Searching for fingerspelled content in American Sign Language

Anonymous ACL submission

Abstract

Natural language processing for sign language video—including tasks like recognition, translation, and search—is crucial for making artificial intelligence technologies accessible to deaf individuals, and is gaining research interest in recent years. In this paper, we address the problem of searching for fingerspelled keywords or key phrases in raw sign language videos. This is an important task since significant content in sign language is often conveyed via fingerspelling, and to our knowledge the task has not been studied before. We propose an end-to-end model for this task, FSS-Net, that jointly detects fingerspelling and matches it to a text sequence. Our experiments, done on a large public dataset of ASL fingerspelling in the wild, show the importance of fingerspelling detection as a component of a search and retrieval model. Our model significantly outperforms baseline methods adapted from prior work on related tasks.

1 Introduction

Sign languages are a type of natural language which convey meaning through sequences of handshapes and gestures as well as non-manual elements, and are a chief means of communication for about 70 million deaf people worldwide[1] Automatic sign language technologies would help to bridge the communication barrier between deaf and hearing individuals, and would make deaf video media more searchable and indexable.

Automatic sign language processing has recently received growing interest in the computer vision (CV) and natural language processing (NLP) communities. Yin et al. (2021) make several recommendations for the study of sign languages in NLP research, including greater emphasis on real-world data. Most studies on sign language are based on data collected in a controlled environment, either in a studio setting (Martínez et al., 2002; Kim et al., 2017) or in a specific domain (Forster et al., 2016). The challenges involved in real-world signing videos, including various visual conditions and different levels of fluency in signing, are not fully reflected in such datasets. Automatic processing of sign language videos "in the wild" has not been addressed until recently, and is still restricted to tasks like isolated sign recognition (Albanie et al., 2020; Joze and Koller, 2019; Li et al., 2020) and fingerspelling recognition (Shi et al., 2018, 2019). In this work we take a step further and study search and retrieval of arbitrary fingerspelled content in real-world American Sign Language (ASL) video (see Figure 1).

Fingerspelling is a component of sign language in which words are signed by a series of handshapes or movements corresponding to single letters (see the Appendix for the ASL fingerspelling alphabet). Fingerspelling is used mainly for lexical items that do not have their own signs, such as proper nouns or technical terms, and has an important place in sign language. For example, fingerspelling accounts for 12-35% of ASL (Padden and Gunsauls, 2003) Since important content like named entities is often fingerspelled, the fingerspelled portions of a sign language video often carry a disproportionate amount of the content.

Most prior work on fingerspelling has focused on recognition (Shi et al., 2018, 2019), that is, transcription of a fingerspelling video clip into text. However, automatic recognition assumes that the boundaries of fingerspelled segments are known at test time, and may not be the end goal in real-world use cases. In addition, complete transcription may not be necessary to extract the needed information. Fingerspelling search, such as retrieving sign language videos based on a query word, is often a more useful task, and is an important component of general video search involving sign language.

In addition to introducing the task, we address

[1] From `https://wfdeaf.org/our-work/`

the research question of whether the explicit temporal localization of fingerspelling can help its search and retrieval, and how best to localize it. As fingerspelling occurs sparsely in the signing stream, explicit detection of fingerspelling could potentially improve search performance by removing unrelated signs. To this end, we propose an end-to-end model, FSS-Net, which jointly detects fingerspelling from unconstrained signing video and matches it to text queries. Our approach consistently outperforms a series of baselines without explicit detection and a baseline with an off-the-shelf fingerspelling detector by a large margin.

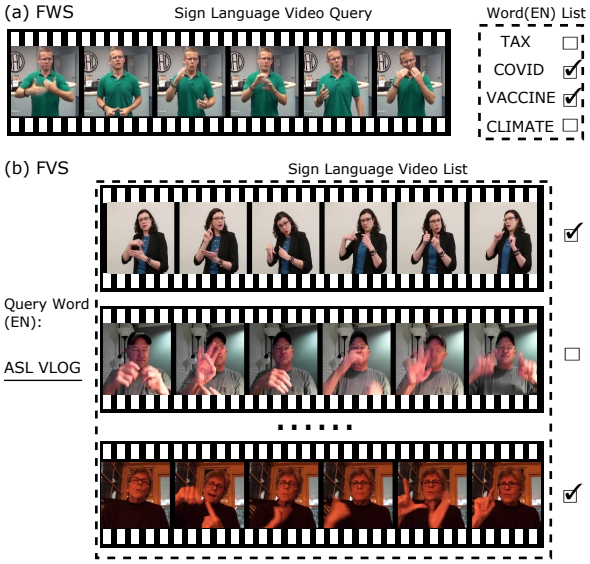


Figure 1: Our two search tasks: (a) *fingerspelled word search* (FWS) for determining which words are fingerspelled in a sign language video clip, and (b) *fingerspelling video search* (FVS) for searching for sign language videos that include a fingerspelled query word/phrase. The sign language videos are untrimmed, i.e. they include regular signs in addition to fingerspelling, and are downsampled for visualization.

