IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 1 December 2023, accepted 7 December 2023, date of publication 12 December 2023, date of current version 18 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3341702

RESEARCH ARTICLE

TY-Net: Transforming YOLO for Hand Gesture Recognition

ZORANA DOŽDOR, ZORAN KALAFATIĆ, (Member, IEEE), ŽELJKO BAN, (Member, IEEE), AND TOMISLAV HRKAĆ, (Member, IEEE)
Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Tomislav Hrkać (tomislav.hrkac@fer.hr)

This work was supported by the Operational Program from the European Structural and Investment Funds through the Development of an Advanced Electric Bicycles Charging Station for a Smart City Project under Grant KK.01.1.1.07.0066.

ABSTRACT Hand gesture recognition is a rapidly expanding field with diverse applications, and the use of skeleton-based methods is gaining popularity due to their potential for lightweight execution on embedded devices. However, ensuring robustness and accuracy in both gesture classification and temporal localization is critical for any gesture recognition system to be successful. In this paper, we propose a novel skeleton-based approach to online gesture recognition that draws inspiration from the YOLO object detection model. Specifically, we propose a transformer-based architecture for online gesture recognition that directly predicts both gesture classes and gesture boundaries from a sliding window input. The model is trained in an end-to-end manner using a proposed loss function that focuses on samples containing gesture centers and learns to predict boundaries in the temporal domain analogous to object centers and spatial bounding boxes in YOLO object detection. To evaluate the effectiveness of our method, we conduct experiments on two publicly available continuous hand gesture datasets: SHREC'22 and IPN Hand. Our results outperform the results of skeleton-based approaches from the SHREC'22 online gesture recognition contest. Moreover, our approach obtained competitive results compared to the approaches utilizing alternative input modalities on the IPN Hand dataset.

INDEX TERMS Online hand gesture recognition, hand skeleton, sliding window, transformer, deep learning.

I. INTRODUCTION

Computer-vision-based hand gesture recognition has the potential to become a non-invasive and intuitive alternative to traditional human-computer interaction devices. Although hand gesture control is often associated with virtual and augmented reality for immersive experiences, it has a wide range of applications in different industries such as robotics, the automotive industry, and medicine. However, to integrate hand gesture recognition technology into various devices and applications, it is crucial to ensure its robustness and low latency performance. Achieving these goals requires further research and development, something that this work aims to contribute to. Hand gesture recognition systems can be developed for different input modalities such as RGB, depth, and optical flow [1]. While most of the existing works use a combination of multiple input modalities to improve accuracy, our proposed approach relies solely on 3D hand

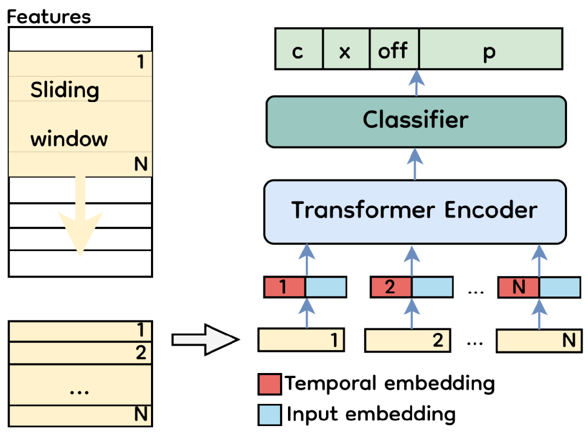

FIGURE 1. Overview of the proposed approach.

The associate editor coordinating the review of this manuscript and approving it for publication was Andrea Bottino.

skeleton input. The goal is to enable lightweight execution while still achieving a reasonable level of accuracy.

Most of the research so far has focused on achieving accurate classification of segmented gestures [2], [3], with limited consideration for the demands of real-world applications. These applications necessitate online, causal processing of continuous input data up to the current moment without access to future information. To address this challenge, various methodologies have emerged for recognizing gestures from continuous input data, broadly classified into two main strategies: two-stage and one-stage detectors.

The two-stage detection involves extracting candidate gesture segments from the input and then classifying them into one of the gesture classes. The upside of this approach is that pre-segmented gestures are relatively easy to classify. However, a lot of works rely on handcrafted heuristics to obtain temporal segmentation [4], [5]. While some recent studies have explored the use of deep learning models for predicting temporal boundaries [6], [7], [8], the overall system is not end-to-end trainable, which is a notable drawback because the errors of the detector propagate to the classifier. Furthermore, early prediction (activating the gesture before it ends) is not possible when using this type of approach since the detector requires frames that extend beyond the actual gesture endpoint in order to accurately determine its conclusion.

The one-stage approach involves modeling the input as a sliding window and predicting one of the gesture classes or a no-gesture class for the last time step of the input. By doing so, temporal segmentation is learned implicitly, simultaneously with classification. The predictions for each time step can be grouped and filtered out in a postprocessing stage, to obtain single-time activations. This approach is more general, with the added benefit of early detection being possible. However, single-time activation error can be significant even with high frame-wise accuracy. For example, it can happen that although most of the frames of a gesture are correctly classified, several consecutive frames become misclassified as non-gesture, leading to double activation of the same gesture. Additionally, training a model to predict gesture classes for the last timestep of a sliding window input is challenging due to ambiguities arising when prior frames provide little information for the final timestep, for instance when the window contains mostly background frames and the last timestep corresponds to the beginning of a gesture.

Approaching the problem of continuous gesture recognition can be compared to the task of object detection. Prior to YOLO [9], most object detection methods employed a two-stage approach, which involved proposing regions of interest and subsequently classifying objects within those regions. However, YOLO introduced a groundbreaking single-stage approach, which enabled the prediction of object locations and classes in a single forward pass, leading to real-time detection with high accuracy. Motivated by this, we propose a loss function for gesture recognition inspired by YOLO to enable explicit learning of both gesture boundaries and classes in a one-stage approach.

We reframe the gesture recognition problem with a model that takes a sliding window input and learns whether it contains the center of the gesture and, if so, what are the boundaries and the class of that gesture. This way, we kept the end-to-end trainability and early detection capability of the existing one-stage approach, while making learning and postprocessing more robust by focusing on samples that contain a significant portion of the gesture (i.e. gesture center) and predicting the temporal boundaries for those samples instead of predicting gesture classes for the last timestep of a sliding window. It is worth noting that in certain applications, the information regarding gesture boundaries may not be necessary. In such cases, a simpler model can be constructed, as our system does not utilise this information for gesture activation.

In recent years, transformers have emerged as a popular choice for various tasks, including online action recognition from video input due to their effectiveness in modeling long-range temporal dependencies [10], [11], [12]. Therefore, we propose the utilisation of a transformer encoder as a feature extractor for skeleton-based input to help the model focus on important parts of the input.

To sum up, our contributions are as follows:

- a model with transformer encoder feature extractor for online gesture recognition that predicts both gesture boundaries and gesture classes;
- a YOLO-inspired loss function for the training of the model that enhances robustness by focusing on samples containing the gesture center and by learning to predict gesture boundaries.

The proposed system achieves state-of-the-art results on two continuous hand gesture datasets.

II. RELATED WORK

A. SEGMENTED HAND GESTURE CLASSIFICATION

Classification of pre-segmented gestures is a simpler task than continuous gesture recognition, but it offers valuable insight into feature extraction architectures. As the choice of model architecture can depend on the input modality, the focus here is on works utilising skeleton input modality.

Various techniques have been developed using convolutional neural networks (CNNs) and recurrent neural networks (RNNs). In [13], an LSTM architecture that incorporates global and finger motion features is presented. However, GRU-based architectures have shown better results with faster training times [14], [15]. In [16], a multi-channel CNN that extracts temporal features for each hand joint individually is proposed. However, all of the aforementioned approaches disregard spatial connections between the joints. To address this issue, authors of [17] have proposed a residual temporal convolutional network (TCN) with an attention branch that can extract both spatial and temporal features from skeleton sequence input. Some models combine CNN and RNN to extract spatial and temporal features. For instance, in [18], a combination of CNN and LSTM networks is presented for skeleton input. More recently, [19] proposed a hybrid approach combining 3D-CNN with a transformer network. The CNN part extracts high-level semantic skeleton embeddings, while the transformer network is responsible for capturing long-range temporal dependencies in skeleton sequence.

