Contents lists available at ScienceDirect

Computers & Graphics

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

SHREC 2022 Track on Online Detection of Heterogeneous Gestures

Ariel Caputo[a], Marco Emporio[a], Andrea Giachetti[a], Marco Cristani[a], Guido Borghi[b], Andrea D'Eusanio[c], Minh-Quan Le[d], Hai-Dang, Nguyen[d], Minh-Triet Tran[d], Felix Ambellan[e], Martin Hanik[e], Esfandiar Nava-Yazdani[f], Christoph von Tycowicz[e]

[a] *University of Verona, Department of Computer Science*
[b] *Università di Bologna, Dipartimento di Informatica, Scienza e Ingegneria*
[c] *Università di Modena e Reggio Emilia, Dipartimento di Ingegneria "Enzo Ferrari"*
[d] *University of Science, Ho Chi Minh City, Vietnam*
[e] *Freie Universit at Berlin, Berlin, Germany*
[f] *Zuse Institute Berlin, Berlin*

ARTICLE INFO

Article history:
Received July 25, 2022

Keywords: Computers and Graphics, Formatting, Guidelines

ABSTRACT

This paper presents the outcomes of a contest organized to evaluate methods for the online recognition of heterogeneous gestures from sequences of 3D hand poses. The task is the detection of gestures belonging to a dictionary of 16 classes characterized by different pose and motion features. The dataset features continuous sequences of hand tracking data where the gestures are interleaved with non-significant motions. The data have been captured using the Hololens 2 finger tracking system in a realistic use-case of mixed reality interaction. The evaluation is based not only on the detection performances but also on the latency and the false positives, making it possible to understand the feasibility of practical interaction tools based on the algorithms proposed. The outcomes of the contest's evaluation demonstrate the necessity of further research to reduce recognition errors, while the computational cost of the algorithms proposed is sufficiently low.

1. Introduction

The online recognition of gestures from 3D fingers' tracking data streams is an extremely interesting and hot problem both from an algorithmic and an application point of view. Algorithms used should combine a geometrical and temporal encoding of the hand movements as well a classification approaches able to avoid false positives. This means that they should be able to discriminate the gesture classes from a non-gesture class, the latter characterized by spurious movements with large variance. There are many applications of online gesture recognition with a potentially huge impact. One of these is certainly that relating to the development of interactive interfaces for virtual and augmented reality applications. These interfaces should allow different types of interaction with objects and widgets without the need for handheld devices. To build such interfaces it is necessary to implement effective recognizers able to cope with a sufficiently large dictionary of gestures of different types (hand motions, hand articulations, static poses).

To evaluate the potential effectiveness of these algorithms, it is not only necessary to test the ability of the methods to classify segmented gestures, but also to test how well they avoid false

detections and which is their detection latency.

As we will discuss in Section 2, the existing benchmarks are not optimally suited for this, so we created a novel one and organized a related contest. The main contributions of the dataset, task, and evaluation method described here are:

- The benchmark is specifically designed for generic mixed reality interaction development and directly captured with the hand tracking system of the Hololens 2 headset, therefore featuring the same viewpoint, field-of-view, and temporal resolution. The methods tested on this benchmark could directly be applied in an interactive application developed on the same platform, locally or via a client-server architecture, depending on the computational complexity.
- The benchmark features heterogeneous gestures, including *static* ones (fixed hand pose for at least 0.5s), *dynamic coarse* (characterized by whole hand trajectory), *dynamic fine* (where the semantic depends also on finger articulations), and *periodic* (with repeated movements) gestures. With respect to the only previous benchmark for online gesture recognition (SHREC 2021 [1]), the dictionary has been changed removing ambiguous classes, avoiding annotation issues affecting that dataset (see Section 2) and introducing the class of periodic gestures.
- The evaluation measures online performances, not only considering non-gestures and false positives but also evaluating the recognition latency, which is an important factor to assess their usability in an interactive application.

2. Related work

A consistent body of research has been dedicated to the problem of hand gesture recognition, and several benchmarks are available. Most of these benchmarks are not, however, measuring online recognition performances. A popular benchmark of this kind is the SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset [2], featuring dynamic gestures involving global motions and fingers' articulation that can be used to build interactive applications. Many methods

Fig. 1: Gesture acquisition: the subject hears vocal commands suggesting predefined sequences of non-significant movements and gestures.

for offline classification of segmented gestures have been evaluated on this benchmark or the similar Dynamic Hand Gesture dataset (DHG) 14/28 [3]. However, the accuracy measured on the offline classification task does not tell too much about the usability for practical gesture detection in a realistic scenario of use.

Garcia-Hernando et al. [4] proposed a benchmark of hand actions captured with both RGB, depth, and magnetic sensors and inverse kinematics and the dataset. The actions recorded, however, are not gestures used in an interaction scenario, and skeletons are not directly provided. The task proposed in the SHREC 2019 track on online gesture detection [5]) is to find gestures in hand skeleton sequences, thus addressing the problem of avoiding false positives, but the dictionary was limited to simple dynamic gestures characterized by the mere hand trajectory. The task proposed in the SHREC 2021 track on Skeleton-based Hand Gesture Recognition in the Wild was better suited to test the potential use for XR interaction of complex gestures, as it featured static, dynamic-coarse, and dynamic-fine gestures and used an evaluation method based on detection rate, false-positive score. However, the dataset has some weaknesses: first, some gestures are ambiguous, as we found that, at the end of the execution of some dynamic gestures, there are static poses of the hand similar to the "pointing" gesture included in the dictionary, but not annotated. Furthermore, the annotation of many dynamic gestures was limited to a very short segment of the whole hand movement. Finally, the evaluation did not consider the recognition latency.

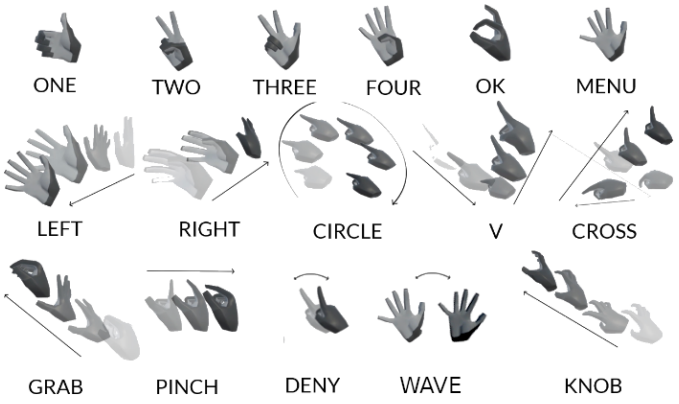


Fig. 2: The 16 gestures in SHREC'22 vocabulary: static (top row), dynamic coarse (middle row), dynamic fine (bottom left) and periodic (bottom right).

