skeleton-based action recognition via spatial and temporal transformer networks". In: *Computer Vision and Image Understanding* 208-209 (July 2021), p. 103219. DOI: 10.1016/j.cviu.2021.103219. URL: https://doi.org/10.1016%2Fj.cviu.2021.103219.

[9] Romero Morais et al. *Learning Regularity in Skeleton Trajectories for Anomaly Detection in Videos*. 2019. DOI: 10.48550/ARXIV.1903.03295. URL: https://arxiv.org/abs/1903.03295.

[10] Chenyang Li et al. "Skeleton-based gesture recognition using several fully connected layers with path signature features and temporal transformer module". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8585–8593.

[11] Pichao Wang et al. "Action Recognition From Depth Maps Using Deep Convolutional Neural Networks". In: *IEEE Transactions on Human-Machine Systems* 46.4 (2016), pp. 498–509. DOI: 10.1109/THMS.2015.2504550.

[12] Aparna Akula, Anuj K. Shah, and Ripul Ghosh. "Deep learning approach for human action recognition in infrared images". In: *Cognitive Systems Research* 50 (2018), pp. 146–154. ISSN: 1389-0417. DOI: https://doi.org/10.1016/j.cogsys.2018.04.002. URL: https://www.sciencedirect.com/science/article/pii/S1389041717302206.

[13] Dipanwita Thakur, Suparna Biswas, and Arindam Pal. "Human Activity Recognition Systems Based on Audio-Video Data Using Machine Learning and Deep Learning". In: *Internet of Things Based Smart Healthcare: Intelligent and Secure Solutions Applying Machine Learning Techniques*. Ed. by Suparna Biswas et al. Singapore: Springer Nature Singapore, 2022, pp. 151–175. ISBN: 978-981-19-1408-9. DOI: 10.1007/978-981-19-1408-9_7. URL: https://doi.org/10.1007/978-981-19-1408-9_7.

[14] Abassin Sourou Fangbemi et al. "Efficient Human Action Recognition Interface for Augmented and Virtual Reality Applications Based on Binary Descriptor". In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis and Patrick Bourdot. Cham: Springer International Publishing, 2018, pp. 252–260. ISBN: 978-3-319-95270-3.

[15] Gheorghe Sebestyen, Ionut Stoica, and Anca Hangan. "Human activity recognition and monitoring for elderly people". In: *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*. 2016, pp. 341–347. DOI: 10.1109/ICCP.2016.7737171.

[16] Ivan Laptev. "On Space-Time Interest Points". In: *International Journal of Computer Vision* 64.2 (Sept. 2005), pp. 107–123. ISSN: 1573-1405. DOI: 10.1007/

s11263-005-1838-7. URL: https://doi.org/10.1007/s11263-005-1838-7.

[17] Dr Mohana and Mahanthesh U M. "Human Action Recognition using STIP Techniques". In: *International Journal of Innovative Technology and Exploring Engineering* 9 (May 2020). DOI: 10.35940/ijitee.G5482.059720.

[18] Anurag Arnab et al. *ViViT: A Video Vision Transformer*. 2021. DOI: 10.48550/ARXIV.2103.15691. URL: https://arxiv.org/abs/2103.15691.

[19] Ce Zheng et al. *3D Human Pose Estimation with Spatial and Temporal Transformers*. 2021. DOI: 10.48550/ARXIV.2103.10455. URL: https://arxiv.org/abs/2103.10455.

[20] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[21] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[22] Jianpeng Cheng, Li Dong, and Mirella Lapata. *Long Short-Term Memory-Networks for Machine Reading*. 2016. DOI: 10.48550/ARXIV.1601.06733. URL: https://arxiv.org/abs/1601.06733.

[23] Yi-Xiang Zhang et al. "RGB+2D skeleton: local handcrafted and 3D convolution feature coding for action recognition". In: *Signal, Image and Video Processing* 15.7 (Oct. 2021), pp. 1379–1386. DOI: 10.1007/s11760-021-01868-8. URL: https://doi.org/10.1007/s11760-021-01868-8.

[24] Onur Temuroglu et al. "Occlusion-Aware Skeleton Trajectory Representation for Abnormal Behavior Detection". In: Apr. 2020, pp. 108–121. ISBN: 978-981-15-4817-8. DOI: 10.1007/978-981-15-4818-5_9.

[25] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. DOI: 10.48550/ARXIV.1406.2199. URL: https://arxiv.org/abs/1406.2199.