Additionally, attempts have been made to classify spatiotemporal joint trajectories by condensing them into an image and using a CNN for classification [20], [21], [22], [23], [24]. Graph-based architectures that encode spatiotemporal connections have also been developed, with some of them using predefined graph structures based on the natural connectivity of joints [25]. Others learn action-specific graph representations through self-attention mechanisms in spatial and temporal domains [26].

Finally, some works combine skeleton input with other input modalities to improve accuracy. Recently, the authors of [27] proposed combining a lightweight spatial vision transformer, bilinear pooling, and attention network for human action recognition. They use two-stream feature pooling and a fusion mechanism to combine RGB frames with skeleton information.

B. CONTINOUS HAND GESTURE RECOGNITION

1) TWO-STAGE RECOGNITION

Temporal segmentation in two-stage approaches is often dataset-specific or has underlying assumptions about the gesture execution. For example, some approaches use hand position or quantity of movement to segment gestures [4], [5]. Deep learning approaches have also been proposed, such as a bidirectional LSTM network for predicting gesture occurrence in each time step of the input [6] and a modified ResC3D network for predicting gesture segments [7]. Another approach, inspired by an object detection technique, uses an R-CNN architecture for hand gesture recognition [28]. Candidate gesture segments are first extracted by an energy-based gesture proposal module. Those candidates are then classified by a simple classification model and temporally extended and served as input to a larger model. In addition to classification, the larger model also refines estimated gesture boundaries. To obtain the final prediction, the two model outputs are compared. Specifically, any predictions where the predicted class differs between the two models or where the refined gesture boundaries do not overlap with the proposed gesture segment are discarded. While the inspiration for that approach is somewhat similar to ours, it is not end-to-end trainable and applies heuristics for the extraction of gesture segments.

2) ONE-STAGE RECOGNITION

The issue of early gesture recognition (activating the gesture before it ends), which is essential for real-world applications, was addressed in [29], [30], and [31]. The authors of [31] proposed an architecture that combines 3D CNN and LSTM for gesture recognition. The model divided continuous input video into segments and assigned a label to a segment when the predicted probability reached a predetermined threshold. The model was trained using the Connectionist Temporal Classification (CTC) loss to identify the nucleus of the gesture while assigning a no-gesture category to the remaining clips. To address the problem of single-time activations, the authors of [32] proposed a two-model architecture for RGB video input consisting of a lightweight 3D CNN model for gesture segmentation and a more complex 3D CNN model for gesture classification. The models processed sliding window inputs, with the classifier activated only when a gesture was detected. Per-frame predictions are obtained and weighted using a weighted average scheme, with lower weights assigned to earlier predictions. The single-time activations occur when a confidence level reaches a selected threshold for early detection or when the gesture ends for late detection. Both models were trained using cross-entropy loss. To mitigate segmentation errors caused by optimizing the model with only classification loss, the authors of [33] proposed a smoothing loss function.

C. TEMPORAL ACTION DETECTION

Temporal action detection (TAD) and continuous gesture recognition share a common goal of predicting class and temporal boundaries of actions or gestures from untrimmed input. However, aside from the differing problem domains, hand gesture recognition requires real-time prediction without having access to the entire gesture, while action recognition typically has the complete video sequence available. Consequently, the training and inference strategies for the two tasks diverge. Nevertheless, various works in TAD draw inspiration from object detection and show the relevance of adapting object detection techniques for the temporal domain.

For example, Faster R-CNN has been adjusted for temporal action localization in [34]. A feature map is first extracted from the input video segment. A segment proposal network then predicts a set of segment proposals, and for each segment proposal, a Deep Neural Network (DNN) predicts class and refined boundaries. Another approach proposed in [35] involves training multiple feature extraction networks followed by a Single Shot Temporal Action Detection network in which there are several prediction layers connected to feature maps of different scales. Each prediction layer is connected to one anchor instance and predicts the categories and location offsets of that anchor instance. In [36], two semantically-constrained Gated Recurrent Unit (GRU) based modules are utilised for feature extraction, outputs of which are then combined to obtain the final detection output. The modules are semantically-constrained to learn features relevant for temporal proposals and classifications.

More recently, [37] proposed a neuro-heuristic approach to fast and robust detection of unusual activities in surveilance videos, designed for use on limited IoT device hardware. The video frames are first analyzed by fast and computationally inexpensive heuristic algorithm. Only those parts of the video that are estimated as unusual by the heuristic algorithm are sent to a more computationally demanding deep neural network to further classify the type of the action.

III. PROPOSED APPROACH

An overview of the proposed system is presented in Fig. 1. A sliding window technique is adopted to model a continuous input sequence, where the input is divided into a series of overlapping windows of fixed length. These windows are then used as input to the model in a sequential manner.

To obtain features for each window, the 3D coordinates of hand keypoints are flattened and concatenated, along with additional features derived from them. These additional features include pointwise velocities of each skeleton keypoint and their pairwise Manhattan distances. Velocity is included to account for variations in the speed at which gestures are performed, while Manhattan distance is included to introduce spatial information.

Per-axis velocity of joint i at time t is obtained using finite differences to approximate its position derivative, resulting in velocities $v_{i,x}$, $v_{i,y}$, and $v_{i,z}$ (Equations 1a to 1c). The Manhattan distance between joint pair i and j at time t is computed as the sum of their absolute position differences along the x, y, and z axes, denoted as $d_{i,j,t}$ (Equation 2).

$$v_{i,x} = x_{i,t} - x_{i,t-1}, \tag{1a}$$
$$v_{i,y} = y_{i,t} - y_{i,t-1}, \tag{1b}$$
$$v_{i,z} = z_{i,t} - z_{i,t-1}. \tag{1c}$$
$$d_{i,j,t} = |x_{i,t} - x_{j,t}| + |y_{i,t} - y_{j,t}| + |z_{i,t} - z_{j,t}|. \tag{2}$$

The input is first embedded by passing through a fully connected layer and combined with temporal embeddings by concatenation. The temporal embedding layer transforms discrete time steps into vector representations that are learned during the training process of the model. The combined dimension of both embeddings is 512 (each is 256). The resulting vector is then fed into the Transformer encoder, which applies a self-attention mechanism to capture the relationships among the time steps in each sliding window. The Transformer encoder consists of multiple layers, each constructed with two key components: a multi-head self-attention mechanism and a position-wise feed-forward network (Fig. 2). Given an input tensor, $X \in \mathbb{R}^{t \times d}$, where t represents sequence length and d is input dimension, self-attention mechanism is expressed as [38]:

$$Q, K, V = XW^Q, XW^K, XW^V \tag{3a}$$
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{3b}$$

Here, Q, K, V correspond to the query, key, and value vectors, which are obtained by applying the weight matrices W^Q, W^K, W^V to the input tensor X, while d_k is a hyperparameter representing the dimensionality of query and key vectors. The multi-head self-attention operation is defined as [38]:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) \tag{4a}$$
$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4b}$$

where h represents the number of attention heads, and W^O denotes a trainable weight matrix.

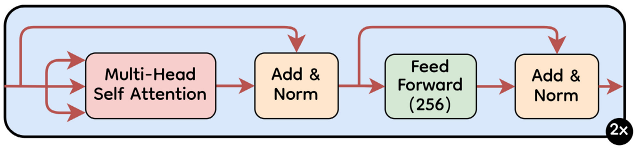

FIGURE 2. Transformer encoder in TY-Net.

Experimental evaluation has shown that a two-layer Transformer encoder with four self-attention heads and a hidden layer size of 256 achieves high performance. Finally, the output of the last time step of the Transformer encoder is passed through a fully connected classifier to predict the confidence (c), the center of the gesture (x), offset (*off*) and a vector of class probabilities ($\mathbf{p}$). In an online setting, we adopt a gesture detection strategy that accumulates predictions with confidence scores greater than 0.5 across time. Specifically, when the model predicts the same class for T timesteps, the gesture is activated, and the remaining predictions for that gesture are ignored (Algorithm 1). This approach allows the system to detect a gesture even before it ends and to calculate the gesture boundaries based on the average of T accumulated predictions. For a timestep t, a gesture G will be activated as follows:

$$G_t = \begin{cases} 1 & \text{if } \sum_{i=(ta+1)}^{t} \delta(C_i = G \;\&\; c_i > 0.5) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Here, *ta* represents the end of the last activated gesture or zero if no gesture has been activated yet, δ is an indicator function equal to one when the condition is satisfied and C_i is the predicted gesture class.