3. Novel dataset, task and evaluation

For the reasons we mentioned, we created another dataset trying to overcome the issues of the previous ones and using directly the data captured by the head-mounted display.

The dataset is composed of 288 sequences including a variable number of gestures (i.e. 3 to 5), divided into two subsets of the same size: a training set, with annotations provided to participants (start and end frame of the gestures with related label), and a test set where the gestures have to be found according to the task requirements.

Fingers data were captured with a Hololens 2 device simulating mixed reality interactions. The gestures in each sequence are interleaved with other hand movements. In the acquisition session, the Hololens 2 app was programmed to ask the subjects (with a vocal command) to start specific gestures at randomized time frames (Figure 2). These time frames were also recorded and, for the training set, they were given as additional data to the participants.

During the remaining time, the subjects were instructed to keep their hands in the field of view of the tracking system and to freely move the hands avoiding movements in conflict with the actual gestures from the dictionary.

The sequences have been designed so that both the training and the test set include the same number of instances of each gesture class (36).

Time sequences were saved as text files where each row represents the data of a specific time frame with the coordinates of 26 joints. Each joint is therefore characterized by 3 floats (x,y,z position). The frame rate of the acquisition is relatively low and not perfectly stable (approximately 20 fps), making the recognition scenario even more realistic considering an unstable frame rate is a common condition for real applications running on stand-alone devices with limited performances. Frame data include, however, also the time stamp of the recordings, making it possible to resample the joint positions at a constant rate.

The gesture dictionary is similar to the one proposed in SHREC 2021, but with a few, important changes. Some gesture classes have been removed (POINTING, TAP, EXPAND). Analyzing the gestures and the annotations in the dataset, we found that several dynamic coarse gestures featuring hand trajectories ended with a static pointing pose lasting several frames and that ideally could have been annotated as a pointing. Similarly, the short tap and expand annotated sequences were quite similar to parts of other gestures. It is true that the gesture recognition procedure could in principle be able to disambiguate the classes using context we considered better to focus in this contest on the recognition of less ambiguous pattern, leaving a more challenging context-based disambiguation for future work.

We also found that the annotation of the dynamic-fine gestures (PINCH, GRAB) in SHREC 2021 did not include the initial global hand movements. In the novel dataset, we annotated those frames as belonging to the gesture as we consider them as characteristic of the gesture classes.

As shown in Figure 1, we included 16 gestures divided in 4 categories: *static* characterized by a pose kept fixed (for at least 0.5 sec), *dynamic coarse*, characterized by a single trajectory of the hand, *dynamic fine*, characterized by fingers' articulation, *periodic*, where the same fingers' motion pattern is repeated more times.

For each of these categories the features actually determining the related semantics are different, and, in principle, it would be possible to develop specialized approaches for their classification. None of the participants, however, considered this option.

Table 1 shows the minimum, maximum and average length of

	ONE	TWO	THREE	FOUR	OK	MENU	LEFT	RIGHT	CIRCLE	V	CROSS	GRAB	PINCH	DENY	WAVE	KNOB
Min length	21	24	15	23	22	20	11	11	25	17	24	19	22	31	27	37
Max length	72	59	71	69	52	59	37	32	64	37	50	70	61	81	73	79
Avg. length	**34.1**	**37.4**	**38.6**	**37.3**	**34.0**	**33.0**	**19.6**	**18.6**	**45.4**	**27.2**	**37.5**	**39.9**	**36.2**	**47.8**	**45.4**	**54.7**

Table 1: Minimum, maximum and average lengths of the gestures (background colors indicate gestures' types.

each gesture in the training set. It is possible to see that the static gestures are kept for variable times. Their semantics, however, do not depend on the full length. We expect, therefore, that they can be recognized just after the start frames. The algorithms should however take care of avoiding multiple detection within the annotated interval.

Dynamic gestures have a variable time length, and their semantics is, in principle dependent on the full duration with the exception of periodic ones that may be fully characterized by the first occurrence of the repeated sequence.

3.1. Task and evaluation

Participants were asked to create an online classification method based on training sequences and to process the test sequences not only annotating the predicted start and end of gestures but also indicating the last frame of the sequence used to perform the prediction, giving the indication of the minimal classification delay (not including the algorithm's execution time).

The evaluation was automatically performed with a script counting the number of correct gestures detected, the false positive score, and the minimal detection delay.

A detected gesture is considered correctly detected if has the same label as the ground truth annotation, and its time window has an intersection with the annotated one larger than half the ground truth length. We call *detection rate* the ratio between the number of correctly detected gestures of the class and the corresponding ground truth number.

The *false positives' score* is the ratio between the number of gesture predictions not corresponding to real ones (e.g. with no intersection with annotated ground truth gestures) and the total number of gestures of the same class included in the test set.

The script also evaluates the Jaccard Index (JI), e.g. the average relative overlap between the ground truth and the predicted annotations of frames belonging to each gesture class in the sequences. This metric had been employed in other online action and gesture recognition contests [6, 7].

An important novelty of our benchmark is the evaluation of the detection delay, estimated as the difference between the actual gesture start and the reported timestamp of the last frame used for the prediction and is related to the algorithmic detection strategy. Given this estimate, it is possible to evaluate the average delay of the gestures' detection with respect to the actual (ground truth) starts as well as the time differences with respect to the gestures' ends. The analysis of these values gives interesting hints about the usability of the proposed methods for practical applications. To complete the temporal data analysis, we also collected from the participants the computational time for a single classification step (i.e. how much time each method takes to elaborate data for a single classification attempt).

4. Participants and proposed methods

Three groups were registered for the contest and submitted up to three results' files obtained with different classification strategies as well as the required additional information on the algorithms' runs. We compared their results using the previously described methods against a baseline method, a modified version of STRONGER [8], an online recognizer based on 1D convolutional neural networks.

4.1. Group 1: Two-stage ST-GCN (2ST-GCN)

Method Description To address the task proposed for this task effectively, Group 1 constructed their method by adapting a two-stage object detection model: the R-CNN family. Figure 3 illustrates the workflow of the two-stage hand gesture detection architecture.