[26] Amin Zare, Hamid Abrishami Moghaddam, and Arash Sharifi. "Video spatiotemporal mapping for human action recognition by convolutional neural network". In: *Pattern Analysis and Applications* 23.1 (Feb. 2020), pp. 265–279. ISSN: 1433-755X. DOI: 10.1007/s10044-019-00788-1. URL: https://doi.org/10.1007/s10044-019-00788-1.

[27] Inwoong Lee et al. "Ensemble Deep Learning for Skeleton-Based Action Recognition Using Temporal Sliding LSTM Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1012–1020. DOI: 10.1109/ICCV.2017.115.

[28] Ge Pan, YongHong Song, and ShengHua Wei. "Combining Pose and Trajectory for Skeleton Based Action Recognition using Two-Stream RNN". In: *2019 Chinese Automation Congress (CAC)*. 2019, pp. 4375–4380. DOI: 10.1109/CAC48633.2019.8997171.

[29] Romero Morais et al. *Learning Regularity in Skeleton Trajectories for Anomaly Detection in Videos*. 2019. DOI: 10.48550/ARXIV.1903.03295. URL: https://arxiv.org/abs/1903.03295.

[30] Zewei Ding et al. "Investigation of different skeleton features for CNN-based 3D action recognition". In: *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2017, pp. 617–622. DOI: 10.1109/ICMEW.2017.8026286.

[31] Yangyang Xu et al. "Ensemble One-Dimensional Convolution Neural Networks for Skeleton-Based Action Recognition". In: *IEEE Signal Processing Letters* 25.7 (2018), pp. 1044–1048. DOI: 10.1109/LSP.2018.2841649.

[32] Pichao Wang et al. "Action recognition based on joint trajectory maps with convolutional neural networks". In: *Knowledge-Based Systems* 158 (2018), pp. 43–53. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2018.05.029. URL: https://www.sciencedirect.com/science/article/pii/S0950705118302582.

[33] Sijie Yan, Yuanjun Xiong, and Dahua Lin. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*. 2018. DOI: 10.48550/ARXIV.1801.07455. URL: https://arxiv.org/abs/1801.07455.

[34] Lei Shi et al. "Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12018–12027. DOI: 10.1109/CVPR.2019.01230.

[35] Yuxin Chen et al. *Channel-wise Topology Refinement Graph Convolution for Skeleton-Based Action Recognition*. 2021. DOI: 10.48550/ARXIV.2107.12213. URL: https://arxiv.org/abs/2107.12213.

[36] Bin Ren et al. *A Survey on 3D Skeleton-Based Action Recognition Using Learning Method*. 2020. DOI: 10.48550/ARXIV.2002.05907. URL: https://arxiv.org/abs/2002.05907.

[37] Chuankun Li et al. "Joint Distance Maps Based Action Recognition With Convolutional Neural Networks". In: *IEEE Signal Processing Letters* 24.5 (May 2017), pp. 624–628. DOI: 10.1109/lsp.2017.2678539. URL: https://doi.org/10.11092Flsp.2017.2678539.

[38] Salman Khan et al. "Transformers in Vision: A Survey". In: (2021). DOI: 10.48550/ARXIV.2101.01169. URL: https://arxiv.org/abs/2101.01169.

[39] Lei Shi et al. *Decoupled Spatial-Temporal Attention Network for Skeleton-Based Action Recognition*. 2020. DOI: 10.48550/ARXIV.2007.03263. URL: https://arxiv.org/abs/2007.03263.

[40] Qingtian Wang et al. *IIP-Transformer: Intra-Inter-Part Transformer for Skeleton-Based Action Recognition*. 2021. DOI: 10.48550/ARXIV.2110.13385. URL: https://arxiv.org/abs/2110.13385.

[41] Zhimin Gao et al. *Focal and Global Spatial-Temporal Transformer for Skeleton-based Action Recognition*.

2022. DOI: 10.48550/ARXIV.2210.02693. URL: https://arxiv.org/abs/2210.02693.

[42] Yuhan Zhang et al. "STST: Spatial-Temporal Specialized Transformer for Skeleton-Based Action Recognition". In: MM '21. Virtual Event, China: Association for Computing Machinery, 2021, pp. 3229–3237. ISBN: 9781450386517. DOI: 10.1145/3474085.3475473. URL: https://doi.org/10.1145/3474085.3475473.

[43] Pengfei Zhang et al. *View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition*. 2018. DOI: 10.48550/ARXIV.1804.07453. URL: https://arxiv.org/abs/1804.07453.

[44] Chenyang Si et al. *An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition*. 2019. DOI: 10.48550/ARXIV.1902.09130. URL: https://arxiv.org/abs/1902.09130.

