Neural Computing and Applications (2023) 35:253–274
https://doi.org/10.1007/s00521-022-08018-2

REVIEW

A shapelet-based framework for large-scale word-level sign language database auto-construction

Xiang Ma[1] · Qiang Wang[1] · Tianyou Zheng[1] · Lin Yuan[1]

Received: 1 March 2022 / Accepted: 26 October 2022 / Published online: 20 November 2022

Abstract
Sign language recognition is a challenging and often underestimated problem that includes the asynchronous integration of multimodal articulators. Learning powerful applied statistical models requires much training data. However, well-labelled sign language databases are a scarce resource due to the high cost of manual labelling and performing. On the other hand, there exist a lot of sign language-interpreted videos on the Internet. This work aims to propose a framework to automatically learn a large-scale sign language database from sign language-interpreted videos. We achieved this by exploring the correspondence between subtitles and motions by discovering shapelets which are the most discriminative subsequences within the data sequences. In this paper, two modified shapelet methods were used to identify the target signs for 1000 words from 89 (96 h, 8 naive signers) sign language-interpreted videos in terms of brute force search and parameter learning. Then, an augmented (3–5 times larger) large-scale word-level sign database was finally constructed using an adaptive sample augmentation strategy that collected all similar video clips of the target sign as valid samples. Experiments on a subset of 100 words revealed a considerable speedup and 14% improvement in recall rate. The evaluation of three state-of-the-art sign language classifiers demonstrates the good discrimination of the database, and the sample augmentation strategy can significantly increase the recognition accuracy of all classifiers by 10–33% by increasing the number, variety, and balance of the data.

Keywords Sign language · Shapelet · Self-learning · Big data computing

1 Introduction

Sign language (SL) is the main communication way for most deaf or hearing-loss people. As of 2013, approximately 1.1 billion people worldwide have varying degrees of hearing loss [1], 124 million of whom are moderate and severe [2]. Unlike the general view that sign language is just the direct gesture description of objects or a translation of the spoken(written) language, sign language is actually a natural language with its own linguistics, words, sentences, and grammar. Sign languages were created and developed in the deaf communities relatively independently, although they are influenced impressively by the cultures and languages of the hearing societies. Sign language is culturally and geographically diverse, much like all other natural languages. Sign language was not widely believed as a natural language until Stokoe's publication of *sign language structure* in the 1960s [3]. This is the first time that sign language was studied with linguistic methodology. *Stokoe* presented persuasive evidence that *American Sign Language* (ASL) is indeed a natural language with grammar and vocabulary independent of English.

Sign language recognition (SLR) research began to emerge and received increasing attention in the 1990s, which can improve communication between the hearing and the deaf. Researches on sign language were initially

✉ Xiang Ma
15b904026@hit.edu.cn

Qiang Wang
wangqiang@hit.edu.cn

Tianyou Zheng
drenego@hit.edu.cn

Lin Yuan
_eunseo_v@hit.edu.cn

[1] Department of Control Science and Engineering, Harbin Institute of Technology, Harbin 150001, People's Republic of China

mostly done along the same technical lines as speech recognition. Numerous effective methods for speech recognition were directly transferred to SLR. However, asynchronous integration of multimodal articulators is a tough and frequently ignored issue in SLR. As shown in Stokoe's [3] sign language model, five individual components are present, i.e. movement, location, orientation, handshape, and facial expression.

The hidden Markov model (HMM) [4] is the most classical and well-known method for SLR, which can model the transfers between latent states of sequence data. Various variants of HMM, such as maximum entropy Markov model (MEMM) [5], conditional random fields (CRF) [6], product-HMM [7], and hierarchy-HMM [8], have also been widely used in SLR studies. However, due to the lack of a unified definition of sign language syllables or primitives so far, HMM methods are generally only applicable to simple, small-category, isolated SLR. In recent years, SLR researches based on deep neural networks (e.g. GRU [9], TGCN [10], I3D [11]) have became more and more popular. Deep neural networks' enormous number of parameters allows them to accommodate sign language's intricate spatio-temporal structures. A significant amount of training data is frequently necessary to learn a practical, powerful statistical model.

However, labelled data is a scarce resource for sign language due to the enormous cost of transcribing these unwritten languages. The majority of publicly accessible sign language databases to date have been created in lab settings. A small corpus of signs will be selected or designed first, and each sign will then be performed repeatedly by a number of signers (native or non-native). Some databases use data-gloves to capture every detail of arms and fingers [12–14], while others use RGB or RGB-depth cameras to record the sign language videos [11, 15]. Signers are typically instructed to wear coloured gloves in order to improve the robustness of palm detection during the creation of visual-based sign language databases. These laboratory-environment databases usually have restricted size and variation due to the high cost of performance and annotation. In fact, more and more videos of TV news, press conferences, and parliamentary speeches have been supplemented with real-time sign language interpretation. There are a lot of sign language-interpreted videos on the Internet (e.g. video websites, online deaf communities). Unfortunately, the supervisory information of these sign language-interpreted videos is weak and noisy due to the lexical differences, time misalignment, and grammatical inconsistencies between sign language and speech language..

Data mining is a process for knowledge discovery that can draw out latent patterns, relationships, and correlations from massive data. Data mining methods have been largely used in many fields, from medical to social life. For example, influenza epidemics can be detected early by analysing large numbers of Google search queries in a population [16], and Kawasaki disease clinical graph signs can be detected by a deep convolutional neural network (CNN) [17]. The characteristic and necessary motions can be extracted using motif-guided attention networks [18]. Then, if the unlabelled signs can be minded from plenty of existing sign language-interpreted videos, it will be a promising solution to the scarcity of labelled sign language resources.

The goal of this work is to propose a novel framework to automatically learn a large-scale sign language database from these sign language-interpreted videos. We achieved this by exploiting supervisory information available in the subtitles (or audio) of the videos through learning shapelets which are discriminative subsequences of time series that best predict the target variable. The main contributions of this paper are:

- We provide a framework that can automatically create a sign language database from massive sign language-interpreted videos by integrating pose extraction, subtitle parsing, shapelet mining, and sample augmentation.
- We introduce the tricks of iterative update and matrix operation to greatly speed up the searching of shapelets.
- We propose a strategy that combines shapelet searching and shapelet learning to benefit both speed and accuracy.
- We demonstrate that adaptive sample augmentation can greatly improve the database's size, variety, and balance.

The rest of paper will be organized as follows: Section 2 summarizes the related works of SLR, sign language databases, multiple instances learning and sample augmentation. Section 3 gives definitions of the main terminology and provides an overall introduction to the proposed framework. Section 4 describes in detail how to process the original sign language videos (including subtitles) and how to generate training samples for a given word based on supervised information. Section 5 demonstrates how to speed up the target sign extraction of two available shapelet discovering algorithms. The construction of a concrete database and experiments on recall rate and classification in Sect. 6 serve to illustrate the usefulness and efficiency of our proposed framework. The experimental results and the impacts of the parameters are analyzed and discussed in Sect. 7. Finally, Sect. 8 concludes the whole work and provides an outlook for future work.

2 Related works

Our work relates to several themes in the literature, such as sign language recognition, sign language databases, multiple instance learning, and sample augmentation.

2.1 Sign language recognition

The study of automatic SLR has developed for about 30 years since the 1990s. Since non-motion features (i.e. facial expression [19, 20]) are difficult to identify and quantify, most of the researches only focus on motion features. Shape-based SLR studies focus on the space features of hands and poses through the designed descriptors of trajectories and shapes [21–24]. HMMs [4–6, 25] have been the dominant method used to model state transfers of sign language sequences until the advent of deep learning. Deep neural networks such as CNNs [26, 27], RNNs [9, 28–30], TGCN [10], and Transformers [31, 32] have been proven to be effective architectures to model the complex spatio-temporal structures of sign language.

Meanwhile, kinds of body models [33–36] have been proposed to extract human skeletons from images. In particular, the *sequence convolution network* in [35, 36] will map images into confidence maps of skeleton keypoints. Additionally, its accessible trained model can be used without further training to perform a real-world posture prediction task, making video-based SLR more convenient and efficient. However, I3D [11], a sign language model that convolves videos along both spatial and temporal dimensions has shown its superiority in video-based SLR [10, 37–39]. In this work, three state-of-the-art models (GRU [9], TGCN [10], and I3D [11]) were adopted to evaluate our final database.

2.2 Sign language databases

A summary and review of some outdated sign language databases can be found in [40]. And here, we will introduce a couple of the most representative sign language databases currently available. Purdue RVL-SLLL [41] contains 104 words and 1834 samples, which were performed by 14 native signers in a laboratory environment under controlled lighting. RWTH-Boston [42] contains three subsets of 50, 104, and 400 signs, which are implemented with two to five native signers to perform isolated words and continuous sentences. DeviSign is a large-scale word-level sign language database containing up to 2000 Chinese sign languages and 24,000 RGB-depth recordings performed by eight non-native signers in a laboratory environment (controlled background) [15]. These databases, regardless of size, are built in a laboratory environment, which is highly expensive and mostly with limited size and variation.

Both MSASL [37] and WLASL [10] are large-scale word-level sign language databases that collect *isolated* signs from the Internet. However, there are significant differences between the *isolated* signs and the *co-articulated* signs during "naturally" continuous signing. Also, there are far fewer isolated sign language videos on the Internet than the continuous sign language videos. BSL-1k [38] is a sign language database that contains a vocabulary of 1064 signs extracted from more than 1000 h of continuous BBC sign language TV programmes. The extraction of signs relies only on lip changes without concern for body movements. Such a sign language database construction framework, which relies exclusively on lip-synthesis models, has low data utilization and is difficult to migrate directly to the database construction of other sign languages.

2.3 Multiple instance learning

Multiple instance learning (MIL) problem belongs to a weak supervisory problem, inferring the label of individual instances with the given labels of bags of instances [43, 44]. Paper [45] shows that localizing the target signs for a given word using subtitles is essentially a two-class MIL problem, and a *sliding window classifier* is also proposed to find the clip with the highest score as the target sign. Paper [46] makes two improvements to [45]: first, it shrinks the search space by exploiting the co-occurrences of lip movements, and second, it trains a discriminative model of the target signs using the MIL-SVM (support vector machine) [47] method. The *Aprior Mining* algorithm was adopted to infer the rules between signs and words with positive and negative discrete-encoded sign language video clips in the paper [48]. And in [49], the correspondence between isolated signs and words was implicitly learned through modelling the translation process between subtitle sentences and their video clips with the transformer model. In the latest study [39], the author used a designed embedding architecture and the InfoNCE [50] loss to train a feature model for the target signs. The architecture comprises an I3D spatio-temporal trunk network [11] attached with a MIL trunk consisting of three linear layers separated by leaky ReLu activation and a skip connection.

The concept of *shapelet* proposed in [51] is actually a solution to the MIL problem discussed above. The shapelet and its nearest subsequence in each positive sample are the desired target signs. In theory, finding a shapelet requires searching all possible subsequences [51], similar to the *sliding window classifier* in [45]. However, there are many tricks, such as early abandoning, reuse of computation, parallel computing, and candidate filtering, that can be used

to speed up the searching process [52–56]. Furthermore, there are works that use all of the shapelet values as unknown parameters, and then use gradient descent methods to learn the shapelet values [57, 58]. There are also studies that add additional constraints during the learning process to make the shapelet have more expected characteristics [59, 60].

In this work, we will find the target signs for a given word from the perspective of solving shapelet with weakly supervised information.

2.4 Samples augmentation

Data augmentation, which generates new samples by slightly varying the available samples, is very common in model training. The sample augmentation of signs is a special case of data augmentation that takes all similar instances of a given sample in the data source as the augmented samples [61]. These augmented samples obtained from real data source have more reasonable, rich, and realistic variations. The experiments in [61] show that the augmented samples obtained from arbitrarily collected unlabelled sign language videos can make a given sign sample have the ability of one-shot learning. Papers [38, 39] also show that although more noisy samples will be introduced if the criteria of the sign spotting are relaxed, more training samples can bring better recognition evaluation results. The experiments in our paper are also consistent with this.