First, the method leverages the idea of a sliding window with a small window size striding along the sequences to propose gesture candidates. Those candidates serve as inputs to a tiny

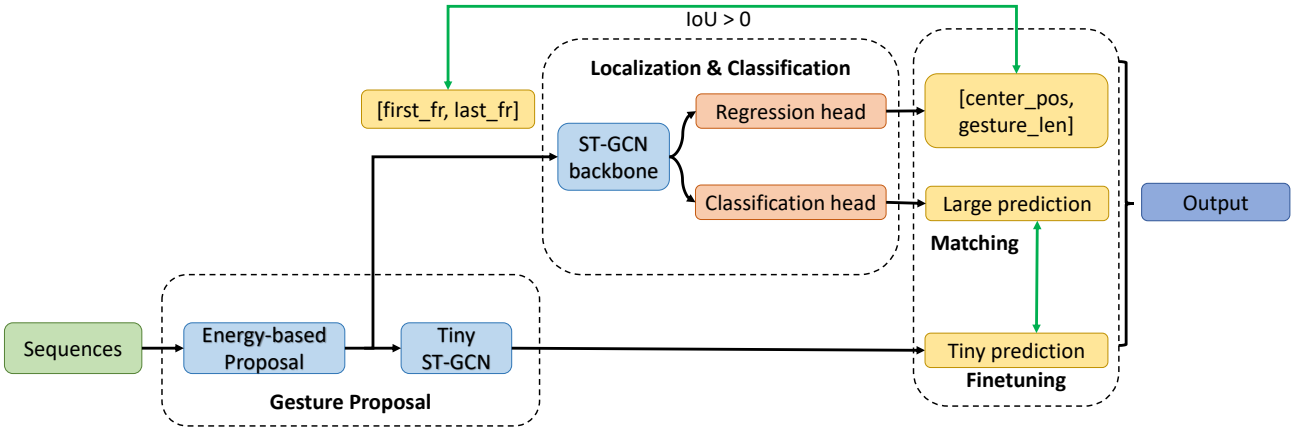


Fig. 3: *The workflow of our two-stage hand gesture detection architecture.*

classification model which results in categories of each short sequence. Next, it extends these sequences to a larger size and feed them to a large model with both classification and localization branches. Finally, a fine-tuning and matching the outputs of tiny and large models is performed to return the final results.

Gesture Proposal Module Group 1 proposes the Gesture Proposal Module which comprises the energy-based sliding window and a tiny classification. They borrow the idea of the energy-based function from [1] to leverage the shift of every joint of human hand over sequences of consecutive frames and calculate the amount of energy accumulated in a window with a size of l

$$E(w) = \sum_{j=1}^{N} \sum_{t=1}^{l} \frac{\|w_{j,t} - w_{j,t-1}\|}{\|w_{j,t-1}\|}$$
$$w_{j,t} = [x_{j,t}; y_{j,t}; z_{j,t}]^T$$
$$\|w_{j,t}\| = \sqrt{x_{j,t}^2 + y_{j,t}^2 + z_{j,t}^2}$$

where $w_{j,t}$ is 3D coordinates of finger-joint j at time step t and N is the number of joints in human-hand.

Next, they utilize a sliding window with a small size $l = 10$ to stride over the entire sequences with step_size $= 1$ and calculate movement energy at each step. A potential segment w_i is chosen if its energy is higher than average energy from w_0 to w_{i-1}.

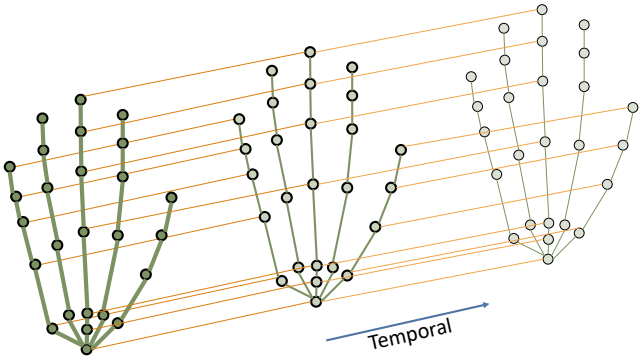


Fig. 4: *Spatial-temporal graph neural networks for hand gesture recognition.*

In the following step, those 10-frame long sequences are fed into a tiny classification model for early prediction of the gestures category. Based on the structure of the dataset, which includes 3D trajectories of finger-joints, the authors consider reasonable to follow the natural structure of human hands and represent them as graphs containing both spatial and temporal information. Therefore, Group 1 applies Spatial-Temporal Graph Convolutional Networks (ST-GCN) [9] with the purpose of learning patterns embedded in the spatial configuration by exploring locality of graph convolution as well as temporal dynamics. Spatio-temporal links are represented in Figure 4. The ST-GCN module outputs categories along with confidence scores of each short segment. The training of this network module is based on 10-frames sequence randomly extracted in the range $[\text{first\_frame} - 5, \text{last\_frame} + 5]$ around the labeled

gestures annotated in the training set and following the protocol described in [9]. A non-gesture class is also added to the training procedure to make the predictions more accurate.

Localization and Classification Module At runtime, when the tiny ST-GCN in the proposal module classifies a small window [first_frame, last_frame] (last_frame − first_frame = 10) as a specific gesture, another ST-GCN module is activated, processing a window of larger size ($L = 80$), placed in the interval [last_frame − 80, last_frame].

In this part, ST-GCN is used as a backbone feature extractor to exploit the spatial relationships between finger-joints as well as temporal information. Furthermore, the regression head for localization and the classification head with a softmax layer are appended to the graph convolution backbone. The regression head will output two values including [center_pos, gesture_len] which denote the center position and the number of frames of each gesture respectively (Figure 5).

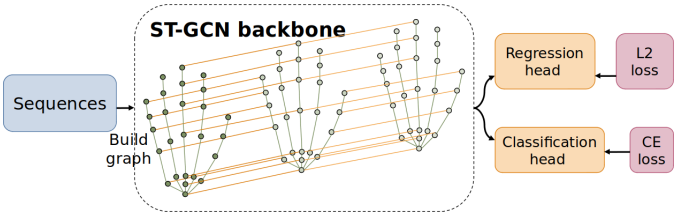


Fig. 5: *The modified architecture of ST-GCN with both regression and classification heads*

The module is trained as follows: From each annotated gesture in the training sequences, including the values of [CLASS, begin_frame, end_frame], training examples of length $L = 80$ between [end_frame − L, start_frame + L] are extracted. Each sample is associated with the corresponding ground-truth of center positions and length of gestures for localization branch training purposes:

$$\text{mid\_frame} = \frac{\text{start\_frame} + \text{end\_frame}}{2}$$

$$\text{GT}_{\text{center\_pos}} = \frac{\text{mid\_frame} - \text{selected\_start\_frame}}{L}$$

$$\text{GT}_{\text{gesture\_len}} = \frac{\text{end\_frame} - \text{start\_frame}}{L}$$

As in the training of the tiny module, examples of 80-frames non-gestures sequences labeled as a further class, are added into the pipeline to improve the model's discriminating ability. L2 loss and Cross-Entropy loss are used for training regression head and classification head respectively.

Moreover, due to the limitation on training data, a stratified 5-fold based on class-distribution is also applied to avoid under and over fitting.