[45] Lei Shi et al. "Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12018–12027. DOI: 10.1109/CVPR.2019.01230.

[46] Lei Shi et al. "Skeleton-Based Action Recognition With Directed Graph Neural Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7904–7913. DOI: 10.1109/CVPR.2019.00810.

[47] Matthew Korban and Xin Li. "DDGCN: A Dynamic Directed Graph Convolutional Network for Action Recognition". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 761–776. ISBN: 978-3-030-58565-5.

[48] Sijie Yan, Yuanjun Xiong, and Dahua Lin. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*. 2018. DOI: 10.48550/ARXIV.1801.07455. URL: https://arxiv.org/abs/1801.07455.

[49] Maosen Li et al. *Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition*. 2019. DOI: 10.48550/ARXIV.1904.12659. URL: https://arxiv.org/abs/1904.12659.

[50] Pengfei Zhang et al. *Semantics-Guided Neural Networks for Efficient Skeleton-Based Human Action Recognition*. 2019. DOI: 10.48550/ARXIV.1904.01189. URL: https://arxiv.org/abs/1904.01189.

[51] Yi-Fan Song et al. "Richly Activated Graph Convolutional Network for Robust Skeleton-Based Action Recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.5 (May 2021), pp. 1915–1925. DOI: 10.1109/tcsvt.2020.3015051. URL: https://doi.org/10.1109%2Ftcsvt.2020.3015051.

[52] Lei Shi et al. "Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12018–12027. DOI: 10.1109/CVPR.2019.01230.

[53] Hao Yang et al. *Feedback Graph Convolutional Network for Skeleton-based Action Recognition*. 2020. DOI: 10.48550/ARXIV.2003.07564. URL: https://arxiv.org/abs/2003.07564.

[54] Ke Cheng et al. "Skeleton-Based Action Recognition With Shift Graph Convolutional Network". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 180–189. DOI: 10.1109/CVPR42600.2020.00026.

[55] Ke Cheng et al. "Decoupling GCN with DropGraph Module for Skeleton-Based Action Recognition". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 536–553. ISBN: 978-3-030-58586-0.

[56] Ziyu Liu et al. *Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition*. 2020. DOI: 10.48550/ARXIV.2003.14111. URL: https://arxiv.org/abs/2003.14111.

[57] Zhan Chen et al. *Multi-Scale Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recognition*. 2022. DOI: 10.48550/ARXIV.2206.13028. URL: https://arxiv.org/abs/2206.13028.

[58] Yi-Fan Song et al. "Constructing Stronger and Faster Baselines for Skeleton-based Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–1. DOI: 10.1109/TPAMI.2022.3157033.

[59] Yuxin Chen et al. *Channel-wise Topology Refinement Graph Convolution for Skeleton-Based Action Recognition*. 2021. DOI: 10.48550/ARXIV.2107.12213. URL: https://arxiv.org/abs/2107.12213.

[60] Lipeng Ke, Kuan-Chuan Peng, and Siwei Lyu. *Towards To-a-T Spatio-Temporal Focus for Skeleton-Based Action Recognition*. 2022. DOI: 10.48550/ARXIV.2202.02314. URL: https://arxiv.org/abs/2202.02314.

[61] Kirill Gavrilyuk et al. *Actor-Transformers for Group Activity Recognition*. 2020. DOI: 10.48550/ARXIV.2003.12737. URL: https://arxiv.org/abs/2003.12737.

[62] Shuaicheng Li et al. *GroupFormer: Group Activity Recognition with Clustered Spatial-Temporal Transformer*. 2021. DOI: 10.48550/ARXIV.2108.12630. URL: https://arxiv.org/abs/2108.12630.

[63] Bing Li et al. "Two-Stream Convolution Augmented Transformer for Human Activity Recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.1 (May 2021), pp. 286–293. DOI: 10.1609/aaai.v35i1.16103. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16103.

[64] Annalisa Franco, Antonio Magnani, and Dario Maio. "A multimodal approach for human activity recognition based on skeleton and RGB data". In: *Pattern Recognition Letters* 131 (2020), pp. 293–299. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2020.

01.010. URL: https://www.sciencedirect.com/science/article/pii/S0167865520300106.

[65] Alessandro Manzi, Paolo Dario, and Filippo Cavallo. "A Human Activity Recognition System Based on Dynamic Clustering of Skeleton Data". In: *Sensors* 17.5 (May 2017), p. 1100. ISSN: 1424-8220. DOI: 10.3390/s17051100. URL: http://dx.doi.org/10.3390/s17051100.