A. LOSS

The problem formulation and consequently the loss function in our approach is inspired by the YOLO model for object detection. The sliding window input of our model corresponds to the image cells in YOLO. Unlike YOLO's non-overlapping cells, our sliding window input is overlapping. This is motivated by the unconstrained nature of the input, enabling the model to handle gestures that extend beyond a single sliding window input more effectively. Moreover, for our online problem setting, non-overlapping cells would introduce undesirable latency. The primary objective of our model is to detect whether a given window contains the center of a gesture. If a gesture center is detected, the model will predict the center location, the gesture length, and the class of the gesture. The center location is predicted relative to the

Algorithm 1 Online Gesture Recognition

Input: Continuous skeleton sequence, Trained model
Output: Activated gestures

```
accum ← []
gest_active ← $False$
consecutive_non ← 0
for each sliding_window in skeleton_sequence do
    input ← prepare_input(sliding_window)
    $c$, $x$, $off$, $class$ ← model.predict(input)
    if gest_active then
        if $c < 0.5$ or $class \neq$ pred_gesture then
            consecutive_non ← consecutive_non + 1
            if consecutive_non ≥ thresh_non then
                gest_active ← $False$
                consecutive_non ← 0
            end if
        else
            consecutive_non ← 0
        end if
    end if
    if gest_active = $False$ and $c > 0.5$ then
        accum.add($off$, $x$, $class$)
    end if
    if count_most_frequent_class(accum) >= $T$ then
        pred_gesture ← most_frequent_class(accum)
        avg_start ← calculate_avg_start(accum)
        avg_end ← calculate_avg_end(accum)
        activate(pred_gesture, avg_start, avg_end)
        gest_active ← $True$
        accum ← []
    end if
end for
```

sliding window length, while the gesture length is predicted as an offset from the center in both temporal directions. This offset value is relative to a pre-determined prior value.

The proposed model takes an input sequence $X = \{x_t\}_{t=0}^{N}$, where N is the sliding window length and x_t is the t-th feature vector, and outputs a vector of four groups of components $(B* c, x, B* off, \boldsymbol{p})$, where B is the number of offset priors. Given the output vector $\mathbf{o} \in \mathbb{R}^d$, obtained by the TY-Net classifier, the predictions can be written as:

$$index = \text{argmax}(\sigma(\mathbf{o}[0:B])) \tag{6a}$$
$$c = \max(\sigma(\mathbf{o}[0:B])) \tag{6b}$$
$$x = \sigma(\mathbf{o}[B]) \tag{6c}$$
$$off = \exp(\mathbf{o}[(B+1+index)]) \tag{6d}$$
$$\mathbf{p} = \text{softmax}(\mathbf{o}[(2*B+1):]) \tag{6e}$$

where σ is sigmoid function and exp is exponential function. The model predicts B confidence scores and offsets to enable having more than one offset prior. However, only the prior with the length closest to the ground truth length is penalized by the loss function, which encourages each prior to specialize for certain gesture lengths, making the model more robust in predicting various gesture lengths. The ground truth confidence c for prior B is set to one given that the window contains a gesture center and the prior B is the closest to ground truth length. Otherwise, c is set to zero.

The loss function for a single example (one position of the sliding window) is given by equation 7, which consists of four terms. The first two terms are confidence losses responsible for learning which input samples contain the gesture center and which do not. The third term corresponds to the localization loss, which penalizes incorrect predictions of the gesture center and the gesture length. Since the center location is predicted relative to the sliding window length and has values between zero and one, it is activated by a sigmoid function. The offset from the center is passed through an exponential activation function. The combination of the center and offset enables the calculation of predicted gesture boundaries as the sum and difference of the center value and the offset value. The optimization process implicitly learns the center and offset by optimizing the intersection over union between the ground truth and the predicted boundaries. The final term is the classification loss, which is computed as cross-entropy between the predicted and ground truth probabilities.

$$\begin{aligned} L = W_{sw}(&\lambda_g 1^g \sum_{i=1}^{B} \text{BCE}(c_i, \hat{c}_i) \\ &+ \lambda_{nog} 1^{nog} \sum_{i=1}^{B} \text{BCE}(c_i, \hat{c}_i) + \sum_{i=1}^{B} 1_i^g (1 - \text{IoU}(GT_b, P_b)) \\ &+ 1^g \text{CE}(\boldsymbol{p}, \hat{\boldsymbol{p}})) \end{aligned} \tag{7}$$

Terms BCE and CE correspond to binary cross-entropy and cross-entropy, respectively. The indicator function 1^g is equal to 1 if the window contains the gesture center. Additionally, subscript i in 1_i^g denotes that i-th gesture length prior is the one best matching the ground truth gesture length. The indicator function 1^{nog} is equal to 1 if the window does not contain the gesture center and is equal to $1 - 1^g$. The weighting factors λ_{nog} and λ_g are used to adjust the contribution of each term in the loss function. Specifically, similar to the observation in [9] on object detection, many samples do not contain gesture center, so λ_{nog} and λ_g are introduced to compensate for that imbalance.

Furthermore, we introduce a weighting scheme that takes into account the position of a gesture within the sliding window, expressed by the factor W_{sw}. This scheme considers three potential scenarios.

First, when the gesture center falls within the sliding window while the gesture end extends beyond it, the model has observed the gesture center but not its completion (Fig. 3b). Since the primary objective of the loss function is to detect the gesture within this time frame, the weight assigned in this case is one.

Second, if the gesture start is within the sliding window but the gesture center is not, the model has observed a portion of the initial phase of the gesture but not its center (Fig. 3a). According to the objective, the model should predict zero confidence (c) in this case, indicating the absence of a gesture. However, the punishment for a positive prediction is adjusted inversely proportional to the percentage of the gesture length within the sliding window. This adjustment is calculated using the following equation:

$$W_{sw} = 1 - (t - S_g)/(E_g - S_g) \tag{8}$$

Here, t represents the last timestep of the sliding window, while S_g and E_g denote the start and end times of the gesture, respectively.

Third, when both the gesture center and the gesture end are within the sliding window, it indicates that the model has observed the entire gesture (Fig. 3c). As the objective is to enable early prediction, the loss is weighted based on the time elapsed since the gesture end. The weight calculation is determined by the following equation:

$$W_{sw} = 1 - \min((t - E_g)/(E_g - S_g), 1) \tag{9}$$

In this equation, the term minimum ensures that the weight does not become negative, even if the time elapsed from the gesture end is greater than the duration of the gesture itself.

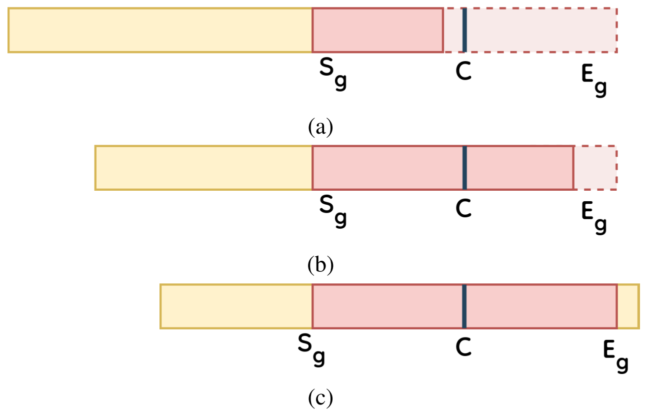

FIGURE 3. Position of a gesture (highlighted in red) within a sliding window (highlighted in yellow). S_g, E_g and C denote the gesture start, gesture end and center of the gesture, respectively. Cases include the presence of only the gesture start (a), both the gesture start and center (b) and the entire gesture (c) within the sliding window.