Fine-tuning and Matching Module The output of tiny models from the proposal module and the output of large models from the localization and classification module are fine-tuned and go through matching condition checking to decide the last results. More concretely, the final outcomes must satisfy two matching conditions: the predicted category from the tiny model is the same as that of the large model and the overlap ratio of short segments from proposal module and predicted segments from regression head is larger than zero.

$$\text{cls\_pred\_tiny} = \text{cls\_pred\_large}$$

$$\text{IoU}([\text{first\_frame}, \text{last\_frame}], [\text{pred\_start}, \text{pred\_end}]) > 0$$

The final outputs consist of 4 components:

$$[\text{cls\_pred\_tiny}, \text{pred\_start}, \text{pred\_end}, \text{last\_frame}]$$

System Configuration Group 1 submitted the results of two experiments, one based on a single model (RUN1) and the other on the stratified 5-fold model (RUN2). The average time for the classification step was 2.1 ms in both the cases, running on a single GPU NVIDIA QUADRO RTX 5000. The group actually planned to perform a further run adding an additional feature, the Orientation Histogram [10] to discriminate gestures characterized by 2D trajectories but the test was finally left for future work.

4.2. Group 2: Causal TCN

Method Description The proposed method by Group 2 is based on a temporal convolutional network (TCN) architecture that employs causal filters preventing any 'leakage' of future information to the past. In recent years, TCNs have been found to convincingly outperform canonical recurrent neural architectures across a broad range of sequence modeling tasks [11] by

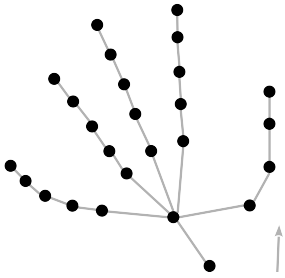


Fig. 6: Segments joining given anatomical landmarks. Angles between all pairs as well as with the fixed global axis (arrow right) serve as input features for gesture detection.

demonstrating longer effective memory and improved stability (mitigating the vanishing/exploding gradient problem). While there are various TCN-based action recognition approaches in the literature (see e.g. [12] and the references therein), Group 2 proposes a notably lightweight network structure that features a very low model size (only 125k parameters) and, hence, lends itself for efficient inference even on devices with limited computing power.

Feature Description Since the given per-frame landmarks are strongly coupled, this inherent structure should be exploited when designing expressive features. This coupling is furthermore naturally invariant under the Euclidean motion (e.g. due to change of camera position and orientation) and hence, they decided to utilize angles between all anatomically definable segments as features. This representation for a skeleton configuration is not only location-viewpoint invariant but is also agnostic to its laterality. However, some of the gesture classes can only be distinguished reliably if their global context is additionally taken into account (e.g. MENU and WAVE). Therefore, another artificial segment along a fixed axis in space was added. As the 26 anatomical landmarks form 25 segments (see Fig. 6), there are a total of 26 different joints/vectors yielding 351 angles for each input frame.

Network Architecture As main building blocks in the proposed model Group 2 employs causal or unidirectional convolutional filters. Commonly dilated convolutions are used in order to build networks with very long effective memory. However, as the contest aims at randomized temporal sequences of gestures each of which being a short-timed activity, they opted for non-dilated convolutions. To reduce the complexity of the learning task they follow a windowing approach, i.e. conditioning the model to detect gestures for short-time windows containing n consecutive frames. In particular, each window is fed through two convolutional layers (feature dimension 64 and 32) each with a kernel size of 5 and followed by a Leaky ReLU activation. Subsequently, a fully connected, linear layer maps the output of all frames within the window onto a 17-dim output representing a one-hot encoding of the probabilities of the 16 gesture classes and a background one. The common cross entropy together with a L_2 regularization term is employed as loss function.

Training As within the training data every gesture appears 36 times, Group 2 decided to perform a stratified six-fold approach on the occurrences to split into training and validation set. They trained six different models, in order to take all available information from the six trained models into the prediction phase. They sample the input windows of length $n = 20$ for every gesture s.t. window and gesture are at least featuring 50% overlap in order to not provide the network with practically meaningless gesture chunks. Additionally, they sample background windows following the same rule. However, since the amount of background windows outnumbers the amount of gesture windows by an order of magnitude only 10% of background ones is considered for training, drawn randomly. Training is performed with a batch size of 15 for 100 epochs. In every step exponential moving average is applied to the model parameters to track averaged parameters for prediction. Every 1000 batches these averaged parameters are employed to determine the validation accuracy. Finally, the parameter set featuring the highest validation accuracy over all evaluations is chosen as final parameters.

Online Detection For every frame, voting contributions from every window it belongs to is collected, i.e. a frame gets up to n single votes for a label within its temporal context. This indicates that there is a constant delay of $n-1$ regarding the gesture start prediction. Each window label prediction is ensembled from the output logits across k nets. To this end logit vectors are

normalized (L_2), summed and finally evaluated with *argmax* to assign a class label.

In order to deal with fuzzy gesture endings Group 2 applies a simple post-processing strategy. If two consecutive frames f_0, f_1 belong to different continuous (non-background) label-chunks and (a) the chunk of f_0 is larger than the one of f_1 and the length of the f_0-chunk is less than n then the f_1-chunk is set to background or (b) the length of the f_0-chunk is less than n, then the f_0-chunk is set to background. Finally, continuous chunks of length less then 9, that are surrounded by background, are also merged into background. These strategies technically use more than $n-1$ future frames. However, the thusly discarded gesture detections are not captured by the evaluation protocol.

The average per-frame detection takes $\approx 2.8 \times 10^{-2} s$ mainly consumed by the temporal voting and the k-ensembling.

System Configuration Training and detection were carried out on a (Debian 11) workstation featuring an Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz processor, 128GB RAM and a NVidia GeForce RTX 3090 (24GB). The implementation was realized with python utilizing jax/jaxlib (0.3.1), dm-haiku (0.0.6) and optax (0.1.1).

4.3. Group 3: Transformer Network + Finite State Machine (TN-FSM)

Method Description The proposed approach is mainly divided into three main logical blocks: the first one enriches the features coming from the input data represented by 3D hand joint positions grouped in consecutive frames. Specifically, each frame contains a set of hand joints acquired at the same time. The second block classifies the computed features, *i.e.* outputs a gesture or a non-gesture label for each input frame. Finally, the third block receives as input a set of consecutive frames, here referred to as *window*, in which each frame is coupled with its gesture label. As output, it provides the boundaries of each gestures, *i.e.* the beginning and the end of each gesture (and therefore implicitly the presence of non-gestures).

With Group 3 approach, the final performance of the proposed method relies on the performance of every single module, in particular on the ability to accurately classify frames as gestures (even if with a wrong label for some frames) or non-gestures.