[66] Enea Cippitelli et al. "A Human Activity Recognition System Using Skeleton Data from RGBD Sensors". In: *Computational Intelligence and Neuroscience* 2016 (Mar. 2016), p. 4351435. ISSN: 1687-5265. DOI: 10.1155/2016/4351435. URL: https://doi.org/10.1155/2016/4351435.

[67] Jiaxu Zhang et al. "Zoom Transformer for Skeleton-based Group Activity Recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2022), pp. 1–1. DOI: 10.1109/TCSVT.2022.3193574.

[68] Xiaolin Zhai et al. "Spatial Temporal Network for Image and Skeleton Based Group Activity Recognition". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Dec. 2022, pp. 20–38.

[69] Ahmed Snoun, Tahani Bouchrika, and Olfa Jemai. "Deep-learning-based human activity recognition for Alzheimer's patients' daily life activities assistance". In: *Neural Computing and Applications* (Oct. 2022). ISSN: 1433-3058. DOI: 10.1007/s00521-022-07883-1. URL: https://doi.org/10.1007/s00521-022-07883-1.

[70] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: 10.48550/ARXIV.2010.11929. URL: https://arxiv.org/abs/2010.11929.

[71] Rajkumar Saini et al. "Trajectory Classification Using Feature Selection by Genetic Algorithm". In: *Proceedings of 3rd International Conference on Computer Vision and Image Processing*. Ed. by Bidyut B. Chaudhuri et al. Singapore: Springer Singapore, 2020, pp. 377–388. ISBN: 978-981-32-9291-8.

[72] Alina-Daniela Matei, Estefania Talavera, and Maya Aghaei. *Crime scene classification from skeletal trajectory analysis in surveillance settings*. 2022. DOI: 10.48550/ARXIV.2207.01687. URL: https://arxiv.org/abs/2207.01687.

[73] Pranay Gupta et al. "Quo Vadis, Skeleton Action Recognition?" In: *International Journal of Computer Vision* 129.7 (July 2021), pp. 2097–2112. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01470-y. URL: https://doi.org/10.1007/s11263-021-01470-y.

[74] Julieta Martinez et al. *A simple yet effective baseline for 3d human pose estimation*. 2017. DOI: 10.48550/ARXIV.1705.03098. URL: https://arxiv.org/abs/1705.03098.

[75] Xingyi Zhou et al. *Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach*. 2017. DOI: 10.48550/ARXIV.1704.02447. URL: https://arxiv.org/abs/1704.02447.

[76] Rishabh Dabral et al. *Learning 3D Human Pose from Structure and Motion*. 2017. DOI: 10.48550/ARXIV.1711.09250. URL: https://arxiv.org/abs/1711.09250.

[77] Dario Pavllo et al. *3D human pose estimation in video with temporal convolutions and semi-supervised training*. 2018. DOI: 10.48550/ARXIV.1811.11742. URL: https://arxiv.org/abs/1811.11742.

[78] Yu Cheng et al. "Occlusion-Aware Networks for 3D Human Pose Estimation in Video". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[79] Yujun Cai et al. "Exploiting Spatial-Temporal Relationships for 3D Pose Estimation via Graph Convolutional Networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[80] Tianshu Zhang, Buzhen Huang, and Yangang Wang. "Object-Occluded Human Shape and Pose Estimation From a Single Color Image". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[81] Jingbo Wang et al. *Motion Guided 3D Pose Estimation from Videos*. 2020. DOI: 10.48550/ARXIV.2004.13985. URL: https://arxiv.org/abs/2004.13985.

[82] Mir Rayat Imtiaz Hossain and James J. Little. "Exploiting Temporal Information for 3D Human Pose Estimation". In: *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 69–86. DOI: 10.1007/978-3-030-01249-6_5. URL: https://doi.org/10.1007%2F978-3-030-01249-6_5.

[83] Zhi Li et al. "On Boosting Single-Frame 3D Human Pose Estimation via Monocular Videos". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[84] Ruixu Liu et al. "Attention Mechanism Exploits Temporal Contexts: Real-Time 3D Human Pose Reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[85] Tianlang Chen et al. *Anatomy-aware 3D Human Pose Estimation with Bone-based Pose Decomposition*. 2020. DOI: 10.48550/ARXIV.2002.10322. URL: https://arxiv.org/abs/2002.10322.

[86] Yufei Xu et al. *ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation*. 2022. DOI: 10.48550/ARXIV.2204.12484. URL: https://arxiv.org/abs/2204.12484.