IV. EXPERIMENTS

A. DATASETS

Obtaining hand gesture datasets that include continuous, unsegmented gestures with hand skeleton input modality can be challenging. In fact, to our knowledge, there is only one such dataset currently available - the SHREC gesture dataset. Furthermore, there is generally a lack of unsegmented hand gesture datasets for any type of input modality. Our evaluation of hand gesture recognition was conducted using two publicly available datasets: the SHREC'22 Hand Gesture Dataset and the IPN Hand dataset.

The SHREC'22 Hand Gesture Dataset [28] is one of several SHREC datasets that have been proposed as a part of the annual 3D Shape Retrieval Challenge. The most recent dataset, proposed in 2022, consists of 16 different static and dynamic gestures in the domain of human-computer interaction (Fig. 5), with a total of 1152 samples across 288 sequences. This dataset was captured using a Hololens2 device and includes only 3D hand skeleton information. The dataset can be downloaded at [40].

The IPN Hand dataset [39] contains 4000 gesture samples for 13 different static and dynamic gestures (Fig. 4), also designed for human-computer interaction. Each RGB video in the IPN Hand dataset is a series of continuously captured gestures with several randomly positioned breaks. The samples in this dataset were collected with variations in background and illumination. In addition to the RGB videos, the dataset also provides optical flow and hand segmentation modality. To evaluate our approach on the IPN Hand dataset, we extracted hand skeleton input modality from the RGB video using MediaPipe Hands [41]. The dataset can be downloaded at [42].

B. EVALUATION

The SHREC Hand Gesture Dataset authors utilised the Jaccard index as an evaluation metric for temporal segmentation. It is used in several online hand gesture recognition challenges [43], [44]. The Jaccard index measures the degree of overlap between the predicted and ground truth boundaries of gestures [45]:

$$JI_{s,i} = \frac{GT_{s,i} \cap P_{s,i}}{GT_{s,i} \cup P_{s,i}}, \tag{10}$$

where $GT_{s,i}$ and $P_{s,i}$ are ground truth and prediction binary vectors for sequence s. Vector values are zero or one, depending on whether the gesture i is being performed for a certain timestep or not.

The authors also computed the detection rate and false positive rate. In the calculation of the detection rate, a gesture is considered correctly detected when the overlap between ground truth and prediction boundaries is above the selected threshold and when their respective labels match.

Authors of the IPN Hand have utilised Levenshtein accuracy [31] as a metric for evaluation. It is obtained by dividing the Levenshtein distance [46] by the total number of gestures in the ground truth sequence. Levenshtein distance measures the dissimilarity between two sequences by counting the number of item-level changes (insertion, deletion, or substitutions) required to transform one sequence into the other. In the case of a continuous video sequence, each sequence is a string of numbers representing gesture classes, without explicit information about the start and end of each gesture (e.g. Levenshtein distance for sequences [1,2,3,4,5] and [1,6,3,4] is 2). The metric is given by the equation:

$$L_{acc} = (1 - (L_d(gt, p)/N_s)) * 100, \tag{11}$$

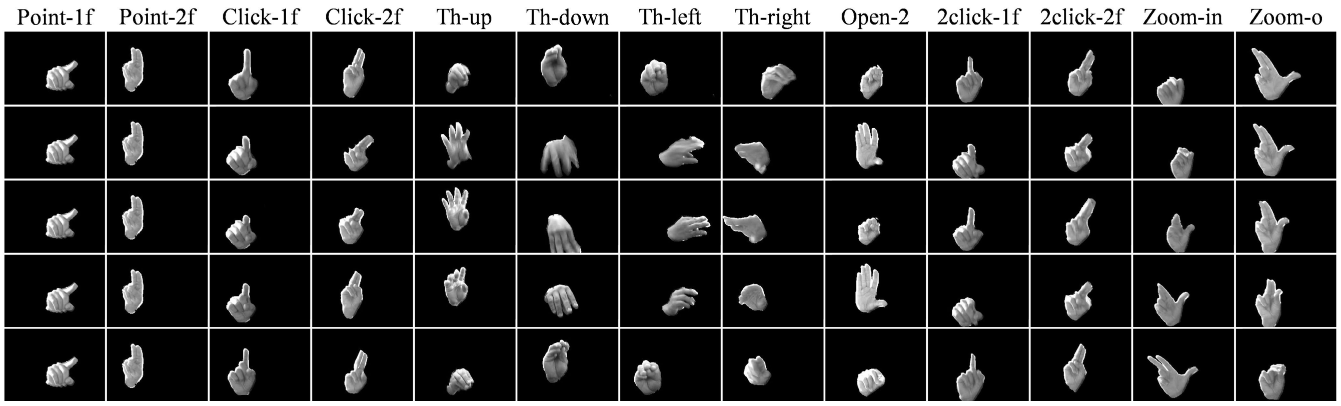

FIGURE 4. **Gesture classes in IPN Hand dataset [39]. The rows display the temporal order, arranged from top to bottom.**

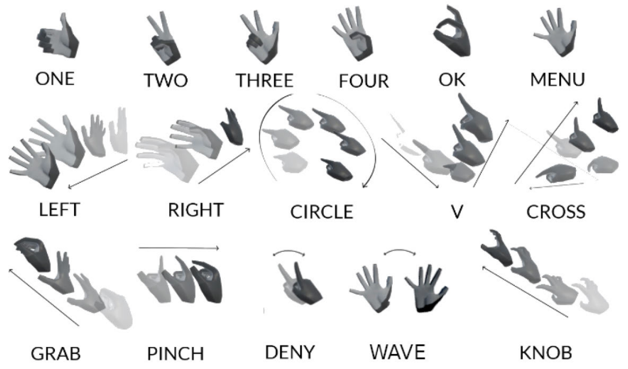

FIGURE 5. **Gesture classes in SHREC'22 dataset (first row - static, second row - dynamic coarse, third row left - dynamic fine, third row right - periodic) [28].**

where L_d is Levenshtein distance, and can be expressed as:

$$L_d(a,b) = \begin{cases} \max(|a|,|b|), & \text{if } \min(|a|,|b|) = 0 \\ L_d(\text{tail}(a), \text{tail}(b)), & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} L_d(\text{tail}(a), \text{tail}(b)), \\ L_d(\text{tail}(a), b), \\ L_d(a, \text{tail}(b)) \end{cases} & \text{otherwise} \end{cases} \tag{12}$$

Here $|x|$ denotes the length of the sequence x, tail(x) represents x without the first character and $x[n]$ is the nth character in x. Additionally, gt and p represent ground truth and predicted sequences, respectively, and N_s is the total number of gestures in the ground truth sequence.

C. TRAINING

The model was trained using the Adam optimizer. The warm startup was employed at the beginning of training by initializing the learning rate to 10^{-9} and increasing it by a factor of 10 for 5 epochs. Subsequently, the learning rate was set to 10^{-4} and reduced by a factor of 10 after 30 and 50 epochs, for a total of 60 epochs. An overview of the training procedure is given in the Algorithm 2.

To prepare the training data for both datasets, a random stratified split was used to divide it into training and validation sets in an 80:20 ratio. Additionally, data augmentation techniques were applied to the data. For each axis (x, y, z) of the input sample, a random number between -0.1 and 0.1 was added to the joint coordinates in a randomly selected segment of the sequence, with a probability of 10%. The length of the modified segment was randomly determined, ranging from 0 to 25% of the input length, and the position of the modified segment in the input sequence was also randomly selected.

D. HYPERPARAMETER SELECTION

1) SLIDING WINDOW LENGTH

We performed 4-fold cross-validation on both datasets to determine the optimal sliding window length. We assessed five different values starting from the mean gesture length of the training dataset: {*mean-15, mean, mean+15, mean+30, mean+45*}. The results on the SHREC'22 gesture dataset, shown in Figure 6a, indicate that the sliding window length of 52 (*mean+15*) yields the highest mean Jaccard index. Therefore, we selected a sliding window length of 52 for all subsequent experiments on this dataset. Regarding the IPN Hand dataset, we observed that the best result was obtained with a sliding window length of 175 (*mean+30*), as depicted in Figure 6b.