3 Notation and overview

3.1 Definition and notations

To make the description of our work clear and distinct, some key terms are defined as follows:

Definition 1 *Time series.* A time series T is a list of data points ordered along time, $T = t_1, t_2, \ldots, t_N$, each data point is a feature vector.

Definition 2 *Subsequence.* A subsequence is a slice of consecutive data points cut from a time series T. For example, $S = T_{k:k+m} = t_k, t_{k+1}, \ldots t_{k+m-1}$ is a subsequence that cut at k_{th} point and with length of m.

Definition 3 *Dis. Dis* is defined as the Euclidean square distance function between two time series with same length.

$$Dis(T,R) = \sum_{i=1}^{N}(t_i - r_i)^2 \tag{1}$$

where N is the length of two time series.

Definition 4 *subDis. subDis* is defined as a distance function between two time series with different length. The shorter time series is query sequence Q, the longer time series is searching sequence T. $subDis(Q, T)$ returns the minimum of the Euclidean square distances between Q and all $|Q|$-length subsequences of T. That is:

$$subDis(Q,T) = \min Dis(Q,S), \quad \forall S \in \mathbb{S}_T^{|Q|} \tag{2}$$

$$\mathbb{S}_T^{|Q|} = \left\{T_{k:k+|Q|} \mid 1 \le k \le |T| - |Q| + 1\right\}. \tag{3}$$

Definition 5 *word & sign.* To avoid the confusion of terminology of the two natural languages. We specify that the term "word" refers to the "isolated word" of the written (spoken) language, and the term "sign" refers to the "isolated word" of the sign language.

Definition 6 *Subtitle frame.* The minimum unit element of a subtitle file. It consists of three basic components: short text, begin timestamp, and end timestamp. Which represents the interpreted information of the video clip from begin timestamp to end timestamp.

Definition 7 *Candidate Time Window.* A potential time window that contains the target "sign" for a given "word". Because of the rough timing of the subtitles and the intricate link between the two natural language, it is difficult to pinpoint the exact location of the target sign.

3.2 Overview of framework

The proposed framework's flowchart is shown in Fig. 1. The framework receives files containing sign language videos and their subtitles as input, and then a sign language database is constructed as output. There are four main steps in the framework. We will describe each of these steps below.

Step one is the sign motion features extraction. For an input video, the region around the signer will be cropped first. Then the keypoints of his/her hands and upper body will be detected and tracked. Finally, the sign motion features can be designed based on the positions of keypoints. *Step two* prepares the training data. A corpus consisting of reasonable words is constructed from the subtitle files. Then, for each *given word* from the corpus, the same number of positive and negative samples determined by the designed *candidate window* are chosen as the training data. *Step three* learns the shapelet from the training data with shapelet searching and shapelet Net methods. The *shapelet* is the subsequence that best classifies the training samples using *subDis* distances. *Step four* is sample augmentation. To increase the sample number per word and the variations of the final database, all motion subsequences that are very

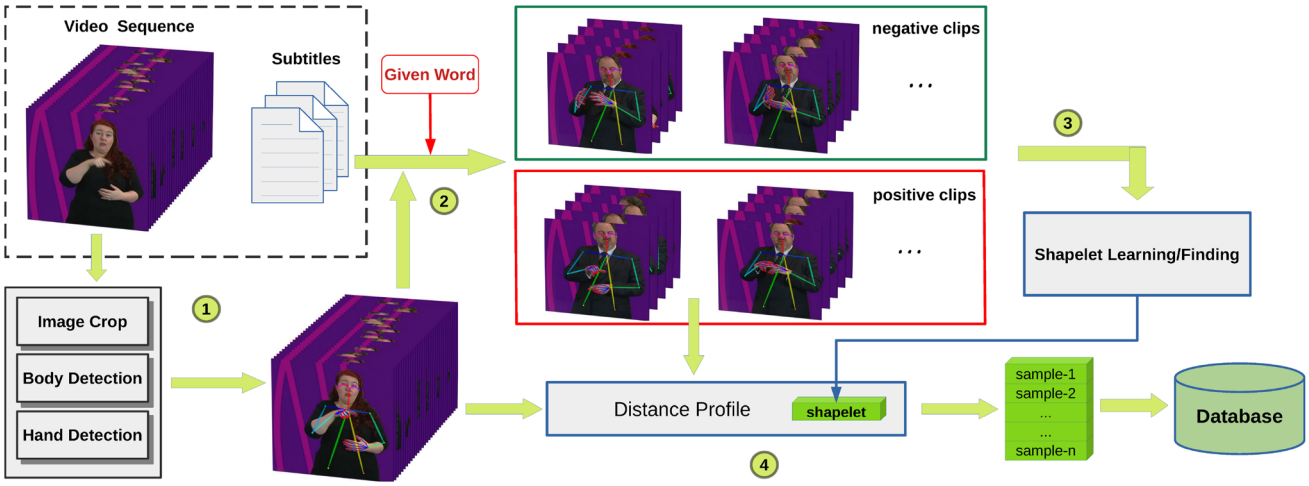


Fig. 1 The flowchart of the sign language database auto-construction framework

similar to the learned shapelet will be appended to the final database as augmented samples. Almost all the parameters of the whole framework can be learned automatically.

4 Automatic generation of training data

This section, which discusses steps one and two, provides the specifics of preprocessing, which covers how to extract motion features, process subtitles, and generate training examples for a given word.

4.1 Sign language features designing

In our framework, the region of interest (ROI) of each video frame is first identified and cropped, and then the *openpose*[1] library is used to detect the keypoints of signers. Figure 2 depicts the distribution of the detected keypoints of openpose. There are 60 keypoints in total, including 18 trunk keypoints and 42 hand keypoints (two hands). One of the most attractive advantages of *openpose* library is its powerful model migration ability that the available trained model can be directly used in our work without retuning. Overall, *openpose* is stable for the detection of joints of the body (including head, limbs, and torso), but the detection of finger joints is usually not good in the situation of low resolution, blurring, and blocking.

For sign language, only the keypoints of the upper body and hands are considered. Due to the disturbing factors of object blocking, target missing, perspective rotating, and background interfering, the position coordinates of the extracted keypoints should be filtered first to eliminate invalid values and outliers. In addition, the coordinates of the obtained keypoints are absolute values. To avoid

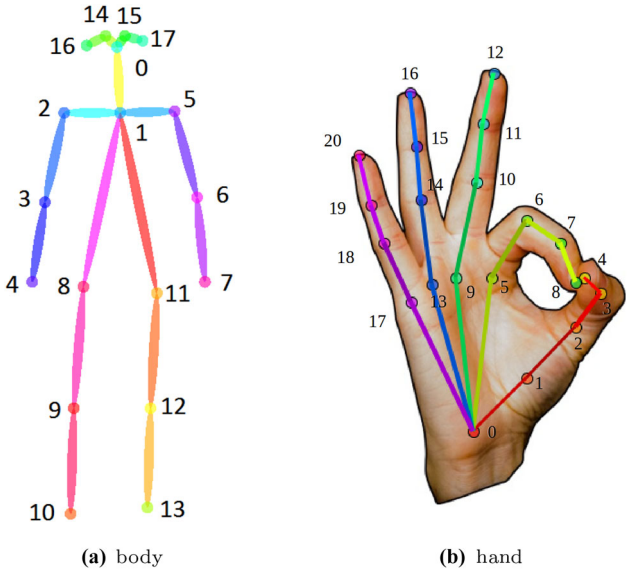


Fig. 2 The output keypoints of *openpose* library

interference from the height, distance, position, viewpoint, etc., the coordinates need to be resized with the formula below. All coordinates are calculated relative to the chest keypoint; then, coordinates will be normalized with the vertical distance between the keypoints of the nose and chest.

$$\hat{\boldsymbol{P}}_i = (\boldsymbol{P}_i - \boldsymbol{P}_1)\frac{||\boldsymbol{P}_i - \boldsymbol{P}_1||_2}{|P_1^y - P_0^y|} \tag{4}$$

where $\boldsymbol{P_i} = (P_i^x, P_i^y)$ is the 2D position coordinate of the keypoint i, and the keypoints $\boldsymbol{0}$ and $\boldsymbol{1}$ represent the keypoints of *chest* and *nose* respectively.

[1] https://github.com/CMU-Perceptual-Computing-Lab/openpose.

4.2 Subtitle processing

The supervised information of sign language-interpreted videos is available in three forms: stand-alone subtitle files (e.g. vtt, srt format files), embedded subtitles, and interpretation soundtracks. The latter two can be transformed into stand-alone subtitle files through *optical character recognition* (OCR) and speech recognition, respectively. This supervised information is both weak and noisy. It is weak since the temporal distance between sign and subtitle is unknown and the act of signing does not follow the text order. It is noisy because subtitles can be signed in different ways, and the occurrence of a subtitle word does not imply the presence of the corresponding sign [45].

In an ideal situation, each word in subtitles may correspond to a video clip, and all words in the subtitles should be collected to construct a corpus. However, more purification operations should be executed to make the corpus more reasonable. First of all, the *stemming* and *lemmatization* methods should be used to remove the inflections (e.g. "s", "es", "ed", "ing") of each word in the corpus, and the lemma form words are obtained. Second, low-frequency words will be removed from the corpus. Third, words with unclear meanings will be removed from the corpus as stop words. Finally, to increase the one-to-one correspondence between *words* and *signs*, the polysemous words (one word has multiple meanings) will also be removed.

4.3 Training samples generation

For a given word in the resulted corpus, all *subtitle frames* [as Definition 6] whose short text contains the given word are defined as the positive frame. Then, the corresponding video clips are defined as the positive samples. Conversely, the negative samples are defined as the video clips that will not overlap with the positive samples of the given word and its synonyms. In general, the candidate time window of a positive sample needs to be widened to ensure the latent target sign can be enclosed. The usual procedure is to extend the time window ahead and backward by one subtitle frame, denoted as:

$$pos\_win = \left[f_{t-1}^{\text{begin}}, f_{t+1}^{\text{end}}\right]$$

where f_t represents the ith subtitle frame, while the superscripts *begin* and *end* indicate the timestamp. However, there is a delay in signing versus speaking for most real-time sign language translation videos. When the time span of the following subtitle frame is too short, it is not guaranteed that the extended time window can enclose the latent target sign.

In this work, an adaptive extension method is proposed to determine the candidate time window: the time window should contain the time span of the preceding, current, and next k subtitle frames, where k is the minimum positive integer that satisfies the following condition:

$$f_{t+k}^{\text{end}} - f_t^{\text{end}} \geq \beta \cdot \left(f_t^{\text{end}} - f_t^{\text{begin}}\right) \tag{5}$$

The above formula states that the extended time along the time increase cannot be less than the β times the time span of the positive subtitle frame. In most cases, β should be larger than 1.

5 Automatic sign extraction

This section will describe how to find target signs for a given word from its positive and negative samples. We anticipate that the target sign will appear in the majority of positive samples and would not occur in negative samples based on the definition of samples. Then, we can formulate the task naturally as a MIL problem. MIL generalizes the pattern recognition problem by making a significantly weaker assumption about the labelling information. Formally, there are two types of bags: positive $\boldsymbol{B_p}$ and negative $\boldsymbol{B_n}$. Each bag has an indefinite number of instances, $\boldsymbol{B} = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots\}$. When a bag is positive, it means that at least one instance in the bag is positive. Conversely, for a negative bag, all the instances in the bag must be negative. And the goal is to learn an binary classifier for instances $\boldsymbol{x}$. In our task, the sample refers to the bag. The subsequence in a sample refers to the instance in a bag, and the target sign refers to the positive instance.

When the instance-level binary classifier is configured as a 1-NN classifier, the training of this classifier is essentially to learn a short sequence that optimally separates the two classes of samples under the *subDis* distances. Now the problem of target sign finding has been transferred to the problem of shapelet discovering.