Feature Description As mentioned above, input data consist of the 3D positions of the hand joints grouped in frames. Formally, at a given time t, it is available a sequence of joints $J_t = \{j_i^t \mid j_i^t = (x_i^t, y_i^t, z_i^t),\ 1 \le i \le N\}$, where N is the total number of the hand joints (in this case $N = 26$ different joints acquired through a *Hololens 2* device simulating a mixed-reality interaction). This block enriches this input vector with other 4 types of features computed starting from 3D joint positions. The first two are represented by the speed (s) and the acceleration (a) of each hand joint, computed with the following equation:

$$\mathbf{s}_i^t = \left[x_i^t - x_i^{(t-1)},\ \ y_i^t - y_i^{(t-1)},\ \ z_i^t - z_i^{(t-1)}\right]$$

$$\mathbf{a}_i^t = \left[x_i^t - 2x_i^{(t-1)} + x_i^{(t-2)},\ \ y_i^t - 2y_i^{(t-1)} + y_i^{(t-2)},\ \ z_i^t - 2z_i^{(t-1)} + z_i^{(t-2)}\right]$$

Then, joint-to-joint distance (JD) features are added, expressed as a matrix $D = 3 \times N \times N$, containing information about the 3D distances. Each element $\mathbf{d} \in D$ is a set of three coordinates and is obtained following this equation:

$$\mathbf{d}_{i,k} = \sqrt{(\mathbf{j}_i^t - \mathbf{j}_k^t)^2}, \quad k,\ j \in N$$

Finally, the input feature vector is further enriched by computing the spherical coordinates (r, θ, φ) of the joints' positions starting from the 3D Cartesian coordinates (x, y, z):

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arctan \frac{\sqrt{x^2 + y^2}}{z}$$

$$\varphi = \arctan \frac{x}{y}$$

It can be observed that the value of the inverse tangent in φ can have different values depending on the correct quadrant of (x, y).

To summarize, the final input vector is a concatenation of 4 different feature vectors that represent different aspects of gestures: hand position, joint movements (in terms of speed and acceleration) and hand shape (distances can represent the state of the hand opening or closing). Therefore, the feature vector length with all types exploited is $26 \times (3+3+3+26+3) = 988$.

Each joint position is then normalized to obtain a zero mean and unit variance for each 3D axis.

Frame classification In this block, each frame is classified using the features computed as detailed in the previous step. Group 3 adopts a classifier architecture divided into two main parts: the first one is a transformer-based model [13], while the second one is a fully connected layer used to output the frame classification with the correct shape. In this way, each frame is classified as non-gesture or with a gesture label.

From a formal point of view, the model is defined as:

$$Y(\mathbf{x}) = \mathrm{F}(\mathrm{Encoders}(\mathbf{x} + PE))$$

As mentioned, there are two main parts. The first is F(·), which corresponds to the fully connected layer with a softmax layer needed for the classification task. This block is applied on each time step $x \in \mathbf{x}$. The second part if a set of Encoders(·), consisting in a sequence of 6 transformer encoders E and the *Positional Encoding* (PE) [13].

An encoder is described by:

$$E(x) = \mathrm{Norm}(x + \mathrm{FC}(\mathrm{mhAtt}(x)))$$

where FC(·) are two fully connected layers with 2048 units, followed by a ReLU activation function and Norm(·) is a normalization layer. The Positional Encoding appears to be essential in order to encode in the adopted model the temporal information of the sequence defining a vector that contains a probability distribution over n gesture classes for each time step included in $\mathbf{x}$. Finally, mhAtt is the multi-head attention block:

$$\mathrm{mhAtt}(x) = (\,\mathrm{Att}_1(x) \oplus \ldots \oplus \mathrm{Att}_8(x)\,)\, W^O$$

where

$$\mathrm{Att}_i(x) = \mathrm{softmax}\left(\frac{Q_i\, K_i}{\sqrt{d_k}}\right) V_i$$

Here, $K_i = xW_i^K$, $Q_i = xW_i^Q$, $V_i = xW_i^V$ are independent linear projections of x into a 64-d feature space, $d_k = 64$ is a scaling factor corresponding to the feature size of K_i, W^O is a linear projection from and to a 512-d feature space and $\oplus$ is the concatenation operator.

The training of the classifier is based only on the provided datasets. This is a non-trivial procedure since gestures tend to be short and then the majority of data are labeled as non-gestures. Therefore, Group 3 adopts the *Focal Loss* [14] in order to contrast the unbalanced training dataset, instead of the most common *Categorical Cross Entropy* loss for multi-classes classification scenarios. As optimizer, Group 3 use the the *AdamW* [15] algorithm, that in our experiments shows better performance w.r.t. *Adam*, with a initial learning rate of 10^{-4}, 0.5 of internal dropout and weight decay of 10^{-4}. The model is trained creating sequences with 10 consecutive frames. All the model parameters rely on a k-fold ($k = 9$) cross-validation.

Gesture Detection A *Finite State Machine* (FSM) is used to define the boundaries of each gesture, relying on the gesture labels provided by the classifier. The FSM is based on four different states and the input is represented by a sliding window (i.e. a buffer) of 10 frames. The FSM runs only when the buffer is full. It is important to note that the final gesture class is predicted only once detected the beginning and the end of a gesture. Specifically, the final class corresponds to the most predicted class into the input window.

The first state of the FSM is used to detect the beginning of a gesture and it is maintained as the current internal state until the window contains only frames classified as non-gesture by the transformer-based classifier.

When at least one frame is classified as a gesture, the internal state of the FSM is increased. In this second state, a check about the beginning of the gesture is conducted: this control is motivated by the possible presence of frames wrongly classified. Indeed, the beginning of a gesture is confirmed only if at least w_i different frames in 10 consecutive windows are classified as gestures and then the FSM passes to the third state. In order to handle gestures with a very limited length, the FSM can pass directly in the fourth state if a non-gesture, i.e. a whole window with all frames classified as non-gesture, is found.

In the third state, the end of the gesture is detected. In this state, the end of a gesture corresponds to a window that contains only non-gesture classifications and when detected, the FSM internal state is increased.

In the fourth state, a check at the end of a gesture is per-

formed, since, as aforementioned, some frames can be wrongly classified. Only if w_e consecutive frames are classified as non-gesture, the FSM detects the end of the gesture. In case of a single frame is classified as gesture, the FSM returns in the third state.

Run tests Group 3 tested three different versions of the proposed pipeline, focusing in particular on different feature types combinations used as input. It has been empirically observed that the classification performance of the transformer-based model achieves a good accuracy with the detailed setting while for the Finite State Machine best results are obtained setting $w_i = 5$ and $w_e = 10$. In the first solution (TN-FSM), it has been trained with feature vector containing only the position, the speed and the acceleration. In the second case, features of the joint distances have been also added (TN-FSM+JD) while in the third one spherical coordinates were added (TN-FSM+SC).

System Configuration The proposed system ran on a computer equipped with an *Intel Core i7-7700K* and the dedicated GPU *NVidia GeForce GTX 1080 Ti*. The observed inference time on GPU is about 4.65 ± 0.39 ms and 4.72 ± 1.13 ms with only CPU (averaged on 100 different runs), denoting that the implementation of the transformer-based model is not optimized for high-level parallelism.