2) TIMESTEP THRESHOLD *T*

The hyperparameter T plays a crucial role in balancing the robustness and latency of the system. To summarize, if the model predicts the same class for T timesteps, it triggers the gesture activation and disregards any subsequent predictions for that particular gesture. Increasing the value of T results in higher latency and reduction of the detection rate while effectively reducing the number of false positives. To determine the optimal value of T, we conducted 4-fold cross-validation on both datasets. We evaluated T using a

Algorithm 2 Offline Training Algorithm

Input: Training data, Model with trainable parameters θ, Loss function, Optimiser, Learning rate scheduler, n_epochs
Output: Trained model

```
valid_loss_min ← ∞
for epoch in n_epochs do
    for input, labels in train_loader do
        output ← model(input)
        loss ← loss_function(output, labels)
        ∇loss ← compute_gradients(loss, θ)
        optimiser.update(θ, ∇loss)
    end for
    Update learning rate using scheduler
    valid_loss ← 0.0
    for input, labels in valid_loader do
        output ← model(input)
        loss ← criterion(output, labels)
        valid_loss ← valid_loss + loss
    end for
    if (valid_loss < valid_loss_min) then
        valid_loss_min ← valid_loss
        Save model
    end if
end for
```

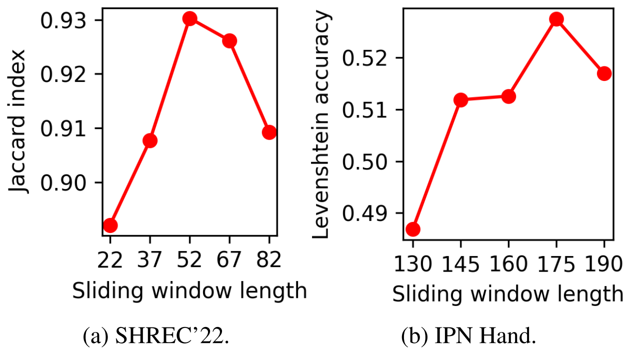

FIGURE 6. Selecting sliding window length for (a) SHREC'22 and (b) IPN Hand dataset.

range of values: {10, 20, 30, 40, 50}. On the SHREC'22 dataset, the optimal value of T was found to be 20, while for the IPN Hand dataset, the optimal value of T was determined to be 40 as visible in Fig. 7.

3) WEIGHTS λ_G AND λ_{NOG}

The values of λ_g and λ_{nog} are determined based on the number of priors, denoted as B. Specifically, λ_{nog} is set to $1/B$, as each of the B confidence outputs should be close to zero when no gesture center is present. The losses corresponding to those values sum up (cf. Equation 7) so that the *no-gesture* loss is increasing with the number of priors. Therefore it has to be scaled accordingly.

On the other hand, λ_g is set to $(1 + B)$. This choice is motivated by the fact that as B increases, each confidence

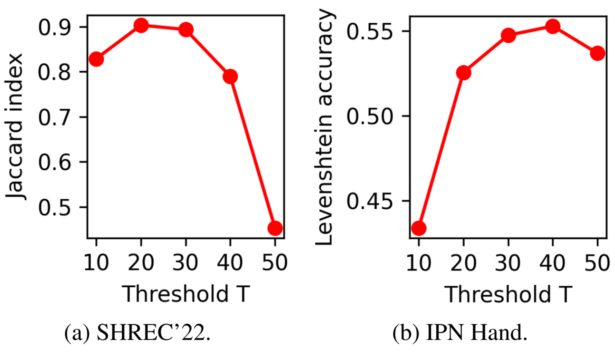

FIGURE 7. Selecting threshold T for (a) SHREC'22 and (b) IPN Hand dataset.

predictor has fewer samples to learn from. Therefore, it is desirable to increase the weights assigned to these samples to prioritize their influence on the training process. The addition of 1 ensures that λ_g remains larger than λ_{nog} in all cases, maintaining the relative importance of gesture-related predictions compared to the absence of gestures.

4) OFFSET PRIORS

An effective initialization of offset priors helps accelerate the learning process and improve the accuracy of gesture length estimation. Following [47], we utilised a k-means clustering algorithm on the training data to determine the appropriate prior values for offset. Instead of the Euclidean distance, we applied the following distance function:

$$d = 1 - \mathrm{IoU}(GT_b, P_b), \tag{13}$$

where GT_b and P_b are ground truth and predicted gesture boundaries.

To determine the optimal number of priors for each dataset, we analyzed histograms of the training data gesture lengths (see Fig. 8 and Fig. 9). The histogram of the IPN Hand dataset exhibits two distinct peaks, indicating a bimodal distribution. Consequently, we selected 2 priors to capture the underlying modes of this dataset. On the other hand, the histogram of the SHREC'22 dataset appears to follow an unimodal distribution. Thus, we determined that a single prior would be sufficient to capture the general distribution pattern of this dataset.

V. RESULTS

A. METHOD VALIDATION

To validate the proposed problem formulation, we compared it against a conventional one-stage approach. Specifically, we modified the objective to only predict the correct gesture class (and not the gesture boundaries) for the last timestep of the sliding window input. The number of model outputs corresponds to the number of gesture classes plus one for the no-gesture class, and the model is optimized using a cross-entropy loss. We maintained all training strategies across both approaches but implemented a modified evaluation strategy due to the different problem formulation. As mentioned,

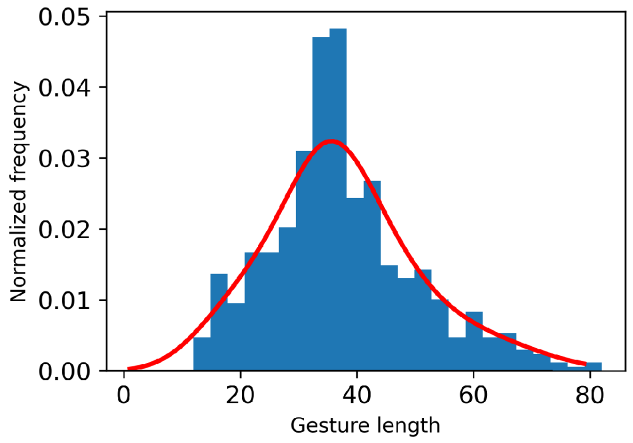

FIGURE 8. Histogram of SHREC'22 training data gesture lengths.

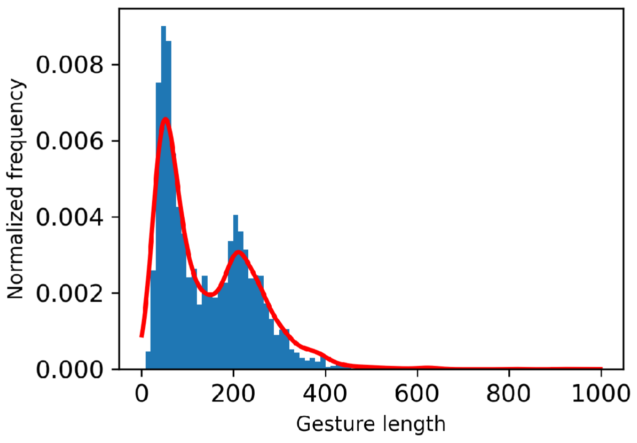

FIGURE 9. Histogram of IPN Hand training data gesture lengths.

a postprocessing stage is needed to estimate the gesture boundaries in this approach. To keep the evaluation similar to the proposed approach, predictions are accumulated and a gesture is activated when the model predicts the same class for T timesteps. From this, the beginning of the gesture is determined as the first timestep of the T accumulated predictions. For a fair comparison, we evaluated the one-stage approach using a set of T values:$\{10, 20, 30, 40, 50\}$. We report the best achieved result, specifically setting T to 10 for SHREC'22 and 20 for the IPNHand dataset. Unlike the proposed approach (TY-Net), we retained the remaining predictions of a gesture to determine its end. The end of a gesture is determined when the model predicted either a no-gesture class or a different class from the one predicted at the beginning of the gesture for a fixed number of consecutive steps E. We conducted experiments by varying the parameter E within the range of 1 to 15 but found that its impact on the results was minimal. Therefore, we chose a value of 5 for E.