5.1 Shapelet searching

In theory, the shapelet can be any sequence shorter than the shortest training sample, then the search space can be infinite. For simplicity, we generally assume that the shapelet is a subsequence of training samples, and then, the shapelet discovering became the subsequence searching. Now, for a given word, the training set is defined as:

$$\mathbb{T} = T_{\text{pos}}^1, T_{\text{pos}}^2, \ldots, T_{\text{neg}}^1, T_{\text{neg}}^2 \ldots$$

where T_{pos}^i represent the ith positive sample, and T_{neg}^i is the ith negative sample. Each sample is a motion feature sequence.

5.1.1 Candidate subsequences

For subsequence searching, the candidate subsequence can be generated using a sliding window strategy. Lines 2–4 of Algorithm 1 show the generation process of all candidate subsequences with the length between $l_{\min}$ and $l_{\max}$, where $\mathbb{S}_T^l$ is the set of every subsequences in sample T with length l, as defined in Eq. 3. And the length range satisfies:

$$0 < l_{\min} \leq l_{\max} \leq \min(|T|), \quad \forall T \in \mathbb{T} \tag{6}$$

In fact, the search range of shapelet is small because only the reasonable lengths of real signs are taken into account. Additionally, only positive sample subsequences will be searched.

5.1.2 Score of subsequence

The *subDis* distance between a sample $T \in \mathbb{T}$ and a subsequence Q is calculated as $d = subDis(Q, T)$. Then, the training set $\mathbb{T}$ can be split into two subsets with a distance threshold d_σ. A sample T will be added to subset $\mathbb{T}_1$ if its *subDis* value $d \leq d_\sigma$, otherwise, it will be added to subset $\mathbb{T}_2$. Finally, a score function is created to assess the discriminating capacity Q.

$$\text{Score}(Q, \mathbb{T}) = \max_{d_\sigma}(I(\mathbb{T}) - I(\mathbb{T}_1, \mathbb{T}_2)) \tag{7}$$

$$= \max_{d_\sigma} I(\mathbb{T}) - \frac{|\mathbb{T}_1|}{|\mathbb{T}|} I(\mathbb{T}_1) - \frac{|\mathbb{T}_2|}{|\mathbb{T}|} I(\mathbb{T}_2) \tag{8}$$

$$Q^* = \arg\max_{Q} \text{Score}(Q, \mathbb{T}), \forall Q \in T, T \in \mathbb{T}_p. \tag{9}$$

where $I(\mathbb{T}) = -\sum_c p(c)\log(p(c))$ is the information entropy function, and c is the label of each sample. In practice, all the *subDis* distances can be sorted by magnitude first, then the optimal threshold d_σ can be approximated by searching the midpoint of all adjacent distances, which allows for fast calculation and without affecting the score. Theoretically, Q has the highest score when $\mathbb{T}_1$, $\mathbb{T}_2$ are identical to $\mathbb{T}_p$, $\mathbb{T}_n$, respectively. At this point, the subsequence Q^* with the highest score is actually the *shapelet* of $\mathbb{T}$.

5.1.3 Shapelet searching

In summary, the whole process of shapelet searching is shown in Algorithm 1. The inputs of this algorithm are the training set $\mathbb{T}$ of a *given word* and the possible length range $(l_{\min}, l_{\max})$ of *target sign*. Length range is a hyper-parameter that is influenced by regions, signers, words, moods, video fps(frames per second), etc. In this paper, the length range is set as $(0.5 * L_{\text{avg}},\ 2 * L_{\text{avg}})$ empirically, where L_{avg} is the average time span per word, is calculated as:

$$L_{\text{avg}} = \frac{\sum_i (f_i^{\text{end}} - f_i^{\text{begin}})}{\sum_i |f_i^{\text{text}}|} \tag{10}$$

where f_i^{begin},f_i^{end},f_i^{text} refer to the three components of the ith subtitle frame, and $|f_i^{\text{text}}|$ is the word count of the short text.

Algorithm 1 ShapeletSearching

Input: Samples $\mathbb{T}$, length range (l_{min}, l_{max})
Output: shapelet

```
1:  shapelet = none, best_score = 0
2:  for l in range(l_min, l_max) do
3:      for T in T_p do
4:          for Q in S_T^l do
5:              for j, T in enumerate(T) do
6:                  d_j = subDis(Q, T)
7:              end for
8:              temp_score = Score(Q, T)
9:              if temp_score ≥ best_score then
10:                 shapelet = Q
11:                 best_score = temp_score
12:             end if
13:         end for
14:     end for
15: end for
16: return shapelet
```

Then, the score of each candidate subsequence will be calculated using Eq. 7, and the subsequence with the highest score is identified as the *shapelet*. The score function is the core part of the shapelet searching algorithm, and the calculation of *subDis* distance is very time-consuming. Tricks like early abandoning, reuse of computation, parallel computing, and candidate filtering were adopted to speed up the searching process [52–55]. Algorithm 2 illustrates a sliding distance computing technique with python–numpy style pseudo-code. Using matrix operations, all distances between subsequence S and each sliding subsequence of the sequence T can be calculated simultaneously. With sliding distances, the *subDis* distance can be obtained as:

$$subDis(S,T) = \min(SldDists(S,T)) \tag{11}$$

In our experiment, the *SldDists* algorithm can improve the speed of shapelet searching by about 15–40 times compared with tricks of early abandon and entropy pruning.

Algorithm 2 SldDists

Input: sequence S, sequence T
Output: sliding distance

```
m, n = len(S), len(T)
b = n − m + 1
dist = (S[0] − T[: b])^2
for i in range(1, m) do
    dist = dist + (S[i] − T[i : i + b])^2
end for
dist = sum(dist, axis = −1)
return dist
```

5.1.4 Time complexity

For a given word, suppose there are $2n$ samples in the training set, and half of them are positive. If the average length of these samples is $\bar{m}$, then there are about $\bar{m}n$ candidate subsequences. For each candidate subsequence, $2n$ times of *subDis* calculations is needed, and the time complexity of the *subDis* function is $O(\bar{m}^2)$. In conclusion, the final time complexity of the shapelet searching algorithm is $O(\bar{m}^3 n^2)$.

5.2 Distance calculation acceleration

The *SldDists* function has greatly sped up the calculation of *subDis*. However, the current speed is still too slow to construct a large-scale sign language database. Inspired by the *matrix profile* in works [62, 63], we find that the *subDis* computation of candidate subsequences with adjacent locations and lengths involves a significant amount of repeated calculations. In this part, we greatly reduce the number of repeated calculations in shapelet searching by drawing on the calculation tricks in work [63].

Given two sequences A and B of lengths m_1 and m_2, the goal is to compute the distances between all subsequence pairs of A and B, denoted as distance matrices:

$$\begin{aligned} \boldsymbol{M} = \{M^l_{i,j} | \quad & 0 \le i \le m_1 - l, \quad 0 \le j \le m_2 - l, \\ & 1 \le l \le \min(m_1, m_2)\} \end{aligned} \tag{12}$$

where $M^l_{i,j}$ is the Euclidean square distance between subsequences $A_{i:i+l}$ and $B_{j:j+l}$.

$$M^l_{i,j} = \sum_{k=0}^{l-1} (A_{i+k} - B_{j+k})^2 = \sum_{k=0}^{l-1} d_{i+k,j+k} \tag{13}$$

5.2.1 Adjacent location subsequences

For the distance of the adjacent location subsequences: $M^l_{i+1,j+1}$, we have the following decomposition:

$$\begin{aligned} M^l_{i+1,j+1} &= \sum_{k=0}^{l-1} d_{i+1+k,j+1+k} = \sum_{k=1}^{l} d_{i+k,j+k} \\ &= M^l_{i,j} + d_{i+l,j+l} - d_{i,j} \end{aligned} \tag{14}$$

which means that we can calculate the distance of two subsequences based on the distance of its preceding subsequences, and we have a iterative update formula:

$$\boldsymbol{M}^l[i+1, 1:] = \boldsymbol{M}^l[i, :-1] + \boldsymbol{d}[i+l, l:] - \boldsymbol{d}[i, :-l] \tag{15}$$

where $\boldsymbol{d}$ is the Euclidean square distance matrix between each point in sequence A and each point in B. With the iterative formula 15, we only need to calculate $\boldsymbol{M}^l[0, :]$, $\boldsymbol{M}^l[:, 0]$, and $\boldsymbol{d}$ in advance. Then, the rest of distance matrix $\boldsymbol{M}^l$ can be calculated with just matrix addition. In fact, $\boldsymbol{M}^l[0, :]$ and $\boldsymbol{M}^l[:, 0]$ are sliding distances as described in Algorithm 2, and the $\boldsymbol{d}$ can also be seen as a special case of the sliding distance (the length of subsequence is 1). So they all can be accelerated using matrix operations as follow:

$$\boldsymbol{d}[i, :] = \sum_c (A[i] - B)^2 \tag{16}$$

$$\boldsymbol{M}^l[0, :] = \sum_{k=0}^{l-1} \boldsymbol{d}[0, k : k + m_2 - l + 1] \tag{17}$$

$$\boldsymbol{M}^l[:, 0] = \sum_{k=0}^{l-1} \boldsymbol{d}[k : k + m_1 - l + 1, 0] \tag{18}$$

where c in Eq. 16 represents the dimension number of features, and Eq. 16 does the same work as the line 5 in Algorithm 2.

5.2.2 Adjacent length subsequences

Next, let us look at the distance of the subsequences with adjacent length. When we change the length of subsequences to $l + 1$, the distance between two subsequences can be written as:

$$M^{l+1}_{i,j} = \sum_{k=0}^{l} d_{i+k,j+k} = M^l_{i,j} + d_{i+l,j+l} \tag{19}$$

In this way, we obtain the iterative formula of the distance matrix $\boldsymbol{M}$ with the adjacent subsequence length:

$$\boldsymbol{M}^{l+1} = \boldsymbol{M}^{l}[:\text{-}1, :\text{-}1] + \boldsymbol{d}[l:, l:] \tag{20}$$

Combing the distance matrix updating Formulas 15 and 20, we can get the algorithm for calculating the distances between all possible subsequences of two sequences as shown in Algorithm 3. For a distance matrix $\boldsymbol{M}^{l}$, each row of data represents the sliding distances between a subsequence of A and the sequence B. Then, the minimum value of the row of data represents the *subDis* between the subsequence and B.

$$\boldsymbol{M}^{l}[i] = \mathit{SldDists}(A[i : i + l], B) \tag{21}$$

$$\min(\boldsymbol{M}^{l}[i]) = \mathit{subDis}(A[i : i + l], B). \tag{22}$$

Algorithm 3 DistanceMatrices

Input: sequences A, B, length range (l_{min}, l_{max})
Output: DisMats

$m_1 = len(A)$, $m_2 = len(B)$
$\mathcal{M} = \emptyset$, $\boldsymbol{d} = zeros(m_1, m_2)$
for i in range(m_1) **do**
    $\boldsymbol{d}[i, :] = \sum_c (A[i] - B)^2$
end for
$l = l_{min}$
$\boldsymbol{M}^{l} = zeros(m_1 - l + 1, m_2 - l + 1)$
$\boldsymbol{M}^{l}[0, :] = \sum_{k=0}^{l-1} \boldsymbol{d}[0, k : k + m_2 - l + 1]$
$\boldsymbol{M}^{l}[:, 0] = \sum_{k=0}^{l-1} \boldsymbol{d}[k : k + m_1 - l + 1, 0]$
for i in range$(m_1 - 1)$ **do**
    update $\boldsymbol{M}^{l}[i + 1, 1:]$ with Eq.15
end for
$\mathcal{M} = \mathcal{M} \cup \boldsymbol{M}^{l}$
for l in range(l_{min}, l_{max}) **do**
    update $\boldsymbol{M}^{l+1}$ with Eq.20
    $\mathcal{M} = \mathcal{M} \cup \boldsymbol{M}^{l+1}$
end for
return $\mathcal{M}$

5.2.3 Improved shapelet searching

We can efficiently obtain the *subDis* distances between all subsequences of two sequences by employing iterative updating of the adjacent distances. In shapelet searching, to calculate the *score* for each candidate subsequence, the *Distance Matrices* between each positive sample and all samples should be computed in advance.

$$\mathit{DistanceMatrices}(T_a, T_b), \quad \forall\, T_a \in \mathbb{T}_p, T_b \in \mathbb{T}$$

In fact, rather than iterating over all sample pairs as described above, a more efficient method is to concatenate all positive samples as a single sequence T_A and all training samples (both positive and negative) as a single sequence T_B. Then directly input T_A and T_B to the algorithm *DistanceMatrices*, and the distance matrix of one pair $< T_a, T_b >$ is the sub-matrix of the distance matrix of $< T_A, T_B >$ and can be directly obtained by slicing.