[87] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[88] Haoqi Fan et al. *Multiscale Vision Transformers*. 2021. DOI: 10.48550/ARXIV.2104.11227. URL: https://arxiv.org/abs/2104.11227.

[89] Yanghao Li et al. *MViTv2: Improved Multiscale Vision Transformers for Classification and Detection*. 2021.

DOI: 10.48550/ARXIV.2112.01526. URL: https://arxiv.org/abs/2112.01526.

[90] Kan Wu et al. *Rethinking and Improving Relative Position Encoding for Vision Transformer*. 2021. DOI: 10.48550/ARXIV.2107.14222. URL: https://arxiv.org/abs/2107.14222.

[91] Yuxiang Yang et al. *APT-36K: A Large-scale Benchmark for Animal Pose Estimation and Tracking*. 2022. DOI: 10.48550/ARXIV.2206.05683. URL: https://arxiv.org/abs/2206.05683.

[92] Kayleigh Boekhoudt et al. *HR-Crime: Human-Related Anomaly Detection in Surveillance Videos*. 2021. DOI: 10.34894/IRRDJE. URL: https://dataverse.nl/citation?persistentId=doi:10.34894/IRRDJE.

[93] Jun Liu et al. "NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (2020), pp. 2684–2701.

[94] Kayleigh Boekhoudt et al. *HR-Crime: Human-Related Anomaly Detection in Surveillance Videos*. Version V1. 2021. DOI: 10.34894/IRRDJE. URL: https://doi.org/10.34894/IRRDJE.

[95] Waqas Sultani, Chen Chen, and Mubarak Shah. *Real-world Anomaly Detection in Surveillance Videos*. 2018. DOI: 10.48550/ARXIV.1801.04264. URL: https://arxiv.org/abs/1801.04264.

[96] Zhanchao Huang and Jianlin Wang. *DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection*. 2019. DOI: 10.48550/ARXIV.1903.08589. URL: https://arxiv.org/abs/1903.08589.

[97] Hao-Shu Fang et al. *RMPE: Regional Multi-person Pose Estimation*. 2016. DOI: 10.48550/ARXIV.1612.00137. URL: https://arxiv.org/abs/1612.00137.

[98] Yuliang Xiu et al. "Pose Flow: Efficient Online Pose Tracking". In: (2018). DOI: 10.48550/ARXIV.1802.00977. URL: https://arxiv.org/abs/1802.00977.

[99] Amir Shahroudy et al. *NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis*. 2016. DOI: 10.48550/ARXIV.1604.02808. URL: https://arxiv.org/abs/1604.02808.

[100] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[101] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[102] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[103] Sijie Yan, Yuanjun Xiong, and Dahua Lin. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*. 2018. DOI: 10.48550/ARXIV.1801.07455. URL: https://arxiv.org/abs/1801.07455.

[104] Catalin Ionescu et al. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (July 2014), pp. 1325–1339.

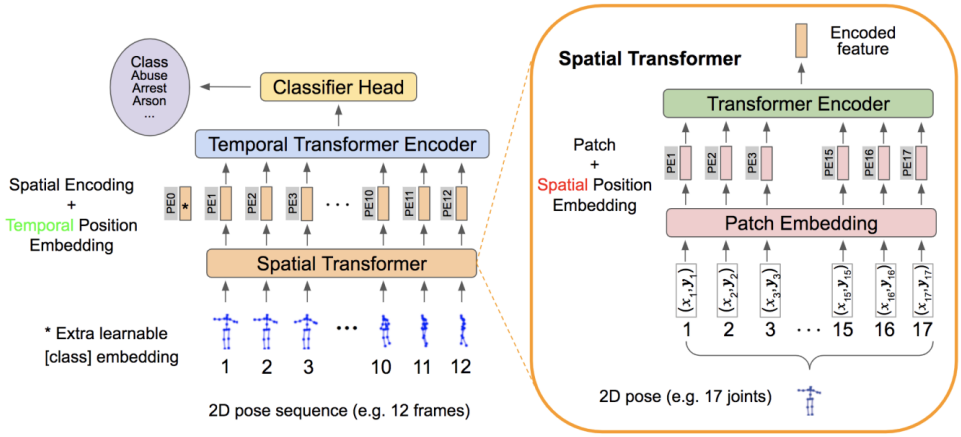


Fig. 27: ST-Tran architecture [7].