Table 1 shows the comparison of the results obtained by the proposed approach and the above-described (conventional) approach on SHREC'22 and IPN Hand datasets. Our results demonstrate the effectiveness of the proposed approach relative to the conventional approach.

Additionally, the two approaches were compared on the IPN Hand dataset using the Levenshtein accuracy. The Levenshtein accuracy on the test set achieved by the conventional one-stage approach was 52.80%, whereas our proposed approach achieved a significantly higher accuracy of 60.46%.

TABLE 1. Comparison of our results with the conventional one-stage approach. The result are averaged over 5 random seeds.

Dataset	Approach	Detection rate (%)	False positives (%)	Jaccard Index (%)
SHREC'22	One-stage	83.5 ±2.0	5.0 ±3.0	79.6 ±2.0
SHREC'22	TY-Net	90.2 ±1.2	4.4 ±0.8	86.6 ±1.1
IPN Hand	One-stage	57.9±1.2	36.6±0.5	43.4±0.9
IPN Hand	TY-Net	75.4±1.1	30.2±1.3	61.7±1.4

B. STATE-OF-THE-ART COMPARISON

Table 2 presents a comparison of our results with state-of-the-art results on the IPN Hand dataset. Our proposed skeleton-based approach is compared to the approaches utilising various input modalities, where *Seg*, *Flow* and *TD* represent hand segmentation, optical flow and temporal difference, respectively. Temporal difference modality is proposed in [48] and it involves retrieving information from binarized absolute differences of RGB video frames. In [49], the authors extracted hand keypoints using MediaPipe from RGB video and employed the hand skeleton as the sole input modality, which is similar to our approach. Nevertheless, our method achieved significantly higher accuracy than theirs. The authors of [50] proposed a multi-modal fusion block to utilise features of multiple modalities and obtained the highest accuracy. They applied a one-step approach, but modified cross-entropy loss by the addition of smoothing loss and mid-point loss to alleviate the segmentation errors. However, they did not specify how they obtained single-time activations. The other approach with higher accuracy than ours is [48], but only when using temporal difference as an input modality, which suggests temporal difference could provide valuable information for gesture recognition.

The comparison of our results with SHREC'22 challenge contestants is presented in Table 3. Our approach demonstrated superior performance across all metrics, particularly in terms of false positives. Figure 10 shows the example of the results obtained by the proposed approach on one sequence of the IPN Hand dataset as well as the SHREC'22 dataset. As visible on the SHREC'22 sample sequence, the gesture is activated (red vertical line) before the real gesture end (green vertical line) for most of the gestures. These annotations were omitted on the IPN Hand sample due to compromised readability with a large number of gestures. We can observe from Tables 1 and 4 that the results on the SHREC 22 dataset are significantly better than those on the

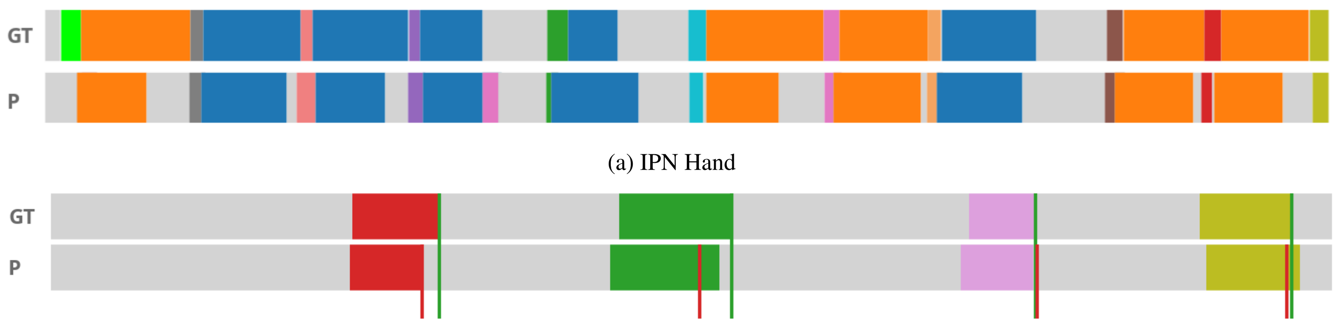

FIGURE 10. Example of ground truth (GT) and predicted (P) gestures for one sequence on (a) IPN Hand and (b) SHREC'22 dataset (different colors represent different gesture classes, gray denotes a no-gesture class).

TABLE 2. Overview of results on IPN Hand dataset. Our results are averaged over 5 random seeds (RS) and 5-fold cross-validation (CV).

Approach	Input modality	Levenshtein accuracy
[39]	RGB	25.3
[48]	RGB	**44.3**
[39]	RGB+Seg	39.0
[48]	RGB+Seg	**51.4**
[39]	RGB+Flow	42.5
[48]	RGB+Flow	48.7
[50]	RGB+Flow	**68.1**
[48]	Flow+Seg	49.3
[48]	Flow+Seg+RGB	56.5
[48]	Flow+Seg+TD	**62.2**
[49]	Skeleton	40.1
TY-Net (RS)	Skeleton	**60.5** ±0.2
TY-Net (CV)	Skeleton	**60.9** ±0.5

TABLE 3. Comparison of our results with the state-of-the-art on SHREC'22 gesture recognition challenge. Our results are averaged over 5 random seeds (RS) and 5-fold cross-validation (CV).

Approach	Detection rate (%)	False positives (%)	Jaccard Index (%)
Stronger [28]	71.9	33.0	59.2
2ST-GCN 5F [28]	73.8	10.4	67.2
Causal TCN [28]	80.0	25.5	68.5
TN-FSM+JD [28]	77.1	18.2	65.8
[8]	85.4	9.2	78.8
TY-Net (RS)	**90.2** ±1.2	**4.4** ±0.8	**86.6** ±1.1
TY-Net (CV)	**89.3** ±0.4	**3.5** ±0.9	**86.2** ±0.5

IPN Hand dataset. This discrepancy can partly be due to the ambiguities in gesture classes within the IPN Hand dataset; specifically, two gesture classes (2click-1f and 2click-2f) are equivalent to performing another gesture class twice (Click-1f and Click-2f). Additionally, the disparity can be attributed to the higher quality of 3D hand skeleton data in SHREC'22, captured using specialized time-of-flight camera sensors, in contrast to the hand skeleton extracted from monocular video through a 3D skeletonization model applied to the IPN Hand dataset. Moreover, 3D skeletonization model is more sensitive to illumination variations, which are present in the IPN Hand dataset, compared to the specialized sensors. Nonetheless, as shown in Table 2, our approach shows robustness and outperforms most of the methods employing different input modalities on IPN Hand, despite the presence of illumination variation in the dataset and the inherent noise introduced through the hand skeleton extraction method.

C. IMPACT OF HANDCRAFTED FEATURES

In addition to capturing the 3D coordinates of hand joints, we have also derived additional features that are concatenated with the input vector. To assess the impact and necessity of these additional features, we conducted experiments training the model on both datasets using only flattened 3D hand coordinates as input. The results, shown in Table 4, indicate that the inclusion of additional features leads to improved performance on the SHREC'22 dataset. However, on the IPN Hand dataset, the results remain similar regardless of the use of the additional features. This suggests that the impact of these features may vary depending on the specific characteristics of the dataset and the gestures being analyzed.

TABLE 4. Comparison of results with and without handcrafted features. The metrics are detection rate (DR), false positives (FP), Jaccard index (JI) and Levenshtein accuracy (LA). The result are averaged over 5 random seeds.

Dataset	Additional features	DR (%)	FP (%)	JI (%)	LA (%)
SHREC'22	No	87.6 ±0.8	13.1 ±1.6	78.7 ±1.9	85.9 ±1.5
SHREC'22	Yes	90.2 ±1.2	4.4 ±0.8	86.6 ±1.1	91.8 ±0.9
IPN Hand	No	76.1±1.4	32.6±0.8	61.1±1.2	59.8 ±0.4
IPN Hand	Yes	75.4±1.1	30.2±1.3	61.7±1.4	60.5 ±0.2

D. MODEL HYPER-PARAMETERS

To validate our model architecture hyperparameters, we conducted experiments on SHREC'22, varying the number of self-attention heads, layers, and hidden layer sizes in the Transformer encoder. The summarized results in Table 5 show that a hidden layer size of 256 yielded the best performance. We also found that increasing the number of self-attention heads from 4 to 8 improved performance, while the ideal number of layers remained consistent at 2.