2 Related Work

In existing work on sign language video processing, search and retrieval tasks have been studied much less than sign language recognition (mapping from sign language video to gloss labels) (Koller et al., 2017; Forster et al., 2016), translation (mapping from sign language video to text in another language) (Yin and Read, 2020; Camgöz et al., 2018). Work thus far on sign language search has been framed mainly as the retrieval of lexical signs rather than fingerspelling. Pfister et al. (2013); Albanie et al. (2020) employ mouthing to detect keywords in sign-interpreted TV programs with coarsely aligned subtitles. Tamer and Saraçlar (2020a,b) utilize whole-body pose estimation to search for sign language keywords (gloss or translated word) in a German Sign Language translation dataset PHOENIX-2014T (Camgöz et al., 2018). All prior work on keyword search for sign language has been done in a closed-vocabulary setting, which assumes that only words from a pre-determined set will be queried. Searching in an open-vocabulary setting, including proper nouns, typically requires searching for fingerspelling.

Some related tasks in the speech processing literature are spoken term detection (STD) and query-by-example search, which are the tasks of automatically retrieving speech segments from a database that match a given text or audio query (Knill et al., 2013; Mamou et al., 2007; Chen et al., 2015). In terms of methodology, our model also shares some aspects with prior work on moment retrieval (Gao et al., 2017; Xu et al., 2019; Zhang et al., 2020), which also combines candidate generation and matching components. However, we incorporate additional task-specific elements that consistently improve performance.

3 Tasks

We consider two tasks: Fingerspelled Word Search (FWS) and Fingerspelling-based Video Search (FVS). FWS and FVS respectively consist of detecting fingerspelled words within a given raw ASL video stream and detecting video clips of interest containing a given fingerspelled word.[2] Given a query video clip v and a list of n words $w_{1:n}$, FWS is the task of finding which (if any) of $w_{1:n}$ are present in v. Conversely, in FVS the input is a query word w and n video clips $v_{1:n}$, and the task consists of finding all videos containing the fingerspelled word w. We consider an open-vocabulary setting where the word w is not constrained to a pre-determined set. The two tasks correspond to two directions of search (video⟶text and text⟶video), as is standard practice in other retrieval work such as video-text search (Zhang et al., 2018; Ranjay et al., 2017; Ging et al., 2020).

4 Model

We propose a single model, FSS-Net (for "Finger-Spelling Search Network"), summarized in Figure 2, to address the two aforementioned search tasks. FSS-Net receives a pair of inputs—a raw

[2]We use "word" to refer to a fingerspelling sequence, which could be a single word or a phrase.

ASL video clip, and a written text sequence—and produces a score indicating the degree of match between the video clip and the text. The text is encoded into an embedding vector via a learned encoder. The visual branch of FSS-Net generates a number of fingerspelling segment proposals and each proposed visual segment is encoded into a feature space shared with the text embeddings. Paired embeddings from both modalities are drawn towards each other in the shared embedding space during training.

Image encoding The input image frames are encoded into a sequence of feature vectors via an image encoder, which consists of the VGG-19 (Simonyan and Zisserman, 2015a) convolutional layers followed by a Bi-LSTM.[3] We use raw RGB images as input, instead of signer pose as used in some prior work (Tamer and Saraçlar, 2020b,a) on sign language search, as estimating pose for hands is particularly hard for signing videos in the wild (see Section 6 for details).

Temporal proposal generation Suppose the visual feature sequence is $\mathbf{f}_{1:T}$, where T is the number of frames in the video clip. The purpose of temporal proposal generation is to produce a number of candidate fingerspelling segments $\mathcal{H}(\mathbf{I}_{1:T}) = \{(s_i, t_i)\}_{1 \le i \le |\mathcal{H}(\mathbf{I}_{1:T})|}$ from $\mathbf{f}_{1:T}$, where s_i,t_i are the start and end frame indices of the i^{th} proposed segment. Below we use $\mathcal{H}$ as a shorthand of $\mathcal{H}(\mathbf{I}_{1:T})$. Here we adopt the strategy in (Xu et al., 2017), which is commonly used to generate proposals for action detection. Briefly, the model assigns a probability p_{det} of each proposal being fingerspelling. See (Xu et al., 2017) for more details. We denote the detection loss as L_{det}.