However, there are two practical limitations to this concatenation method. First, as shown in Algorithm 3, at least two matrices $\boldsymbol{d}$ and $\boldsymbol{M}$ need to be constructed. It was supposed that we have 200 training samples (100 positives) with an average length of 400 and the storage data type is *float*32. Then, more than $23GB$ memory usage is needed, and the size will grow exponentially with the number of samples. Second, for the iteration formula 15, the matrix addition will accumulate the error of each iteration. The error can be remarkable when the length of the sequence T_A is too large.

Finally, as shown in Algorithm 4, we get an improved shapelet searching algorithm based on distance matrices. In the algorithm, we first obtain the distance matrices $\mathcal{M} = \{\boldsymbol{M}^{l} | l \in [l_{\min}, l_{\max}]\}$ between each positive sample T_a and concatenated training sequence T_B. Now, the r_{th} row of a distance matrix $\boldsymbol{M}^{l}$ represents the *sliding distances* between a candidate subsequence $T_a^i[r : r + l - 1]$ and T_B. After that, by slicing $\boldsymbol{M}^{l}[r]$ in accordance with where the training sample T_b is located in T_B, the sliding distances between the subsequence and T_b is determined, and the minimum of sliding distances is *subDis*. Once we obtained its *subDis* distances with all the training samples, we could use Eq. 7 to calculate the score of the subsequence. Finally, the subsequence with the largest score will be returned as *shapelet*. If there are two subsequences with the same score, the one with the smaller standard deviation (std) of *subDis* distances will be chosen.

Algorithm 4 ImprovedShapeletSearching

Input: samples $\mathbb{T}$, length range (l_{min}, l_{max})
Output: shapelet

$T_B = concatenate(\mathbb{T})$
$best\_score = 0, \quad min\_std = \infty$
for T in $\mathbb{T}_p$ **do**
  $\mathcal{M} = DistanceMatrices(T, T_B, l_{min}, l_{max})$
  for M^l in $\mathcal{M}$ **do**
    for r in $range(len(M^l))$ **do**
      $dists = \text{zeros}(|\mathbb{T}|)$
      for j, T_b in enumerate($\mathbb{T}$) **do**
        $slddis = slicing(M^l[r], T_b)$
        $dists[j] = \min(slddis)$
      end for
      $score = \text{Score}(dists, \mathbb{T})$
      $std = \text{std}(dists)$
      if $score > best\_score$ **then**
        $shapelet = T[r : r + l]$
        $best\_score = score$
        $min\_std = std$
      end if
      if $score == best\_score$ **then**
        if $std < min\_std$ **then**
          $shapelet = T[r : r + l]$
          $best\_score = score$
          $min\_std = std$
        end if
      end if
    end for
  end for
end for
return shapelet

5.3 Shapelet net

In contrast to the above brute force searching strategy, another option of shapelet discovering is to remove the restriction that only searches the shapelet from all candidate subsequences, treating the shapelet as a sequence with unknown parameters to learn. Paper [57] is one of the original papers to propose this idea, and this subsection provides a detailed introduction to the idea. When evaluating a shapelet Q as an unknown sequence, we should first calculate the *subDis* distances between Q and all training samples, and then determine the linear separation with maximum information gain using the *subDis* distances.

For simplicity, the *subDis* distance between Q and a training sample T is denoted as X, and then its expression (Eq. 2) is expanded in the following.

$$X = \min_j Dis(Q, T_{j:j+l}) \tag{23}$$

$$Dis(Q, T_{j:j+l}) = \sum_{k=0}^{l-1} (Q_k - T_{j+k})^2 \tag{24}$$

$$= \sum_{k=0}^{l-1} Q_k^2 + \sum_{k=0}^{l-1} T_{j+k}^2 - 2\sum_{k=0}^{l-1} Q_k T_{j+k} \tag{25}$$

$$= Q \cdot Q + I \cdot T_{j:j+l}^2 - 2Q \cdot T_{j:j+l} \tag{26}$$

where the *dot operate* indicates the inner product, and the square $(\cdot)^2$ is a point-wise operation.

When j is varying from 0 to $|T| - l$, the term $Q \cdot Q$ is irrelevant to j and will remain constant. The terms $I \cdot T_{j:j+l}^2$ and $Q \cdot T_{j:j+l}$ will become the convolution operations of $I \otimes T^2$ and $Q \otimes T$. Therefore, we have a new calculation formula of the sliding distances:

$$SldDist(Q, T) = Q^T Q + I \otimes T^2 - 2Q \otimes T \tag{27}$$

And if the linear separation of the training set can be represented by a linear binary classifier, then the classification result for sample T can be written as:

$$\hat{Y} = \omega X + \omega_0 \tag{28}$$

where $\hat{Y}$ is the predicted target of positive and negative, the (ω, ω_0) are the weights. Then, the logistics loss can be used to evaluate the performance of the classifier:

$$\mathcal{L}(Y, \hat{Y}) = -Y \ln \delta(\hat{Y}) - (1 - Y) \ln(1 - \delta(\hat{Y})) \tag{29}$$

where $\delta(\hat{Y}) = (1 + e^{-\hat{Y}})^{-1}$.

Up to now, the above process of *subDis* computation, sample classification can be represented as a *Shapelet Net* shown in Fig. 3. The shapelet kernel and the binary classifier are the unknown parameters to be learnt. When a

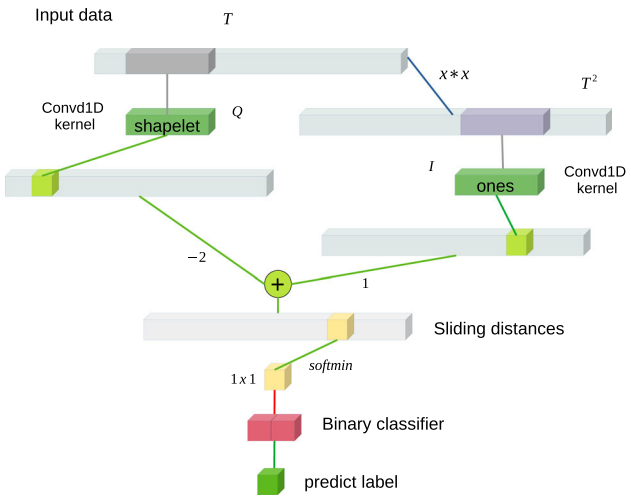


Fig. 3 The structure of the shapelet Net

sample T is entered, a predicted label is returned. It needs to be noted that the term Q^TQ in Eq. 27 is ignored because it has no impact on how the samples are classified. Then, with a training set $\mathbb{T}$ and its labels $\boldsymbol{Y} = [Y_1, \ldots, Y_{|\mathbb{T}|}]$, the optimal shapelet Q and linear hyper-plane $\boldsymbol{\omega}$ can be learned by minimizing a the regularized objective function, written as $\mathcal{F}$.

$$\mathcal{F}(Q, \boldsymbol{\omega}) = \sum_{i=1}^{|\mathbb{T}|} \mathcal{L}(Y_i, \hat{Y}_i) + \lambda \|\boldsymbol{\omega}\|^2 \tag{30}$$

There are two issues that arise during the training of shapelet Net. First, the $\min(\cdot)$ function in Eq. 23 is not derivable, which makes it impossible to use gradient descent method in the training. This can be solved by replacing the minimum function with a derivative *softmin* function as follows:

$$softmin(\boldsymbol{x}) = \frac{\sum_i x_i e^{\alpha x_i}}{\sum_j e^{\alpha x_i}} \tag{31}$$

where $\alpha < 0$ is the parameter that controls the *precision* of the *softmin*, and when $\alpha \to -\infty$, the *softmin* is nearly equal to the true minimum function.

Second, the objective function given by Eq. 30 is not a convex function, and the gradient-based optimization converges usually to a local minimum. To get a reasonable suboptimal result, the learning parameters must be initialized well. The shapelet parameters in the works [57, 59] were initialized with clustering centres of candidate subsequences. However, since our work only involves one shapelet, employing cluster centres for initialization makes it simple to converge to a non-target sign result. Therefore, we finally choose to use the shapelet searched from a small subset of training samples by Algorithm 1 as our initial parameters.

Compared with shapelet searching, shapelet Net no longer discovers shapelets by brute force searching but learns the shapelet as unknown parameters through network training. The shapelet Net brings two changes: the first is the computational complexity. Shapelet Net has the complexity of $O(\bar{m}^2 n \times Ite_{\max})$ [57] instead of the $O(\bar{m}^3 n^2)$ of shapelet searching. In general, $Ite_{\max} << \bar{m}n$. The second is that, rather than being a true candidate subsequence, the learned shapelet is more akin to the generalization of the best candidates. which makes it theoretically more representative than the searched shapelet.

5.4 Sample augmentation

The target sign for a given word can be assumed to be the shapelet that was acquired through shapelet searching or shapelet Net. With the shapelet, more instances of the target sign can be gathered by collecting the subsequence at the shortest distance with the shapelet in each positive sample. However, the subtitles are weak and noisy as introduced in Sect. 4.2. There are uncertain differences in time displacement, occurrence, and order between subtitles and signing. For a given word, its positive sample does not guarantee that it contains a target sign, and its negative sample may also contain a target sign. Thus, two tasks need to be done: removing fake target signs from positive samples and identifying latent target signs from negative samples.

For a shapelet Q, the target signs from all positive samples can be identified using the following formula:

$$S_i = \arg\min_S Dis(Q, S), \quad \forall S \in \mathbb{S}_{T_{pos}^i}^{|Q|} \tag{32}$$

According to the paper [45], only 67% of their positive samples contain true target signs. In our data resource, the true rate of positive samples is even less than 60%. Now, we set the true rate as f, and only the proportion f of the most similar target signs to the shapelet are considered to be true target sign. Without losing generality, we assume that the identified target signs from all positive samples were sorted in increasing order based on their distance from Q. Then, the chosen true target signs are represented as: $S_1, S_2, \ldots, S_{n_f}$ satisfying:

$$\|Q - S_1\| \le \|Q - S_2\| \le \ldots \le \|Q - S_{n_f}\| \tag{33}$$

where $n_f = \lfloor f \times |\mathbb{T}_p| \rfloor$, and $\lfloor \cdot \rfloor$ is the floor round function. Then, the threshold $\tau = \|Q - S_{n_f}\|$ can used to judge whether a target sign is true or not:

$$S' = \begin{cases} \text{target sign} & \|Q - S'\| \le \tau \\ None & \|Q - S'\| > \tau \end{cases} \tag{34}$$

When the number of positive samples is small, however, τ has a limited number of candidate values and is not sensitive to variations in f. An additional factor θ is proposed to allow for finer and smoother adjustment of τ. First we denote $d_{\min} = \|S - S_1\|$, $d_{\max} = \|S - S_{n_f}\|$. If $\|S - S_1\| = 0$, $d_{\min}$ will be replaced with the smallest nonzero distance. Then, the distance threshold τ is redefined as:

$$\tau = d_{\min} + \theta(d_{\max} - d_{\min}) \quad 0 \le \theta \le 1 \tag{35}$$

In order to obtain more instances, we propose a sample augmentation strategy that collects all subsequences satisfying Eq. 34 in all videos as target signs. Since we have at least 1000 words and over 96 h of sign language videos, an efficient sliding distance calculation is one of the cores of sample augmentation. Two sliding distance calculation methods have been proposed in our work: Algorithm 2 and formula 27. Formula 27 transforms the sliding distance into a combination of convolution operations.