Both the architecture with 8 self-attention heads and our 4-head variant have approximately five million parameters and require 0.18 GFLOPs, indicating suitability for real-time applications. To further verify this, we measured the combined execution time for hand keypoint extraction via Mediapipe, keypoints preprocessing, and gesture recognition. It took approximately 19 ms per input frame when utilizing an NVIDIA GeForce RTX 3070Ti GPU and an Intel Core i9 CPU. Consequently, the system achieves an effective frame rate of approximately 52 frames per second (fps).

Figure 11a illustrates validation loss variation across training epochs for models with varying hidden layer sizes. As visible, the model with a size of 512 shows more pronounced variations, while others maintain smoother loss curves. Adjusting the number of self-attention heads or layers was shown to have minimal impact on the convergence of the models, so the training and evaluation curves are similar to those presented in Figure 11b for the best-performing model, showcasing the robustness of the models in terms of convergence.

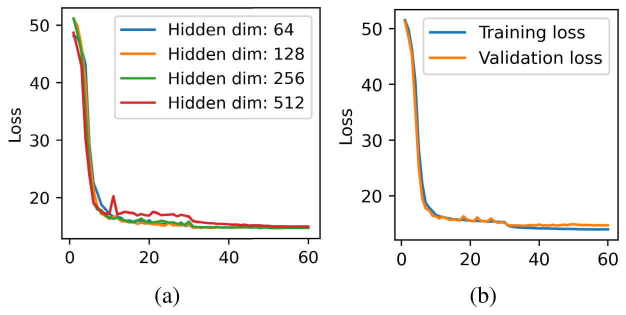

FIGURE 11. Variation of (a) validation loss for models with different hidden layer sizes (b) training and validation loss for the best model, across training epochs.

E. IMPACT OF λ_G AND λ_{NOG}

To assess the impact of factors λ_g and λ_{nog} on model performance, we conducted experiments with various combinations of values using the SHREC'22 dataset, and the results are summarized in Table 6. The Jaccard Index consistently decreases as λ_{nog} increases across all values of λ_g. This observation aligns with our intuition, suggesting that setting λ_{nog} to a smaller value than λ_g is favorable due to the prevalence of no-gesture samples. The optimal configuration is achieved when λ_g is set to 2 and λ_{nog} is set to 0.5, slightly differing from our original choice of 2 for λ_g and 1 for λ_{nog}.

TABLE 5. Comparison of results on SHREC'22 with varying number of self-attention heads, layers, and hidden layer sizes in the transformer encoder.The result are averaged over 5 random seeds.

Hidden	Heads	Layers	Params	GFLOPs	JI (%)
64	4	2	357,843	0.01	82.2 $\pm$1.3
128	4	2	1,305,491	0.04	82.8 $\pm$0.8
256	4	2	4,970,256	0.18	**85.6** $\pm$1.0
512	4	2	19,377,683	0.68	78.5 $\pm$1.8
256	1	2	4,970,256	0.18	82.2 $\pm$1.9
256	2	2	4,970,256	0.18	85.5 $\pm$1.1
256	4	2	4,970,256	0.18	85.6 $\pm$1.0
256	8	2	4,970,256	0.18	**86.7** $\pm$1.7
256	8	1	3,392,275	0.03	83.3 $\pm$1.0
256	8	2	4,970,256	0.18	**86.7** $\pm$1.7
256	8	4	8,126,227	0.35	85.9 $\pm$0.8

TABLE 6. Comparison of results on SHREC'22 with different values of λ_g and λ_{nog}. The result are averaged over 5 random seeds.

λ_g	λ_{nog}	JI
1	0.5	86.5 $\pm$1.1
1	1	85.7 $\pm$1.6
1	2	85.6 $\pm$0.8
1	4	85.4 $\pm$0.9
2	0.5	86.9 $\pm$1.3
2	1	86.0 $\pm$1.0
2	2	85.4 $\pm$1.5
2	4	85.3 $\pm$0.2
4	0.5	85.7 $\pm$0.9
4	1	85.5 $\pm$1.1
4	2	85.4 $\pm$0.9
4	4	84.5 $\pm$0.9

VI. CONCLUSION

In this paper, we presented a novel formulation of online hand gesture recognition problem inspired by the YOLO model for object detection. For a segment of continuous input, the proposed model can directly predict gesture boundaries and gesture class. The transformer-based model is trained in an end-to-end manner using a proposed loss function that focuses on samples containing gesture centers and learns to predict gesture boundaries. Moreover, the proposed problem formulation enables early predictions, which is crucial for real-world applications. The results obtained on two continuous hand gesture datasets underscore the effectiveness of our approach. On the SHREC'22 gesture dataset, we achieved a significant reduction in false positives and obtained state-of-the-art performance. Our results on the IPN Hand dataset outperformed all uni-modal approaches and even surpassed several multi-modal approaches from the literature. By relying solely on the hand skeleton as input, our approach offers a simpler and more computationally efficient alternative to techniques that incorporate additional input modalities. The findings presented in this study demonstrate the potential of our approach to enable more accurate and efficient gesture recognition. The objective of future work is to incorporate the gesture boundary information obtained by the model into the gesture activation process.

REFERENCES

[1] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: A review of techniques," *J. Imag.*, vol. 6, no. 8, p. 73, Jul. 2020.

[2] J. Suarez and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *Proc. 21st IEEE Int. Symp. Robot Hum. Interact. Commun.*, Sep. 2012, pp. 411–417.

[3] L. Guo, Z. Lu, and L. Yao, "Human–machine interaction sensing technology based on hand gesture recognition: A review," *IEEE Trans. Hum.-Mach. Syst.*, vol. 51, no. 4, pp. 300–309, Aug. 2021.

[4] Z. Liu, X. Chai, Z. Liu, and X. Chen, "Continuous gesture recognition with hand-oriented spatiotemporal feature," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3056–3064.

[5] H. Wang, P. Wang, Z. Song, and W. Li, "Large-scale multimodal gesture recognition using heterogeneous networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3129–3137.

[6] N. N. Hoang, G.-S. Lee, S.-H. Kim, and H.-J. Yang, "Continuous hand gesture spotting and classification using 3D finger joints information," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 539–543.

[7] G. Zhu, L. Zhang, P. Shen, J. Song, S. A. A. Shah, and M. Bennamoun, "Continuous gesture segmentation and recognition using 3DCNN and convolutional LSTM," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 1011–1021, Apr. 2019.

[8] Z. Doždor, T. Hrkac, and Z. Kalafatic, "Two-model-based online hand gesture recognition from skeleton data," in *Proc. 18th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2023, pp. 838–845.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[10] X. Wang, S. Zhang, Z. Qing, Y. Shao, Z. Zuo, C. Gao, and N. Sang, "OadTR: Online action detection with transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7545–7555.

[11] M. Xu, Y. Xiong, H. Chen, X. Li, W. Xia, Z. Tu, and S. Soatto, "Long short-term transformer for online action detection," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 1086–1099.

[12] L. Hedegaard, A. Bakhtiarnia, and A. Iosifidis, "Continual transformers: Redundancy-free attention for online inference," 2022, *arXiv:2201.06268*.

[13] X. Chen, H. Guo, G. Wang, and L. Zhang, "Motion feature augmented recurrent neural network for skeleton-based dynamic hand gesture recognition," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2881–2885.

[14] M. Maghoumi and J. J. LaViola, "DeepGRU: Deep gesture recognition utility," in *Proc. Int. Symp. Vis. Comput.*, 2018, pp. 16–31.

[15] S. Shin and W.-Y. Kim, "Skeleton-based dynamic hand gesture recognition using a part-based GRU-RNN for gesture-based interface," *IEEE Access*, vol. 8, pp. 50236–50243, 2020.

[16] G. Devineau, F. Moutarde, W. Xi, and J. Yang, "Deep learning for hand gesture recognition on skeletal data," in *Proc. 13th IEEE Int. Conf. Autom. Face Gesture Recognit.*, May 2018, pp. 106–113.