4.4. Our baseline: Stronger

The baseline we added in the comparison is a variation of the recognizer used in [8]. The method is based on modified version of the DDNet architecture [12], customized by adding novel features and related branches and trained for online detection using a sliding window approach for the continuous classification. The classifier is trained to process and automatically label segmented hand pose sequences, that are resampled to a standard number of time steps and pre-processed to extract a set of features passed to the network.

The network architecture is shown in Figure 7. Five input vectors are processed in parallel with 1D convolutions to obtain latent vectors that are then concatenated and passed through 3 more convolutional layers, a Global Average Pooling (GAP) and a Fully Connected layer (FC) providing the class probabilities.

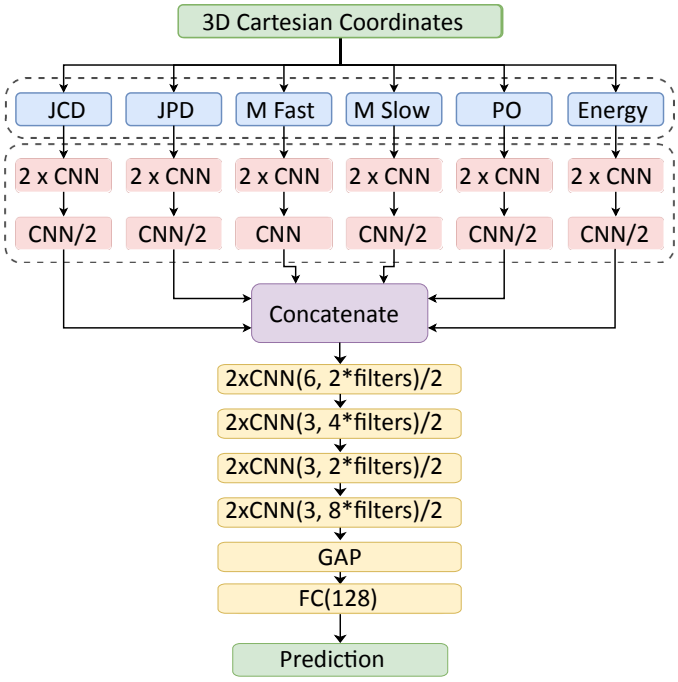


Fig. 7: STRONGER uses a modified DDNet architecture to perform the gesture classification. Six input features are processed in different branches: joint collection distances (JCD), joints pair differences (JPD), joints' motion estimated at two different scales (Mslow and Mfast), Palm Orientation and Energy.

The input vectors are joints' velocities computed at two different scales (Mslow and Mfast), the linearized matrix with the joint distances (JCD), the palm orientation and a set of unit vectors directed as the segment joining selected couple of joints (JPD).

In the implementation used on the SHREC'22 benchmark a sixth input feature was also tested, namely the re-sampled sequence of kinetic energies.

For the online classification, a specific training and online processing has been designed. The network is trained with examples of 17 classes, 16 representing the gestures in the dictionary and the other representing the non-gestures. Segmented gestures are obtained from the training sequences and the annotated initial and final frames. Non-gesture sequences are randomly extracted as variable length sequences not included in the annotated gesture intervals. Independently of the original length, all the feature vectors in the segmented training samples are uniformly re-sampled to a fixed number of samples (30) before sending them to the network.

After the network training, we defined specific thresholds for

the acceptance of the classification results as follows: for each gesture class we estimated the class probabilities of the correctly estimated gestures (almost 100%) and the corresponding second highest probability among the other classes. The average of these probabilities has been taken as a threshold to discard gesture detection for that class with lower probability. This trick is able to drastically reduce the amount of false positives.

As the gesture duration is variable the test classification is performed over sliding windows of variable sizes (ranging from 5 frames to 60 frames). Feature vectors estimated in the various windows are resampled to 30 elements and sent to the trained classifier.

If a gesture is detected in a window and the related class probability is higher than the related threshold, the corresponding frames are assigned to the class.

4.5. System Configuration

The recognizer, developed using PyTorch and CUDA, has been trained and tested on a *Lenovo Legion* 5 PC with a RAM of 16 Gb, a *Nvidia RTX 2060 (6Gb)* graphics card and *AMD Ryzen 7 4800H(8 Cores)* processor and the classification of a single frame takes about 100ms.

5. Results

Table 2 summarizes the results for the different methods and runs. The performances are rather similar on average. The table shows that, overall, Group 2 with the Casual TCN method achieved the highest detection rate (0.80). However, Group 3 managed to reach the lowest amount of false positives with the 2ST-GCN 5F method with a ratio of 0.23. Looking at the Jaccard Index, Casual TCN is again the one with the best performance.

Major differences between the proposed methods appear when we look at the per-class performance metrics. In Figure 8 the bar charts show detection rates, Jaccard Indexes and false positives' scores for the single gesture classes. It can be noticed how no method is performing consistently across the entire gesture dictionary.

The inconsistency is quite noticeable in the false positives' scores (Figure 8c). In fact, methods like Casual TCN that have overall the highest Jaccard Index and overall FP rate in the average of the groups, when looking at the FP by-class, present near-to-zero FP for a large amount of gestures and an high FP spike for specific gestures such as CROSS and KNOB. The same behavior can be observed for the Stronger methods.

The analysis of the results grouped by gesture types can also be important to understand how the methods handle heterogeneous gestures where the semantics depends on completely different features. Figure 9 shows huge variability in the effectiveness of the methods for the different classes. For example Casual TCN and Stronger present good performances on static gestures, and have relevant false positive issues on coarse dynamic and periodic gestures. TN-FSM the most consistent detection rates across the categories, being, however, the worst for static gestures and presenting relevant false positives issues for fine-dynamic gestures, with a surprising exception for the TN-FSM+JD run.

The fact that in the contest's rules the detection of gestures is considered correct if the intersection with the annotated time frame is higher than half the length of the ground truth gesture duration should be considered with care. The threshold is, in fact, arbitrary, and some of the detected gestures are discarded by not meeting its requirement and are, therefore, not classified neither as correct nor as a false positive. We analyzed the effect on the detection rate when changing this threshold. Figure 10 shows the detection rate of the different methods as a function of it. As expected, a lower threshold would improve the scores of most methods, however, some methods benefit from a larger improvement compared to others (up to 15% for 2ST-GCN and Stronger+EN). With small thresholds, 2ST-FCN-5F is the method providing the best results and could be the best choice for an application task that doesn't require the accurate localization of the complete gesture frame. On the other hand, Causal TCN is the method less influenced by the threshold, meaning that it provides most of the detection with a large overlap with the ground truth frame.