The convolution theorem states that *the convolution in the time domain equals the point-wise multiplication in the frequency domain*. Due to the fact that point-wise multiplication is much faster than convolution, we adopt the *SlidingDotProduct* algorithm proposed in the paper [62] to calculate the convolution operations. The details of the algorithm are shown in Algorithm 5, which transforms between time and frequency domains by forward and inverse Fast Fourier transforms(FFT and IFFT). Before performing the FFT, the input sequences must be padded and reversed to ensure the following convolution is essentially produced in the right order. The time complexity of FFT and IFFT is $O(n \log n)$, and it can be calculated at a very high speed with many available mathematical libraries. According to our testing, the sliding distance calculation based on Eq. 27 plus Algorithm 5 is faster than Algorithm 2 by **4–5** times, whether running on CPUs or GPUs.

Algorithm 5 SlidingDotProduct(S,T)

Input: A query sequence S, long time series T
Output: the sliding dot products

$m \leftarrow len(S)$, $n \leftarrow len(T)$
$T_a \leftarrow$ Append T with n zeros
$S_r = Reverse(S)$
$S_{ra} \leftarrow$ Append S_r with $2n - m$ zeros
$S_{raf} \leftarrow FFT(S_{ra}), T_{af} \leftarrow FFT(T_a)$
$ST = IFFT(S_{raf} \cdot T_{af})$
return $ST_{m:n}$

Algorithm 6 shows the details of our final sample augmentation strategy. For a given word, all the subsequences with a distance less than τ from the shapelet in all video sequences are collected as instances of the target sign. The distance threshold τ is determined by Eq. 35. And the sliding distances between the shapelet and a long video sequence are calculated by combining Eq. 27 and Algorithm 5. However, it is important to note that the surrounding subsequences of a target sign subsequence may also be very similar to the shapelet. Therefore, in lines 10–17 of the algorithm, we need to make sure the collected instances have a lower distance from the shapelet than their neighbours.

Algorithm 6 SampleAugmentation

Input: video seqs $\mathbb{A}$, shapelet S, threshold τ
Output: instances of target sign

1: $m = len(S)$, $\mathcal{C} = \emptyset$
2: **for** A in $\mathbb{A}$ **do**
3: calculate $I \otimes A^2$, $S \otimes A$ with Alg.5
4: $\boldsymbol{d} = S^T S + I \otimes A^2 - 2S \otimes A$
5: *# Initialize the candidate sample*
6: $i_0, d_0 = 0, +\infty$
7: **for** i, d in enumerate($\boldsymbol{d}$) **do**
8: **if** $d \leq \tau$ **then**
9: *# Judge the time interval*
10: **if** $i - i_0 \leq \frac{m}{2}$ **then**
11: **if** $d < d_0$ **then**
12: $i_0, d_0 = i, d$
13: **end if**
14: **else if** $d_0 \neq +\infty$ **then**
15: $\mathcal{C} = \mathcal{C} \cup A[i_0 : i_0 + m]$
16: $i_0, d_0 = i, d$
17: **end if**
18: **end if**
19: **end for**
20: *# save the last candidate*
21: $\mathcal{C} = \mathcal{C} \cup A[i_0 : i_0 + m]$
22: **end for**
23: **return** $\mathcal{C}$

6 Experiments

Two main experiments will be evaluated in this section. First, we will show how the proposed framework can automatically construct a large-scale word-level sign language database from sign language-interpreted videos. The experimental environments and parameter settings will be described in detail, and the results of two shapelet methods (*shapelet searching* and *shapelet Net*) will also be compared and discussed. Second, three cutting-edge SLR methods are utilized to evaluate the database in order to demonstrate the practicability of our created database. Additionally, the sample augmentation strategy and the effects of various shapelet methods will be examined.

6.1 Original data source

In order to avoid the impact of language differences in different regions and fields, we collected the sign language

videos with similar occasions and themes. Eighty-nine Scottish Parliament's live sign language translating videos are downloaded from YouTube [64]. These real-time sign language translations are supported by *The Scottish Parliament's BSL Plan 2018–2024*.[2] Our downloaded 89 videos are interpreted by 8 native British sign language (BSL) signers and have a total video duration of about 96 h. The original downloaded sign language videos had no subtitles, and then, we generated individual subtitle files for these videos using the *English speech recognition tool* provided by YouTube. There are about 32,000 meaningful and meaningless words in all these subtitle files.

After deleting the meaningless words, stop words, polysemy words, and low-frequency words, the remaining words will be processed with *stemming* and *lemmatization* methods. Finally, we chose 1000 words with good properties to form the final corpus of our target database. The Fig. 4 demonstrates the frequency distribution of these words. In the figure, the words appear roughly 10 to 800 times in the subtitles (the minimum frequency is set to 10). However, the frequency distribution shows a significant imbalance: 80% of the words have fewer than 213 occurrences and 95% of the words have fewer than 553 occurrences. The average frequency of all words is 210, while the average frequency of the least 80% of words is only 67.

6.2 Motion feature extraction

The skeleton keypoints of the signers in all sign language-interpreted videos were extracted using the *openpose* library. The provided models (body and hand) that trained with *CMU Panoptic Dataset*[3] can be directly applied to new videos without parameters retuning. In our work, the models were loaded by the *pytorch* and evaluated on GPUs. The body keypoints were first extracted from the cropped regions containing the signers. Then, the hands keypoints were extracted from the regions that determined by the extracted body points (i.e. shoulder, elbow, wrist) [36]. It took us about a week to extract the skeleton keypoints of all 96 h of sign language video using a single *GeForce-RTX-3090* GPU.

In our work, only the keypoints of the upper body and hands were selected to construct the motion feature using Eq. 4. Before the feature construction, the coordinates of the chosen keypoints will be filtered using a median filter to remove outliers. The invalid coordinates (i.e. the position of non-detected keypoints) will remain unchanged as (0, 0). In addition, our work took into account the effects of left-handedness and right-handedness, and for each motion feature subsequence, an additional *x-mirror* of the motion feature subsequence (multiplying all x coordinates by -1) will be provided.

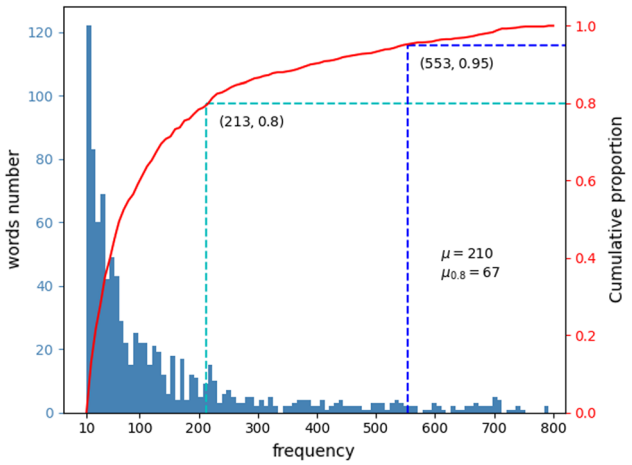


Fig. 4 The frequency distribution of the final corpus

6.3 Shapelets learning

As Sect. 4.3, the positive and negative samples were generated for each word in the final corpus. The candidate windows were determined using the Eq. 5, and the parameter β was set to 1.5 by trial. In this case, most of the ground truths of the target signs can be contained in the positive sequences.

6.3.1 Ground truths

To be able to evaluate the shapelet methods, the ground truths of the target signs should be provided. In this paper, we selected 99 words and then annotated 10 to 30 positive samples of each word. To make the annotation more appropriate, most of the selected words should have a clear meaning, and the corresponding examples should be available on the Internet. If a positive sample contains the *target sign* of the word, the sample is called a *true positive sample* and the indexes of the begin and end frames of the sign were recorded. Otherwise, the sample is called a *false positive sample*. From the annotations of these 99 words, we can deduce that the average *truth rate* of these positive samples is about 60%, which means that about 40% of the positive samples do not contain a target sign. And the time span of the sign is typically 10 to 40 frames.

6.3.2 Speed test

With the training samples of a given word, the shapelet can be discovered with two shapelet methods (*Searching* and *Net*). In order to show the effectiveness of the *distance calculation acceleration* discussed in Sect. 5.2 and

[2] https://external.parliament.scot/help/109625.aspx.

[3] http://domedb.perception.cs.cmu.edu/.

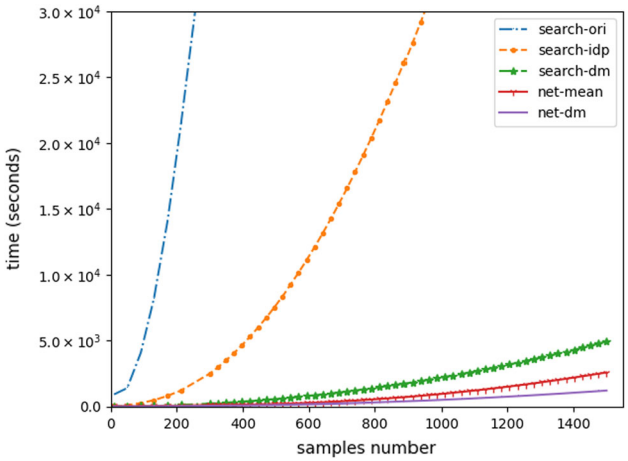


Fig. 5 The time-consuming comparison

compare the speed of different available shapelet methods, Fig. 5 illustrates the time consumption of different shapelet discovering methods with different sample numbers.

Five shapelet methods are compared in Fig. 5, including three shapelet searching methods and two shapelet Net methods. The *search-ori* is the baseline shapelet searching algorithm proposed in the paper [51], which uses a two-level pruning strategy to abort the hopeless candidates early. The *search-idp* is the algorithm proposed in the paper [55] that uses *sampling* and *filtering* strategies to greatly reduce the number of candidate shapelets. However, since the labels of our time series are weak and noisy, and the length and content are also varied a lot, the *sampling* strategy is not used here, only the *filtering* strategy is adopted. In addition, our proposed algorithm 2 is also used to speed up the calculation of the *search-idp*. The *search-dm* is the improved shapelet searching algorithm 4, which significantly reduces the repetition in the calculation of *subDis* distances by the proposed *distances matrices*. *net-mean* and *net-dm* are the proposed shapelet learning algorithm with two different parameter initialization methods. The *net-mean* uses the average values of all the candidate shapelets for initialization [57, 59], while the *net-dm* uses the shapelet obtained by *search-dm* as the initial shapelet. In the test, an equal number of positive and negative samples were chosen randomly. We fixed five candidate lengths for the shapelet: 15, 20, 25, 30, 35. Meanwhile, the length range of *search-idp* is also set to [15, 35]. In addition, the time consumption of the *net-dm* method involves the time consumed for the shapelet initialization with 10% samples.

From the experimental results in Fig. 5, we can find that compared to the baseline method *search-ori*, the other four shapelet discovering methods can greatly reduce the calculation time, and the *net-dm* has the highest learning speed. Among them, the time consumption of methods *search-ori*, *search-idp* and *search-dm* are roughly increasing squarely, which is consistent with the idea of brute force searching. The two *net* methods have similar learning speeds, and their time consumption increases roughly linearly. The fact that *net-dm* is faster than *net-mean* is probably due to the convolutional network can converge faster with a more reasonable initialization. And the reason why *search-dm* is not as efficient as method *net* is that although *search-dm* is very efficient in computing sliding distances (the time consumption of a new distance matrix calculation based on Eq. 20 is almost negligible), the time consumption of both the initialization of *distance matrix* and the calculation of *score* also increases exponentially with the number of training samples.