[17] J. Hou, G. Wang, X. Chen, J. Xue, R. Zhu, and H. Yang, "Spatial–temporal attention Res-TCN for skeleton-based dynamic hand gesture recognition," in *Proc. Comput. Vis.-ECCV Workshops*, 2019, pp. 273–286.

[18] J. C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vélez, "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition," *Pattern Recognit.*, vol. 76, pp. 80–94, Apr. 2018.

[19] E. Zhong, C. R. Del-Blanco, D. Berjón, F. Jaureguizar, and N. García, "Real-time monocular skeleton-based hand gesture recognition using 3D-jointsformer," *Sensors*, vol. 23, no. 16, p. 7066, Aug. 2023.

[20] P. Elias, J. Sedmidubsky, and P. Zezula, "Motion images: An effective representation of motion capture data for similarity search," in *Proc. Int. Conf. Similarity Search Appl.*, 2015, pp. 250–255.

[21] J. Sedmidubsky, P. Elias, and P. Zezula, "Effective and efficient similarity searching in motion capture data," *Multimedia Tools Appl.*, vol. 77, no. 10, pp. 12073–12094, May 2018.

[22] J. Gesnouin, S. Pechberti, G. Bresson, B. Stanciulescu, and F. Moutarde, "Predicting intentions of pedestrians from 2D skeletal pose sequences with a representation-focused multi-branch deep learning network," *Algorithms*, vol. 13, no. 12, p. 331, Dec. 2020.

[23] A. Caputo, A. Giachetti, F. Giannini, K. Lupinetti, M. Monti, M. Pegoraro, and A. Ranieri, "SFINGE 3D: A novel benchmark for online detection and recognition of heterogeneous hand gestures from 3D fingers' trajectories," *Comput. Graph.*, vol. 91, pp. 232–242, Oct. 2020.

[24] K. Lupinetti, A. Ranieri, F. Giannini, and M. Monti, "3D dynamic hand gestures recognition using the leap motion sensor and convolutional neural networks," in *Proc. Int. Conf. Augmented Virtual Reality (AVR)*, 2020, pp. 420–439.

[25] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 7444–7452.

[26] Y. Chen, L. Zhao, X. Peng, J. Yuan, and D. N. Metaxas, "Construct dynamic graphs for hand gesture recognition via spatial–temporal attention," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–13.

[27] Y. Sun, W. Xu, X. Yu, J. Gao, and T. Xia, "Integrating vision transformer-based bilinear pooling and attention network fusion of RGB and skeleton features for human action recognition," *Int. J. Comput. Intell. Syst.*, vol. 16, no. 1, pp. 1–11, Jul. 2023.

[28] M. Emporio, A. Caputo, A. Giachetti, M. Cristani, G. Borghi, A. D'Eusanio, M.-Q. Le, H.-D. Nguyen, M.-T. Tran, F. Ambellan, M. Hanik, E. Nava-Yazdani, and C. von Tycowicz, "SHREC 2022 track on online detection of heterogeneous gestures," *Comput. Graph.*, vol. 107, pp. 241–251, Jan. 2022.

[29] Y. Yin and R. Davis, "Real-time continuous gesture recognition for natural human–computer interaction," in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput. (VL/HCC)*, Jul. 2014, pp. 113–120.

[30] E. Coupeté, F. Moutarde, and S. Manitsaris, "Multi-users online recognition of technical gestures for natural human–robot collaboration in manufacturing," *Auto. Robots*, vol. 43, no. 6, pp. 1309–1325, Aug. 2019.

[31] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in *Proc. 14th IEEE Int. Conf. Autom. Face Gesture Recognit.*, May 2019, pp. 1–8.

[32] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4207–4215.

[33] Y. A. Farha and J. Gall, "MS-TCN: Multi-stage temporal convolutional network for action segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3570–3579.

[34] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster R-CNN architecture for temporal action localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1130–1139.

[35] T. Lin, X. Zhao, and Z. Shou, "Single shot temporal action detection," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 988–996.

[36] S. Buch, V. Escorcia, B. Ghanem, and J. C. Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.

[37] D. Polap, "Neuro-heuristic analysis of surveillance video in a centralized IoT system," *ISA Trans.*, vol. 140, pp. 402–411, Sep. 2023.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6000–6010.

[39] G. Benitez-Garcia, J. Olivares-Mercado, G. Sanchez-Perez, and K. Yanai, "IPN hand: A video dataset and benchmark for real-time continuous hand gesture recognition," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 4340–4347.

[40] (2022). *SHREC 2022*. [Online]. Available: https://univr-vips.github.io/Shrec22

[41] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe hands: On-device real-time hand tracking," 2020, *arXiv:2006.10214*.

[42] (2020). *The IPN Hand Dataset*. [Online]. Available: https://gibranbenitez.github.io/IPN_Hand

[43] J. Wan, S. Z. Li, Y. Zhao, S. Zhou, I. Guyon, and S. Escalera, "ChaLearn looking at people RGB-D isolated and continuous datasets for gesture recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 761–769.

[44] Y. Zhang, C. Cao, J. Cheng, and H. Lu, "EgoGesture: A new dataset and benchmark for egocentric hand gesture recognition," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1038–1050, May 2018.

[45] A. Caputo, A. Giachetti, S. Soso, D. Pintani, A. D'Eusanio, S. Pini, G. Borghi, A. Simoni, R. Vezzani, R. Cucchiara, A. Ranieri, F. Giannini, K. Lupinetti, M. Monti, M. Maghoumi, J. J. LaViola Jr., M.-Q. Le, H.-D. Nguyen, and M.-T. Tran, "SHREC 2021: Skeleton-based hand gesture recognition in the wild," *Comput. Graph.*, vol. 99, pp. 201–211, Oct. 2021.

[46] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys.-Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[47] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[48] N. Nayan, D. Ghosh, and P. M. Pradhan, "A CNN bi-LSTM based multimodal continuous hand gesture recognition," in *Proc. IEEE India Council Int. Subsections Conf. (INDISCON)*, India, Jul. 2022, pp. 1–4.

[49] T.-T. Nguyen, N.-C. Nguyen, D.-K. Ngo, V.-L. Phan, M.-H. Pham, D.-A. Nguyen, M.-H. Doan, and T.-L. Le, "A continuous real-time hand gesture recognition method based on skeleton," in *Proc. 11th Int. Conf. Control, Autom. Inf. Sci. (ICCAIS)*, Nov. 2022, pp. 273–278.

[50] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, "TMMF: Temporal multi-modal fusion for single-stage continuous gesture recognition," *IEEE Trans. Image Process.*, vol. 30, pp. 7689–7701, 2021.

ZORANA DOŽDOR received the M.Sc. degree in computing from the University of Zagreb, in 2021. She is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering and Computing, University of Zagreb.

She is currently a Junior Researcher with the Faculty of Electrical Engineering and Computing, University of Zagreb. Her research interests include computer vision and deep learning.

ZORAN KALAFATIĆ (Member, IEEE) received the Ph.D. degree in computer science from the University of Zagreb, Croatia, in 1999.

He is currently an Associate Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb. He is a member of the Center of Excellence for Computer Vision, Faculty of Electrical Engineering and Computing, University of Zagreb. He is also a member of the Centre of Research Excellence for Data Science and Advanced Cooperative Systems, University of Zagreb. His research interests include computer vision and deep learning.

ŽELJKO BAN (Member, IEEE) received the Ph.D. degree from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 1999. He is currently a Full Professor with the Department of Control and Computer Engineering in Automation, Faculty of Electrical Engineering and Computing, University of Zagreb. Since 2006, his research activities have been focusing on intelligent control of fuel cell energy sources and control of microgrids consisting of photovoltaic systems, fuel cell systems, and wind energy sources. His research interests include adaptive and optimal control and control of energy storage systems.

TOMISLAV HRKAĆ (Member, IEEE) received the Ph.D. degree in computer science from the University of Zagreb, Croatia, in 2009.

He is currently an Associate Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb. He is a member of the Laboratory for Pattern Recognition and Biometric Security Systems and the Center of Excellence for Computer Vision, Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include computer vision and deep learning.