(a) Detection rate

(b) Jaccard Index

(c) False Positives' score

Fig. 8: Performance metrics per class, averaged on all the test sequences.

The evaluation of the actual delay of the recognition with respect to the ground truth annotations of gestures' starts and end is summarized in Table 3 and represented in Figure 11 as a function of gesture types. Here the bars represent on the right (positive values) the average recognition delay with respect to the ground truth gesture start, measured in frames. In the negative part (left) they represent the advance with which they are recognized, with respect to the actual gesture end. Here we see consistent and very low delays for the Causal TCN and the first TN-FSM run, while 2ST-GCN is incredibly fast in detecting static gestures, but not on the other categories. We will discuss in detail these results in the next section.

6. Discussion

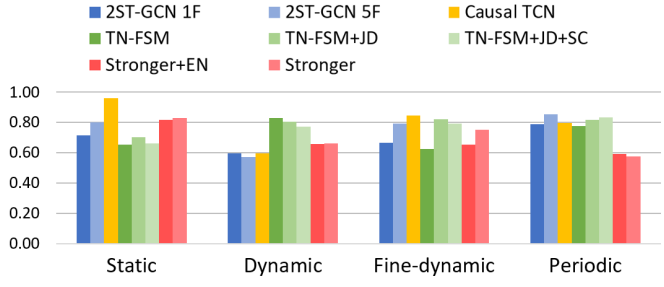
The outcomes of the SHREC '22 evaluation provide interesting insights on the feasibility of effective online recognition of heterogeneous gestures based on Hololens 2 streams or similar hand pose trackers data.

The proposed techniques employ some of the most popular network architectures: transformer networks, temporal convolutional networks (TCN), and graph convolutional networks (GCN). The methods are efficient, but the classification performances are still far from those required by usable, practical applications.

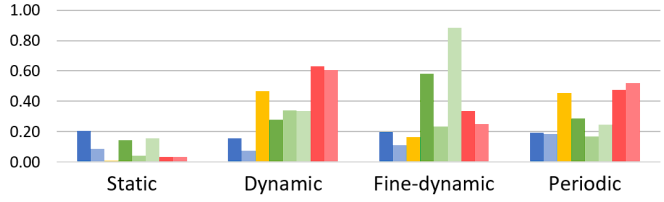
Detection rates are not exceeding 80%, meaning that a relevant percentage of the gestures is missed. This happens for all the methods and all the gestures types.

Group	Method	DR	FP	JI	Delay(fr.)	time(ms)
B.Line	Stronger+EN	0.70	0.35	0.56	16.40	100
	Stronger	0.72	0.34	0.59	14.79	100
G1	2ST-GCN 1F	0.68	0.33	0.52	12.55	**2.1**
	2ST-GCN 5F	0.74	**0.23**	0.61	13.28	**2.1**
G2	Causal TCN	**0.80**	0.29	**0.68**	19.00	28
G3	TN-FSM	0.73	0.34	0.56	**10.00**	4.65
	TN-FSM+JD	0.77	0.23	0.63	**10.00**	4.65
	TN-FSM+JD+SC	0.74	0.36	0.56	**10.00**	4.65

Table 2: The table shows the summary of the results for each group method and all the variants proposed averaged on all the gestures. The metrics displayed are the detection rate, the false positives' scores, the Jaccard Index and the delay (in frames) between the start of a gesture marked by the method and the last frame reached at the moment of the marking.



(a) Detection rate



(b) False positives

Fig. 9: Scores by gesture type

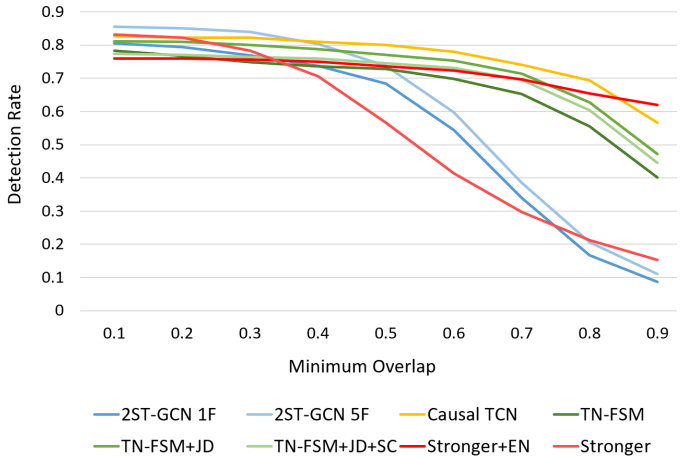


Fig. 10: The Detection Rate of each method as a function of the minimum overlap ratio between the gestures' windows, for the detected gesture to be marked as correct detection.

Group	Method	from start	from end
B.Line	Stronger+EN	10.72	-22.32
	Stronger	10.27	-22.13
G1	2ST-GCN 1F	7.85	-26.65
	2ST-GCN 5F	7.64	-22.90
G2	Causal TCN	**4.36**	**-28.79**
G3	TN-FSM	4.52	-28.63
	TN-FSM+JD	18.30	-14.73
	TN-FSM+JD+SC	10.54	-22.24

Table 3: Average delays in frames from ground truth start and end of annotated gestures derived by algorithm design (excluding computation time).

The number of false positives is definitely too high for all the proposed methods to support reliable interfaces. The accuracy in the detection is also not optimal. As the false positives score is the number of false positives divided by the number of real gestures, even the lowest value obtained, 0.1 is rather high. In our dataset, we have sequences of about 30 seconds with 4 gestures on average, which means that the best method proposed would detect 4 non-existing gestures in 5 minutes of continuous hand movements. The others methods 8 or more. These figures should be reduced.

It seems that the network architecture has little impact on the classification performances and differences seem to be mostly caused by the training and online testing procedures. Methods based on 1D convolutions (Causal TCN and STRONGER) perform best on static gestures Causal TCN performs well also on Dynamic fine and periodic gestures and has only lower scores on dynamic ones. This fact, however, does not seem to be related to the network architecture, but rather to the fact that the network is trained to recognize sequences of 20 frames, while dynamic gestures are longer. The use of more complex networks does not seem to help as transformers and graph networks are not providing much better results. However, the method based on GCN seems the only one to use raw data