6.3.3 Recall rate test

The performance of the four shapelet methods (search-dm, search-idp, net-mean, net-dm) on the annotated set of 99 words was evaluated in this experiment. The maximum number of training samples for the *search-* and *net-* methods is set to 100 and 600, respectively, to allow for the learning of large-scale shapelets to be completed in a reasonable amount of time. Moreover, according to Eq. 10, we set the final shapelet length range to $(10,\ 40)$. And the commonly used metric *recall rate* [38, 45, 46] was adopted to measure the correctness of discovered shapelets. It is considered to be *identified* correctly if the overlap rate between an identified sign and the ground truth is more than 50%. Similarly, a word is considered to be correctly *recalled* if more than 50% of the target signs are correctly identified. And the *recall rate* indicates the proportion of the correctly recalled words.

The overlap rate, identification rate, and recall rate of the four shapelet methods are compared in Tables 1 and 2 under different conditions. The best results for each rate under specific conditions are bolded for intuitive method comparison. Firstly, we can find that the true ratio and the number of positive samples have a significant impact on the learning results. The higher true ratio means the higher rate of overlap, identification, and recall. This is actually predictable because the smaller the true ratio is, the more the shapelet discovering deviates from the assumptions of the MIL problem. At the same time, the larger the number of training samples, the higher the rate of overlap, identification, and recall, which indicates that more training samples can effectively enhance the ability of shapelet discovering methods to learn true target signs.

Furthermore, we discovered that the *net-mean* method produces the worst learning results, demonstrating two facts: first, a good initialization of the shapelet net is critical, and second, the strategy of mean-valued initialization works poorly in our task. Overall, the *search-idp* method is

Table 1 The average overlap rate and identification rate

Opts	Conditions	Words	Overlap rate				Identification rate			
			dm	idp	net-mean	net-dm	dm	idp	net-mean	net-dm
True ratio	>0.5	83	0.5010	0.3915	0.2325	**0.5428**	0.5379	0.4171	0.2432	**0.5819**
	<0.5	16	0.2427	**0.2873**	0.2080	0.2823	0.2803	**0.3283**	0.2089	0.2955
Sample num	>50	59	0.5266	0.4831	0.2747	**0.5834**	0.5667	0.5168	0.2927	**0.6280**
	<50	40	0.3920	0.2113	0.1557	**0.4058**	0.4238	0.2272	0.1515	**0.4316**
Total[a]	–	99	0.4759	0.3814	0.2302	**0.5165**	0.5128	0.4084	0.2398	**0.5541**

[a]The averaging is performed at samples level

Table 2 The recall rate

Opts	Cond	Words	Recall rate			
			dm	idp	net-mean	net-dm
True ratio	>0.5	83	0.5783	0.4217	0.1687	**0.6506**
	<0.5	16	0.1250	**0.3125**	0.1250	0.1875
Sample num	>50	59	0.6271	0.5763	0.2373	**0.6780**
	<50	40	0.3250	0.1500	0.0500	**0.4250**
Total	–	99	0.5051	0.4040	0.1616	**0.5758**

worse than the *search-dm* because it will filter out good candidate shapelets compared to the all-search strategy of *search-dm*. However, we also find that *search-idp* performs best in the case of true ratio <0.5, which is probably because the filtering strategy can also remove a lot of interference candidates under noisier conditions. The *net-dm* method performs best under nearly all conditions. This is probably because the *net-dm* is initialized with searched shapelets, which is actually equivalent to a two-stage learning model. The first stage is the rough learning process using *search-dm* on a small subset of samples, while the second stage removes the restrictions on the shapelet values and fine-tunes the rough shapelets.

Based on the results of the above recall rate test, only the *search-dm* and *net-dm* methods will be used in the following database construction, and the terms of *search* and *Net* in the subsequent section refer to *search-dm* and *net-dm*.

6.4 Database construction

6.4.1 Signs collection

In this section, we applied two best shapelet methods (i.e. *search-dm* and *Net*) to the whole corpus with the same parameter settings as in the above subsection. Then, the discovered shapelets will be used to collect the instances of target signs.

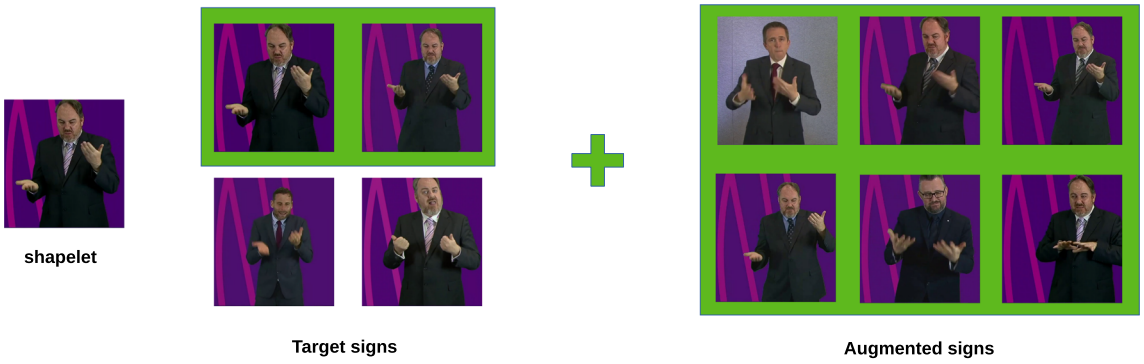
Figure 6 shows the process of signs collection for a given word "balance". Given a discovered shapelet, the *target signs* are identified from the positive samples first. Here, we believe that the target signs closer to the shapelet (with green background) have higher confidence of identification, and they will be used as the criterion to judge whether a subsequence similar to the shapelet or not. Finally, the additional similar subsequences will be collected as *augmented signs*. Figure 6a, b show the signs collection process based on shapelet searching and shapelet Net, respectively. The only difference between them is the shapelets. As shown in figure, the shapelet discovered by shapelet searching is an actual existing motion clip, while the shapelet learned by shapelet Net is a parameterized pose sequence without a corresponding actual motion clip.

6.4.2 Four subsets

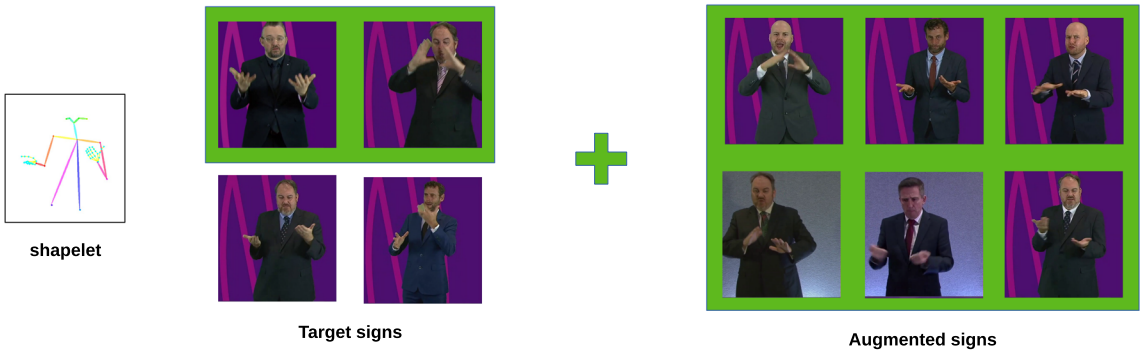
Finally, we constructed a Scottish Parliament British Sign Language database, SPBSL, which includes four large-scale sign language subsets based on different shapelet methods and sample strategy. That is: *no_aug-search, no_aug-net, aug-search, aug-net*, where *search* and *net* correspond to two shapelet methods, while *no_aug* and *aug* indicate whether the samples contain the augmented signs or not. The databases, models, and code are available at our project page.[4]

The construction strategy of the *no_aug* sign language databases is to only collect basic target signs (localized from the positive samples) as samples. This strategy has

[4] https://github.com/hitmaxiang/SPBSL.



(a) signs collection based on search-based shapelet



(b) signs collection based on net-based shapelet

Fig. 6 The signs collection illustration of the word "balance"

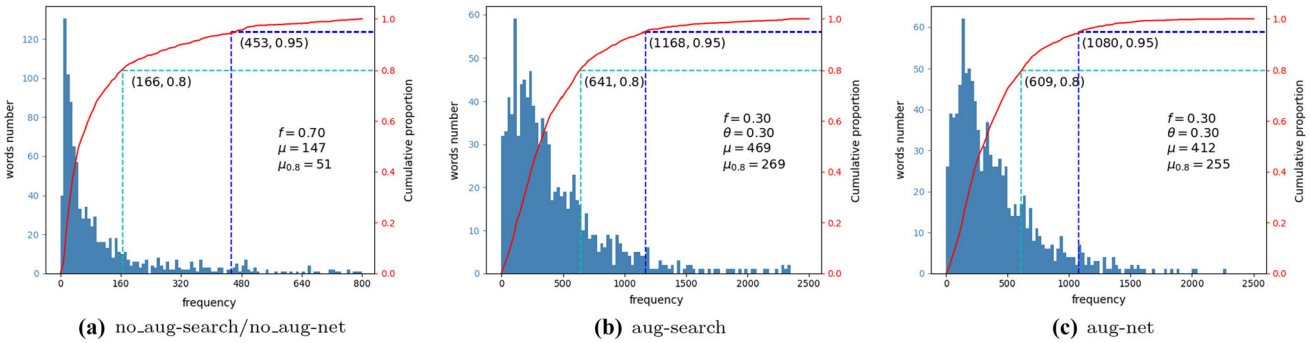
been adopted in many studies [38, 45, 46]. Based on our analysis and experiments on the annotated words, we know that the *true ratio* of positive samples is about 0.6, while the proportion of true positive samples being correctly localized (i.e. the identification rate) is only about 0.5. As a result, when building the non-augmented sign language database, we only keep the f ratio of nearest target signs as samples according to Eq. 33. With the test tuning, there is a better balance between the number and confidence of the non-augmented sign language database when f is set to 0.7.

The augmented sign language databases are constructed by collecting similar instances (the signs with green backgrounds in Fig. 6) of the shapelet from the whole data source as samples. According to Algorithm 6, an instance is considered a sample if its distance from the shapelet is less than the threshold τ and it has the shortest distance among its neighbours. The two-level conditioning function Eq. 35 determines the threshold τ, and the parameters f and θ are both set to 0.3 in our construction.

Figure 7 illustrates the sample distributions of these four sign language databases. Since the number of samples in the non-augmented database is completely determined by the number of positive samples and the parameter f, the sample distributions of the non-augmented databases based on two shapelet methods (*no_aug-search* and *no_aug-net*) are identical and are represented commonly as Fig. 7a. The sample distributions of the two augmented databases are shown in Fig. 7b, c. As shown in the figure, the sample augmentation strategy can significantly increase the number of samples. Compared with the original database, the *aug-search* and *aug-net* databases showed a 3.2 and 2.8 times increase in the mean number of samples, and a 5.3 and 5 times increase in the mean number of samples of the 80% smallest-sized classes, respectively. In summary, sample augmentation can both increase the number of samples and improve the balance of the database (the distribution is flatter).

6.4.3 Test sets

To evaluate the four constructed sign language databases and compare the impact of different shapelet methods and sample strategies, we will give the construction method of the corresponding test set here: The test set is designed as a collection of target signs that satisfy the condition of Eq. 35, where both factors f and θ are also set to 0.3. First, the smaller of the f makes the test set have higher identification confidence. Second, as a common subset of the augmented and non-augmented datasets, the test set can be used to evaluate the sample augmentation strategy. In our



(a) no_aug-search/no_aug-net **(b)** aug-search **(c)** aug-net

Fig. 7 The occurrence distributions of four constructed databases

experiments, two test sets were constructed corresponding to two shapelet methods.

6.5 Database evaluation

6.5.1 Compared methods

In this work, we chose three state-of-the-art SLR methods to evaluate the four subsets of SPBSL. They are I3D [11], GRU [9], and TGCN [10]. Their structures are shown in Fig. 8.

I3D is a spatio-temporal CNN architecture that takes a multiple-frame video as input and outputs class probabilities over sign categories. We adopt the I3D architecture due to its success in action recognition benchmarks. The original I3D network was trained on ImageNet [65] and fine-tuned on Kinetics-400 [11]. To apply I3D to our SLR task, a common approach is to treat the original trained model as a pre-training model, then modify the final linear classification layer to match the number of classes. Finally, the model is further fine-tuned using the training data.