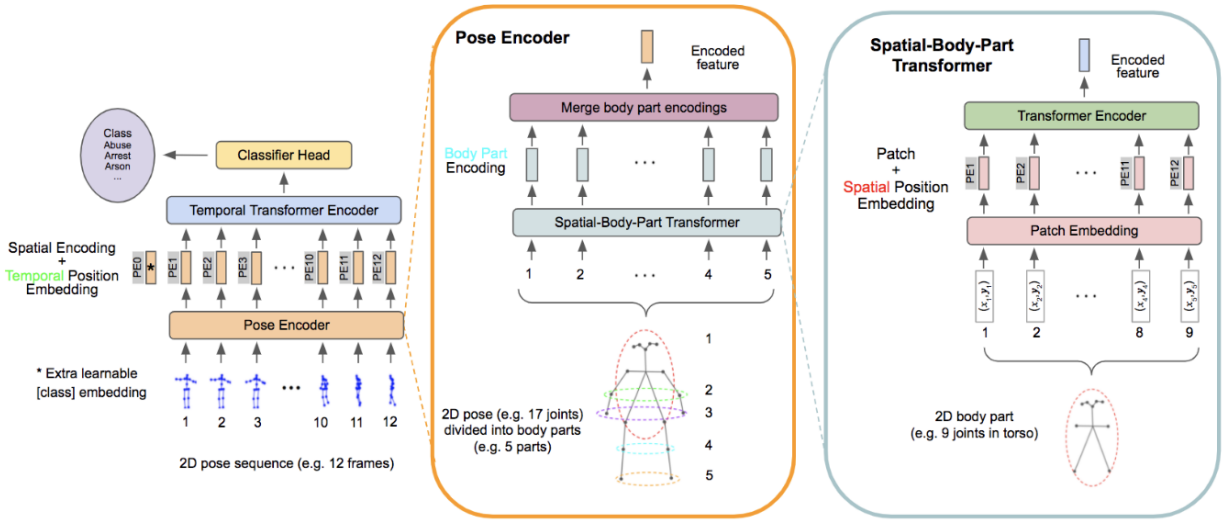


Fig. 28: SBPT-Tran architecture [7].