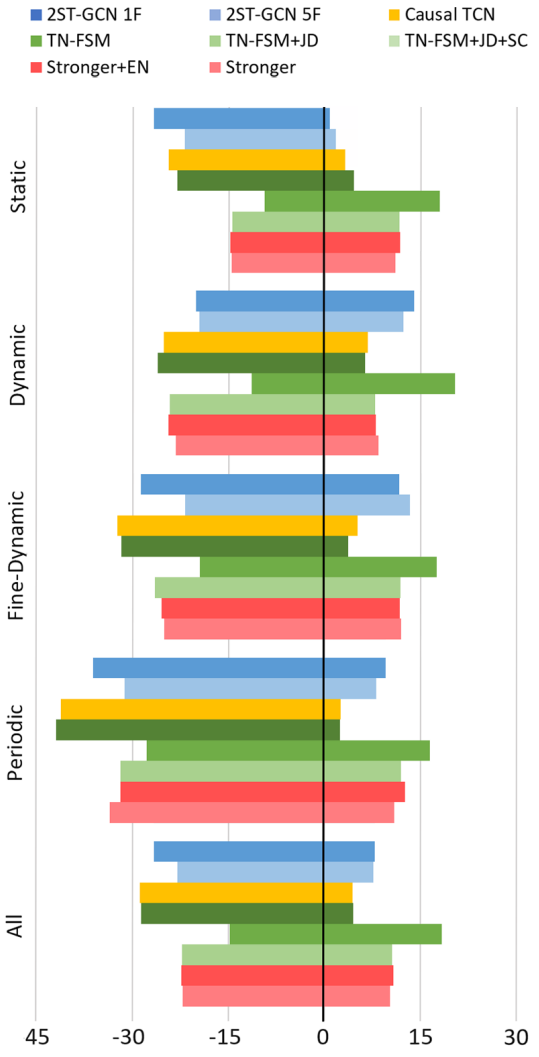


Fig. 11: The delay in frames of the last frame used by each method from the start and end marks of Ground Truth gestures, shown by gesture type. Chart bars are centered on the last frame used.

(coordinates as input), while the other networks process hand-crafted features. Some groups tested the same methods with different feature sets, but additional features do not always help, as in the case of energy for Stronger and the spherical coordinates for TN-FSM.

Gesture prediction times are, instead, negligible and definitely not an issue. The classification times of the methods proposed by Group 1 and Group 3 are almost negligible (2.1 *ms* and 4.65 *ms*) and the only method with a non-negligible computation time, albeit compatible with an online application is Stronger (100 *ms*). These values are small with respect to the intrinsic time delay depending on the algorithms' design and shown in Figure 11.

The analysis of these delays is indeed quite interesting. Some predictions are surprisingly fast and are provided after very few frames from the annotated starts. In some cases gesture prediction does anticipate the actual start of the gesture itself. Causal TCN and the first run of TN-FSM provide results with delays of less than 5 frames (250ms) consistently for all the gesture types. 2ST-GCN is the fastest method on static gestures even if not so efficient for the other classes. Dynamic-coarse and fine gestures are all recognized far before their completion, even if the semantics of the gesture depends on the whole palm and fingers trajectories. This shows that the methods learn to predict the complete gestures from the first frames only.

This early detection is allowed by the fact that classifiers like 2ST-GCN or Casual TCN are trained to recognize not entire gestures, but also smaller time frames including the first part of the gestures.

While this provides amazing response times, it could be, on the other hand, one of the reasons for the high number of false positives. For practical applications, more complex recognition strategies increasing the time delay but increasing detection accuracy and reducing false positives would be more effective.

7. Conclusion

In this paper we presented the benchmark proposed in the SHREC 2022 track on Online Recognition of Heterogeneous Gestures, and we reported and analyzed the results of the evaluation of the participants' submissions. The outcomes of our analysis are quite interesting, showing that the results are still far from being usable in a practical setting, but also indicating possible way to tackle the weaknesses of the methods. For this reason we believe that it is necessary to continue this evaluation on novel methods and improved recognition strategies. We will create a website where the data and the evaluation code will be available and it will be possible to submit novel methods. We will continuously update a leaderbord with the evaluation results.

8. Acknowledgments

Empty section for anonymous submission.

References

[1] Caputo, A, Giachetti, A, Soso, S, Pintani, D, D'Eusanio, A, Pini, S, et al. Shrec 2021: Skeleton-based hand gesture recognition in the wild. Computers & Graphics 2021;99:201–211.

[2] De Smedt, Q, Wannous, H, Vandeborre, JP, Guerry, J, Le Saux, B, Filliat, D. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In: 3DOR-10th Eurographics Workshop on 3D Object Retrieval. 2017, p. 1–6.

[3] de Smedt, Q, Wannous, H, Vandeborre, JP. Skeleton-based dynamic hand gesture recognition. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2016 IEEE Conference on. 2016, p. 1206–1214.

[4] Garcia-Hernando, G, Yuan, S, Baek, S, Kim, TK. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In: Proceedings of Computer Vision and Pattern Recognition (CVPR). 2018,.

[5] Caputo, FM, Burato, S, Pavan, G, Voillemin, T, Wannous, H, Vandeborre, JP, et al. Shrec 2019 track: online gesture recognition. In: Eurographics Workshop on 3D Object Retrieval. The Eurographics Association; 2019,.

[6] Wan, J, Zhao, Y, Zhou, S, Guyon, I, Escalera, S, Li, SZ. Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016, p. 56–64.

[7] Zhang, Y, Cao, C, Cheng, J, Lu, H. Egogesture: A new dataset and benchmark for egocentric hand gesture recognition. IEEE Transactions on Multimedia 2018;20(5):1038–1050.

[8] Emporio, M, Caputo, A, Giachetti, A. STRONGER: Simple TRajectory-based ONline GEsture Recognizer. In: Frosini, P, Giorgi, D, Melzi, S, Rodolà, E, editors. Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference. The Eurographics Association. ISBN 978-3-03868-165-6; 2021,doi:10.2312/stag.20211481.

[9] Yan, S, Xiong, Y, Lin, D. Spatial temporal graph convolutional networks for skeleton-based action recognition. CoRR 2018;abs/1801.07455. URL: http://arxiv.org/abs/1801.07455. arXiv:1801.07455.

[10] Thompson, EM, Biasotti, S, Giachetti, A, Tortorici, C, Werghi, N, Obeid, AS, et al. SHREC'20 track: Retrieval of digital surfaces with similar geometric reliefs. Computers and Graphics 2020;91:199–218. URL: https://hal.archives-ouvertes.fr/hal-02916788. doi:10.1016/j.cag.2020.07.011.

[11] Bai, S, Kolter, JZ, Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:180301271 2018;.

[12] Yang, F, Wu, Y, Sakti, S, Nakamura, S. Make skeleton-based action recognition model smaller, faster and better. In: Proceedings of the ACM multimedia asia. 2019, p. 1–6.

[13] Vaswani, A, Shazeer, N, Parmar, N, Uszkoreit, J, Jones, L, Gomez, AN, et al. Attention is all you need. In: Advances in neural information processing systems (NIPS). 2017, p. 5998–6008.

[14] Lin, TY, Goyal, P, Girshick, R, He, K, Dollár, P. Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 2980–2988.

[15] Loshchilov, I, Hutter, F. Decoupled weight decay regularization. arXiv preprint arXiv:171105101 2017;.