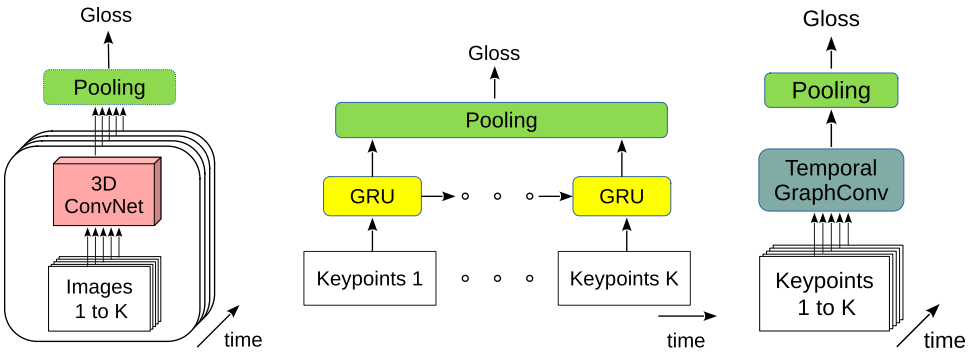
GRU is a classical architecture for motion recognition that has good modelling ability for sequence data. In this paper, the designed motion features were concatenated and then fed to a stacked GRU of 2 layers. In practice, we can adjust the fitting ability of the model by changing the hidden layer size.

TGCN is an architecture proposed in the paper [10] that stacks multiple residual graph convolution blocks and takes the average pooling result along the temporal dimension as the feature representation of pose trajectories. In the first layer, the networks take as input the $K \times 2T$ matrix coordinates of body keypoints, where K is the number of keypoints and T is the number of frames per sample. As above, the coordinates of the keypoints also need to be processed by Eq. 4. Then, a softmax layer followed by the average pooling layer is employed for classification.

6.5.2 Implementation details

The above three recognition models are implemented using *pytorch*. Our experiment settings basically follow the training configurations in the paper [10]. For the input video frames of I3D, the original video frames will be resized to 256 × 256 first, then we randomly crop a 224 × 224 patch from an input frames and apply horizontal flipping with a probability of 0.5. For the hidden layer size of the GRU, it is empirically set to 64 when the number of classes is less than 500, and 128 when it is greater than (equal to) 500. And the same setting is used for the hidden layer size of TGCN, and the stacking number of residual graph convolution blocks in TGCN is set to 24.

According to the average length of our manually annotated sign samples, the time length of the input



(a) Image 3D Conv **(b)** Pose GRU **(c)** Pose TGCN

Fig. 8 The structures of three state-of-the-art sign language classifiers

samples to all three models is set to 20. When the frame number of an original sample is greater than 20, input frames are chosen using uniform sampling. When the frame number of an original sample is less than 20, the input frames will be padded forward or backwards randomly using boundary frames. Finally, the *Adam* optimizer is chosen to minimize the cross-entropy for all three models.

Before the database evaluation, different sized (100, 300, 500, 1000) subsets of the four sign language datasets were constructed first, and then, each subset was evaluated using three recognition models (i.e. I3D, GRU, and TGCN). In order to keep the number of samples balanced among different classes, the maximum number of samples for each class is set at 200. In the training, the ratio of the number of training and validation samples is set to 3:1. The training process will be terminated when the performance of the validation samples no longer increases. And the two test sets designed in Sect. 6.4.3 were used to evaluate the performance of trained models. We use top-k classification accuracy with $k = \{1, 5, 10\}$ to evaluate the performance of the models on databases. In SLR, many misclassifications can be corrected with contextual knowledge. Therefore, it is more reasonable to choose top-k accuracy to evaluate the word-level SLR.

7 Discussion

7.1 Impacts of video parameters

The impacts of video parameters on database construction are qualitatively discussed in this subsection. In general, the higher the quality of the video resources, the higher the quality of the constructed database. In general, high quality video requires a high frame rate, high spatial and contrast resolution, etc. Fast and little movements can be captured well with a high frame rate, while more details (e.g. face, handshape) can be clarified with a high spatial resolution. Additionally, a high contrast resolution makes it easier to identify the ROI. In our study, first, high quality videos can assist the extraction of motion features with high clarity. Then, the higher confidence shapelet can be discovered with more accurate features and thus influence the constructed database based on the shapelet. Furthermore, the database is essentially a collection of video clips, and good video parameters can also make it easier for various classifiers to recognize them. In the end, it is worth noting that the input images for pose extractors and sign recognizers will be resized to a small fixed size (e.g. 256×256). Therefore, it is unnecessary to seek a very high spatial resolution, and the input images typically need to be cropped to ensure a high ROI ratio.

7.2 Performance evaluation of networks

The recognition accuracy of the three models on all database subsets is shown in Table 3. The highest recognition accuracies for different size databases under each classifier are bolded to represent the impact of database size. Overall, models I3D and TGCN have better performance than GRU, which is consistent with the experimental results of the paper [10]. However, what is not inconsistent is that the classification accuracy of I3D is lower than TGCN, and it performs even worse than GRU on non-augmented databases with class numbers below 300. We argue for two reasons behind this inconsistent phenomenon. First, since I3D is more complex and larger than the other two models, it tends to be overfitting when the number of training samples is small. For the non-augmented databases, the deficit of samples makes I3D poor and unstable (the classification accuracy of I3D on the non-augmented databases of 500 classes is higher than that on the non-augmented databases of 300 and 100 classes). And the better and more stable performance of I3D on the augmented database also revealed that the deficiency of training samples of the non-augmented database is the main reason for the poor performance of I3D. Secondly, since our databases are constructed based on shapelets that learned with extracted pose features, the image-based I3D does not perform as well as the pose-based TGCN.

7.3 Effects of shapelets and sample augmentation

As shown in Table 3, the performance of the databases based on different shapelet methods is similar for the three models. The evaluation results are almost the same for the augmented databases. Overall, the classification accuracy of the net-based databases is slightly higher than that of the search-based databases. Compared to the slight impact of different shapelet methods, the sample augmentation strategy improves the recognition accuracy of the models on the databases significantly (about 20% on average), and the top-1 accuracy of GRU, TGCN, and I3D on the augmented databases can be improved by 33%, 10%, and 28% on average.

Table 3 Top-1, top-5, top-10 accuracy (%) achieved by each model (by column) on subsets of SPBSL

Databases	Augment	Pose-GRU			Pose-TGCN			I3D		
		top-1	top-5	top-10	top-1	top-5	top-10	top-1	top-5	top-10
Search-100	False	76.18	95.48	98.77	87.89	97.74	99.18	63.86	90.55	96.92
	True	**92.61**	99.59	**100.0**	96.92	**100.0**	**100.0**	**94.05**	98.97	99.38
Net-100	False	74.01	96.05	98.87	90.77	98.31	99.44	63.84	90.02	94.73
	True	89.64	**99.81**	**100.0**	**97.18**	**100.0**	**100.0**	90.96	**99.81**	**100.0**
Search-300	False	60.32	87.16	93.22	82.11	94.59	97.04	61.98	85.57	92.28
	True	**82.61**	98.05	99.42	90.62	99.71	99.93	**90.62**	**98.34**	**99.42**
Net-300	False	63.56	88.47	95.06	83.25	94.72	97.46	62.32	85.45	92.11
	True	81.06	**99.11**	**99.86**	**90.80**	**99.73**	**100.0**	86.69	97.46	98.97
Search-500	False	54.73	84.29	91.19	79.43	91.71	94.57	76.78	92.75	96.14
	True	**77.86**	97.48	99.26	**87.76**	**99.26**	99.83	**87.59**	**98.44**	**99.26**
Net-500	False	58.22	86.22	93.01	78.49	90.00	94.13	73.63	92.08	95.60
	True	76.80	**97.95**	**99.58**	87.37	99.23	**99.92**	84.79	97.72	99.23
Search-1k	False	50.14	78.14	86.60	69.68	86.88	92.21	71.59	91.40	95.05
	True	72.88	95.83	98.52	80.08	97.03	98.99	80.34	96.71	98.24
Net-1k	False	51.30	79.57	87.94	71.79	89.08	93.16	69.97	91.93	94.23
	True	**73.20**	**96.51**	**99.05**	**81.82**	**98.14**	**99.63**	**81.55**	**97.35**	**99.51**

8 Conclusion

We propose a novel framework based on shapelet that can build a large-scale word-level sign language database automatically from online sign language-interpreted videos. Two modified shapelet methods show that they can identify the target sign from the weak and noisy supervision information in a shorter time. The shapelet net method that is initialized using a roughly searched shapelet shows superiority in speed and performance. Then, we construct a 1000 words Britten sign language database, SPBSL, which contains four subsets based on different shapelet methods and different sample strategies. Finally, we evaluated the three state-of-the-art network architectures and approaches as the baselines on our database and demonstrated promising results of our proposed framework.

For future works, in response to the current low recall rate, a judging method should be proposed to remove invalid words and signs. In addition, a more robust and reasonable sign language feature descriptor needs to be designed to hold all kinds of variants, and an appropriate time-scale scaling needs to be introduced to make the distance between motions more accurate.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant No. 61876054.

Code availability The database, models, and code are available at https://github.com/hitmaxiang/SPBSL.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Vos T, Barber RM, Bell B, Bertozzi-Villa A, Biryukov S, Bolliger I, Charlson F, Davis A, Degenhardt L, Dicker D (2015) Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990–2013: a systematic analysis for the global burden of disease study 2013. The Lancet 386(9995):743–800. https://doi.org/10.1016/S0140-6736(15)60692-4
2. Olusanya BO, Neumann KJ, Saunders JE (2014) The global burden of disabling hearing impairment: a call to action. Bull World Health Organ 92:367–373. https://doi.org/10.2471/BLT.13.128728
3. Stokoe J, William C (2005) Sign language structure: an outline of the visual communication systems of the American deaf. J Deaf Studi Deaf Educ 10(1):3–37. https://doi.org/10.1093/deafed/eni001
4. Rabiner LR (1989) Tutorial on hidden Markov models and selected applications in speech recognition. In: Proceedings of the IEEE, vol 77, pp 257–286. https://doi.org/10.1109/5.18626
5. McCallum A, Freitag D, Pereira FCN (2000) Maximum entropy Markov models for information extraction and segmentation. In: Proceedings of the seventeenth international conference on machine learning (ICML 2000), pp 591–598
6. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the eighteenth international conference on machine learning (ICML 2001), pp 282–289