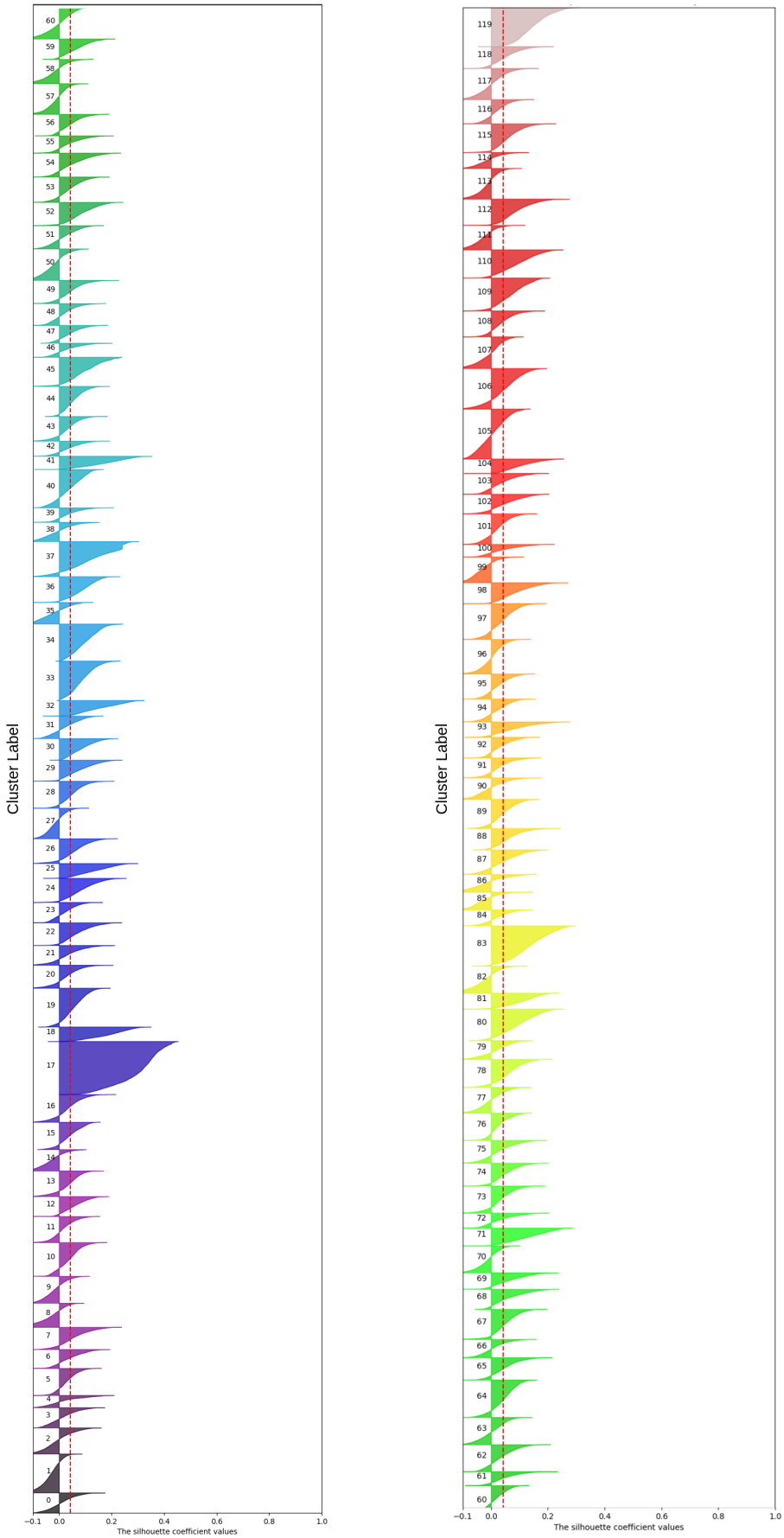


Fig. 29: Silhouette plot for the classifications of the BPTubeFormer model on the NTU120 dataset. The y axis labels indicate the activity class labels. The red line indicates the average silhouette score for all the classes.

Class Label	Activity
A1	drink water.
A2	eat meal/snack.
A3	brushing teeth.
A4	brushing hair.
A5	drop.
A6	pickup.
A7	throw.
A8	sitting down.
A9	standing up (from sitting position).
A10	clapping.
A11	reading.
A12	writing.
A13	tear up paper.
A14	wear jacket.
A15	take off jacket.
A16	wear a shoe.
A17	take off a shoe.
A18	wear on glasses.
A19	take off glasses.
A20	put on a hat/cap.
A21	take off a hat/cap.
A22	cheer up.
A23	hand waving.
A24	kicking something.
A25	reach into pocket.
A26	hopping (one foot jumping).
A27	jump up.
A28	make a phone call/answer phone.
A29	playing with phone/tablet.
A30	typing on a keyboard.
A31	pointing to something with finger.
A32	taking a selfie.
A33	check time (from watch).
A34	rub two hands together.
A35	nod head/bow.
A36	shake head.
A37	wipe face.
A38	salute.
A39	put the palms together.
A40	cross hands in front (say stop).
A41	sneeze/cough.
A42	staggering.
A43	falling.
A44	touch head (headache).
A45	touch chest (stomachache/heart pain).
A46	touch back (backache).
A47	touch neck (neckache).
A48	nausea or vomiting condition.
A49	use a fan (with hand or paper)/feeling warm.
A50	punching/slapping other person.
A51	kicking other person.
A52	pushing other person.
A53	pat on back of other person.
A54	point finger at the other person.
A55	hugging other person.
A56	giving something to other person.
A57	touch other person's pocket.
A58	handshaking.
A59	walking towards each other.
A60	walking apart from each other.

TABLE XVII: List of activities in the NTU60 [99] dataset

Class Label	Activity
A61	put on headphone.
A62	take off headphone.
A63	shoot at the basket.
A64	bounce ball.
A65	tennis bat swing.
A66	juggling table tennis balls.
A67	hush (quite).
A68	flick hair.
A69	thumb up.
A70	thumb down.
A71	make ok sign.
A72	make victory sign.
A73	staple book.
A74	counting money.
A75	cutting nails.
A76	cutting paper (using scissors).
A77	snapping fingers.
A78	open bottle.
A79	sniff (smell).
A80	squat down.
A81	toss a coin.
A82	fold paper.
A83	ball up paper.
A84	play magic cube.
A85	apply cream on face.
A86	apply cream on hand back.
A87	put on bag.
A88	take off bag.
A89	put something into a bag.
A90	take something out of a bag.
A91	open a box.
A92	move heavy objects.
A93	shake fist.
A94	throw up cap/hat.
A95	hands up (both hands).
A96	cross arms.
A97	arm circles.
A98	arm swings.
A99	running on the spot.
A100	butt kicks (kick backward).
A101	cross toe touch.
A102	side kick.
A103	yawn.
A104	stretch oneself.
A105	blow nose.
A106	hit other person with something.
A107	wield knife towards other person.
A108	knock over other person (hit with body).
A109	grab other person's stuff.
A110	shoot at other person with a gun.
A111	step on foot.
A112	high-five.
A113	cheers and drink.
A114	carry something with other person.
A115	take a photo of other person.
A116	follow other person.
A117	whisper in other person's ear.
A118	exchange things with other person.
A119	support somebody with hand.
A120	finger-guessing game (playing rock-paper-scissors).

TABLE XVIII: Additional activities included in the NTU120 [93] dataset, apart from those in Table XVII.

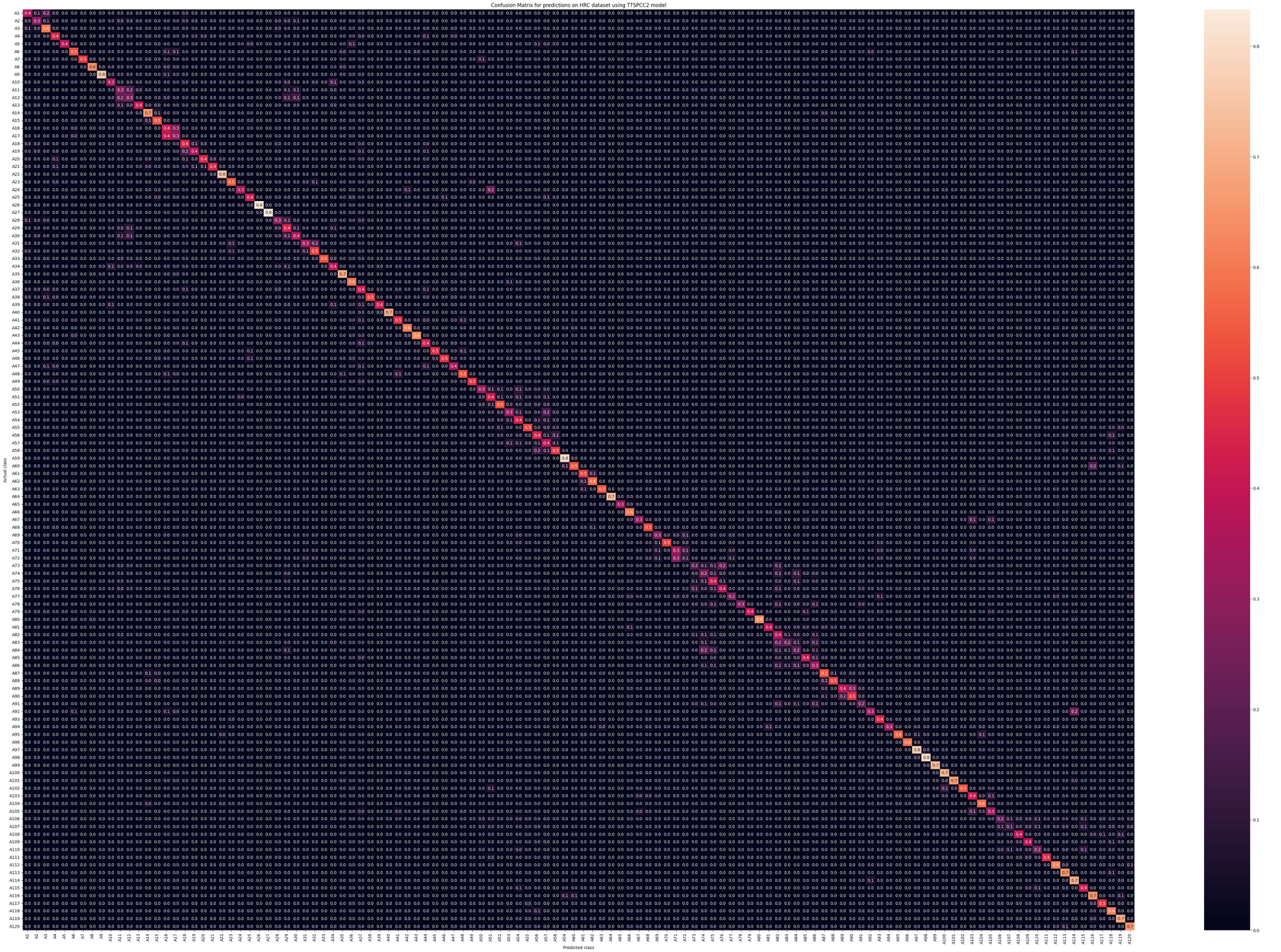


Fig. 30: Confusion matrix for BPTubeFormer-6-V2 using NTU120 dataset.