7. Yu S-H, Huang C-L, Hsu S-C, Lin H-W, Wang H-W (2011) Vision-based continuous sign language recognition using product hmm. In: The first Asian conference on pattern recognition, pp 510–514. https://doi.org/10.1109/ACPR.2011.6166631
8. Wu C-H, Lin J-C, Wei W-L (2013) Two-level hierarchical alignment for semi-coupled hmm-based audiovisual emotion recognition with temporal course. IEEE Trans Multimedia 15(8):1880–1895. https://doi.org/10.1109/TMM.2013.2269314
9. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1724–1734. https://doi.org/10.3115/v1/D14-1179
10. Li D, Opazo CR, Yu X, Li H (2020) Word-level deep sign language recognition from video: a new large-scale dataset and methods comparison. In: 2020 IEEE winter conference on applications of computer vision (WACV), pp 1448–1458. https://doi.org/10.1109/WACV45572.2020.9093512
11. Carreira J, Zisserman A (2017) Quo Vadis, action recognition? A new model and the kinetics dataset. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 4724–4733. https://doi.org/10.1109/CVPR.2017.502
12. Kadous MW (2002) Temporal classification: extending the classification paradigm to multivariate time series. PhD thesis, School of Computer Science and Engineering, University of New South Wales
13. Fels SS, Hinton GE (1993) Glove-talk: a neural network interface between a data-glove and a speech synthesizer. IEEE Trans Neural Netw 4(1):2–8. https://doi.org/10.1109/72.182690
14. Gao W, Ma J, Shan S, Chen X, Zeng W, Zhang H, Yan J, Wu J (2000) Handtalker: a multimodal dialog system using sign language and 3-d virtual human. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 1948. Beijing, China, pp 564–571. https://doi.org/10.1007/3-540-40063-x_74
15. Chai X, Wang H, Chen X (2014) The Devisign large vocabulary of Chinese sign language database and baseline evaluations. Technical report VIPL-TR-14-SLR-001. Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS
16. Ginsberg J, Mohebbi MH, Patel RS, Brammer L, Smolinski MS, Brilliant L (2009) Detecting influenza epidemics using search engine query data. Nature 457(7232):1012–1014. https://doi.org/10.1038/nature07634
17. Xu E, Nemati S, Tremoulet AH (2022) A deep convolutional neural network for Kawasaki disease diagnosis. Sci Rep 12(1):1–6. https://doi.org/10.1038/s41598-022-15495-x
18. Morales J, Yoshimura N, Xia Q, Wada A, Namioka Y, Maekawa T (2022) Acceleration-based human activity recognition of packaging tasks using motif-guided attention networks. In: 2022 IEEE international conference on pervasive computing and communications (PerCom), pp 1–12. https://doi.org/10.1109/PerCom53586.2022.9762388
19. Kumar P, Roy PP, Dogra DP (2018) Independent Bayesian classifier combination based sign language recognition using facial expression. Inf Sci 428:30–48. https://doi.org/10.1016/j.ins.2017.10.046
20. Saeed S, Mahmood MK, Khan YD (2018) An exposition of facial expression recognition techniques. Neural Comput Appl 29(9):425–443. https://doi.org/10.1007/s00521-016-2522-2
21. Shao Z, Li YF (2013) A new descriptor for multiple 3d motion trajectories recognition. In: 2013 IEEE international conference on robotics and automation, pp 4749–4754. https://doi.org/10.1109/ICRA.2013.6631253
22. Shao Z, Li Y (2015) Integral invariants for space motion trajectory matching and recognition. Pattern Recogn 48(8):2418–2432. https://doi.org/10.1016/j.patcog.2015.02.029
23. Wang H, Chai X, Chen X (2016) Sparse observation (so) alignment for sign language recognition. Neurocomputing 175:674–685. https://doi.org/10.1016/j.neucom.2015.10.112
24. Kumar EK, Kishore PVV, Kiran Kumar MT, Kumar DA (2020) 3d sign language recognition with joint distance and angular coded color topographical descriptor on a 2 stream CNN. Neurocomputing 372:40–54. https://doi.org/10.1016/j.neucom.2019.09.059
25. Ma X, Yuan L, Wen R, Wang Q (2020) Sign language recognition based on concept learning. In: 2020 IEEE international instrumentation and measurement technology conference (I2MTC), pp 1–6. https://doi.org/10.1109/I2MTC43012.2020.9128734
26. Wadhawan A, Kumar P (2020) Deep learning-based sign language recognition system for static signs. Neural Comput Appl 32(12):7957–7968. https://doi.org/10.1007/s00521-019-04691-y
27. Güney S, Erkuş M (2021) A real-time approach to recognition of Turkish sign language by using convolutional neural networks. Neural Comput Appl. https://doi.org/10.1007/s00521-021-06664-6
28. Huang J, Zhou W, Zhang Q, Li H, Li W (2018) Video-based sign language recognition without temporal segmentation. In: 32nd AAAI conference on artificial intelligence, AAAI 2018, vol 32, pp 2257–2264
29. Kumar P, Gauba H, Pratim Roy P, Prosad Dogra D (2017) A multimodal framework for sensor based sign language recognition. Neurocomputing 259:21–38. https://doi.org/10.1016/j.neucom.2016.08.132
30. Gao L, Li H, Liu Z, Liu Z, Wan L, Feng W (2021) RNN-transducer based Chinese sign language recognition. Neurocomputing 434:45–54. https://doi.org/10.1016/j.neucom.2020.12.006
31. Cihan Camgöz N, Koller O, Hadfield S, Bowden R (2020) Sign language transformers: joint end-to-end sign language recognition and translation. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10020–10030. https://doi.org/10.1109/CVPR42600.2020.01004
32. Liu Y, Zhang H, Xu D, He K (2022) Graph transformer network with temporal kernel attention for skeleton-based action recognition. Knowl Based Syst 240:108146. https://doi.org/10.1016/j.knosys.2022.108146
33. Sun M, Savarese S (2011) Articulated part-based model for joint object detection and pose estimation. In: Proceedings of the IEEE international conference on computer vision, Barcelona, Spain, pp 723–730. https://doi.org/10.1109/ICCV.2011.6126309
34. Tompson J, Goroshin R, Jain A, LeCun Y, Bregler C (2015) Efficient object localization using convolutional networks. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol 07-12-June-2015, Boston, MA, USA, pp 648–656. https://doi.org/10.1109/CVPR.2015.7298664
35. Wei S-E, Ramakrishna V, Kanade T, Sheikh Y (2016) Convolutional pose machines. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol 2016-December, Las Vegas, NV, USA, pp 4724–4732. https://doi.org/10.1109/CVPR.2016.511
36. Simon T, Joo H, Matthews I, Sheikh Y (2017) Hand keypoint detection in single images using multiview bootstrapping. In: Proceedings-30th IEEE conference on computer vision and pattern recognition, CVPR 2017, vol 2017-January, Honolulu, HI, USA, pp 4645–4653. https://doi.org/10.1109/CVPR.2017.494

37. JOZE HV, Koller O (2016) Ms-asl: a large-scale data set and benchmark for understanding American sign language. In: Proceedings of the British machine vision conference, pp 41–14116. https://doi.org/10.5244/C.33.41
38. Albanie S, Varol G, Momeni L, Afouras T, Chung JS, Fox N, Zisserman A (2020) Bsl-1k: scaling up co-articulated sign language recognition using mouthing cues. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 12356 LNCS, Glasgow, UK, pp 35–53. https://doi.org/10.1007/978-3-030-58621-8_3
39. Momeni L, Varol G, Albanie S, Afouras T, Zisserman A (2021) Watch, read and lookup: learning to spot signs from multiple supervisors. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 12627 LNCS, pp 291–308. https://doi.org/10.1007/978-3-030-69544-6_18
40. Barbara L, Loeding AP, Sudeep Sarkar, Karshmer AI (2004) Progress in automated computer recognition of sign language. In: Computers helping people with special needs, 9th international conference, ICCHP 2004, Paris, France, July 7–9, 2004, Proceedings. Lecture notes in computer science, vol 3118, pp 1079–1087. https://doi.org/10.1007/978-3-540-27817-7_159
41. Martinez AM, Wilbur RB, Shay R, Kak AC (2002) Purdue RVL-SLLL ASL database for automatic recognition of American sign language. In: Proceedings 4th IEEE international conference on multimodal interfaces, ICMI 2002, pp 167–172. https://doi.org/10.1109/ICMI.2002.1166987
42. Zahedi M, Keysers D, Deselaers T, Ney H (2005) Combination of tangent distance and an image distortion model for appearance-based sign language. In: Lecture notes in computer science, vol 3663, Vienna, Austria, pp 401–408. https://doi.org/10.1007/11550518_50
43. Liu B, Xiao Y, Hao Z (2018) A selective multiple instance transfer learning method for text categorization problems. Knowl Based Syst 141:178–187. https://doi.org/10.1016/j.knosys.2017.11.019
44. Zhang Y, Zhang H, Tian Y (2020) Sparse multiple instance learning with non-convex penalty. Neurocomputing 391:142–156. https://doi.org/10.1016/j.neucom.2020.01.100
45. Buehler P, Everingham M, Zisserman A (2009) Learning sign language by watching tv (using weakly aligned subtitles). In: 2009 IEEE computer society conference on computer vision and pattern recognition workshops, CVPR workshops 2009, pp 2961–2968. https://doi.org/10.1109/CVPRW.2009.5206523
46. Pfister T, Charles J, Zisserman A (2013) Large-scale learning of sign language by watching tv (using co-occurrences). In: Proceedings of the British machine vision conference, pp 20–12011. https://doi.org/10.5244/C.27.20
47. Andrews S, Tsochantaridis I, Hofmann T (2002) Support vector machines for multiple-instance learning. In: Advances in neural information processing systems, vol 15, pp 561–568
48. Cooper H, Bowden R (2009) Learning signs from subtitles: a weakly supervised approach to sign language recognition. In: 2009 IEEE conference on computer vision and pattern recognition, pp 2568–2574. https://doi.org/10.1109/CVPR.2009.5206647
49. Varol G, Momeni L, Albanie S, Afouras T, Zisserman A (2021) Read and attend: temporal localisation in sign language videos. In: 2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 16852–16861. https://doi.org/10.1109/CVPR46437.2021.01658
50. Miech A, Alayrac J-B, Smaira L, Laptev I, Sivic J, Zisserman A (2020) End-to-end learning of visual representations from uncurated instructional videos. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp 9876–9886. https://doi.org/10.1109/CVPR42600.2020.00990
51. Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, Paris, France, pp 947–955. https://doi.org/10.1145/1557019.1557122
52. Mueen A, Keogh E, Young N (2011) Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 1154–1162. https://doi.org/10.1145/2020408.2020587
53. Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: SIAM international conference on data mining 2013, SMD 2013, Austin, TX, USA, pp 668–676
54. Chang K-W, Deka B, Hwu W-MW, Roth D (2012) Efficient pattern-based time series classification on GPU. In: Proceedings-IEEE international conference on data mining, ICDM, Brussels, Belgium, pp 131–140. https://doi.org/10.1109/ICDM.2012.132
55. Ji C, Zhao C, Liu S, Yang C, Pan L, Wu L, Meng X (2019) A fast shapelet selection algorithm for time series classification. Comput Netw 148:231–240. https://doi.org/10.1016/j.comnet.2018.11.031
56. Hu Y, Zhan P, Xu Y, Zhao J, Li Y, Li X (2021) Temporal representation learning for time series classification. Neural Comput Appl 33(8):3169–3182. https://doi.org/10.1007/s00521-020-05179-w
57. Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2014) Learning time-series shapelets. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, New York, NY, USA, pp 392–401. https://doi.org/10.1145/2623330.2623613
58. Zhang Z, Zhang H, Wen Y, Zhang Y, Yuan X (2018) Discriminative extraction of features from time series. Neurocomputing 275:2317–2328. https://doi.org/10.1016/j.neucom.2017.11.002
59. Shah M, Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2016) Learning DTW-Shapelets for time-series classification. In: Proceedings of the 3rd IKDD conference on data science, 2016, pp 1–8. https://doi.org/10.1145/2888451.2888456
60. Ma Q, Zhuang W, Li S, Huang D, Cottrell G (2020) Adversarial dynamic shapelet networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 5069–5076
61. Pfister T, Charles J, Zisserman A (2014) Domain-adaptive discriminative one-shot learning of gestures. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 8694 LNCS, Zurich, Switzerland, pp 814–829. https://doi.org/10.1007/978-3-319-10599-4_52
62. Yeh C-CM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Silva DF, Mueen A, Keogh E (2016) Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: Proceedings-IEEE international conference on data mining, ICDM, vol 0, Barcelona, Catalonia, Spain, pp 1317–1322. https://doi.org/10.1109/ICDM.2016.89
63. Zhu Y, Zimmerman Z, Senobari NS, Yeh C-CM, Funning G, Mueen A, Brisk P, Keogh E (2016) Matrix profile ii: exploiting a novel algorithm and GPUS to break the one hundred million barrier for time series motifs and joins. In: Proceedings-IEEE international conference on data mining, ICDM, vol 0, Barcelona, Catalonia, Spain, pp 739–748. https://doi.org/10.1109/ICDM.2016.126
64. Parliament S (2021) The playlist of BSL videos. https://youtube.com/playlist?list=PL4l0q4AbG0mmB3AEL6F-DCjK7uhRp0ll_. Accessed 21 July
65. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015)

Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y