Note that the training requires known ground-truth fingerspelling boundaries. In the fingerspelling datasets we use here (Shi et al., 2018, 2019), the fingerspelling boundaries are already annotated, so no further annotation is needed.

Filtering A visual embedding is produced for each segment. We denote a labeled fingerspelling segment (shortened as fingerspelling segment below) as a tuple (s, t, w), where s, t and w represent the start frame index, the end frame index, and the written text it represents. A naive approach would be to use only the ground-truth fingerspelling segments $\mathcal{P}_g = \{(s_i, t_i, w_i)\}_{1 \le i \le |\mathcal{P}_g|}$ for training. However, this approach does not take into account the potential shifts (errors) that may exist at test time between the ground-truth and generated segment proposals. The embeddings produced by the fingerspelling encoder should be robust to such shifts. To this end, we incorporate proposals in forming positive pairs at training time. Formally, let the set of time intervals from the temporal proposal generator be $\mathcal{H} = \{(s_i, t_i)\}_{1 \le i \le |\mathcal{H}|}$. We sample K intervals from $\mathcal{P}_t$ to form the set of generated fingerspelling segments:

$$\begin{aligned}\mathcal{P}_k = \{&(s_k, t_k, w_g) | IoU((s_k, t_k), (s_g, t_g)) > \delta_{IoU}, \\ &IS((s_t, t_k), (s_g, t_g)) > \delta_{IS}, \\ &(s_k, t_k) \in \mathcal{H}, (s_g, t_g, w_g) \in \mathcal{P}_g\}\end{aligned} \tag{1}$$

where $\text{IS}(x, y) = \frac{\text{Intersection}(x,y)}{\text{Length}(y)}$ and $\text{IoU}(x, y) = \frac{\text{Intersection}(x,y)}{\text{Union}(x,y)}$. We use δ^k_{IoU} and δ^k_{IS} to control the degree to which the proposals can deviate from the ground-truth. In addition to the intersection over union (IoU), we use the normalized intersection IS to eliminate proposals with many missing frames. We take the union of the two sets, $\mathcal{P}_+ = \mathcal{P}_g \cup \mathcal{P}_k$, as the filtered proposal set to be encoded.

Fingerspelling visual encoding (FS-encoding) The visual encoding of each segment $(s, t, w) \in \mathcal{P}_+$ is $\mathbf{e}_v^{(s,t)} = \text{BiLSTM}(\mathbf{f}_{s:t})$.[4]

Text encoding A written word (or phrase) $\mathbf{w}$ is mapped to an embedding vector $\mathbf{e}_x^w$ via a text encoder. To handle words not seen at training time (and better handle rarely seen words), we first decompose $\mathbf{w}$ into a sequence of characters $c_{1:|w|}$ (e.g. 'ASL'='A'-'S'-'L') and feed the character sequence $c_{1:|w|}$ into a text encoder (here, a Bi-LSTM[5]).

Visual-text matching With the above pairs of visual and textual embeddings, we use a training objective function consisting of two triplet loss terms:

$$\begin{aligned}&L_{tri}(\mathbf{I}_{1:T}, \mathcal{P}_+) = \\ &\sum_{(s,t,w) \in \mathcal{P}_+} \max\{m + d(\mathbf{e}_v^{(s,t)}, \mathbf{e}_x^w) \\ &- \frac{1}{|\mathcal{N}_w|} \sum_{w' \in \mathcal{N}_w} d(\mathbf{e}_v^{(s,t)}, \mathbf{e}_x^{w'}), 0\} \\ &+ \max\{m + d(\mathbf{e}_v^{(s,t)}, \mathbf{e}_x^w) \\ &- \frac{1}{|\mathcal{N}_v|} \sum_{(s',t') \in \mathcal{N}_v} d(\mathbf{e}_v^{(s',t')}, \mathbf{e}_x^w), 0\}\end{aligned} \tag{2}$$

[3] Transformers (Vaswani et al., 2017) can also be used, but in our initial experiments, they were outperformed by Bi-LSTMs on our tasks and data.

[4] We compared the Bi-LSTM encoder with average/max pooling of $f_{s:t}$, and found the former to perform better.

[5] Again, transformers can also be used, but in our experiments Bi-LSTM show better performance.

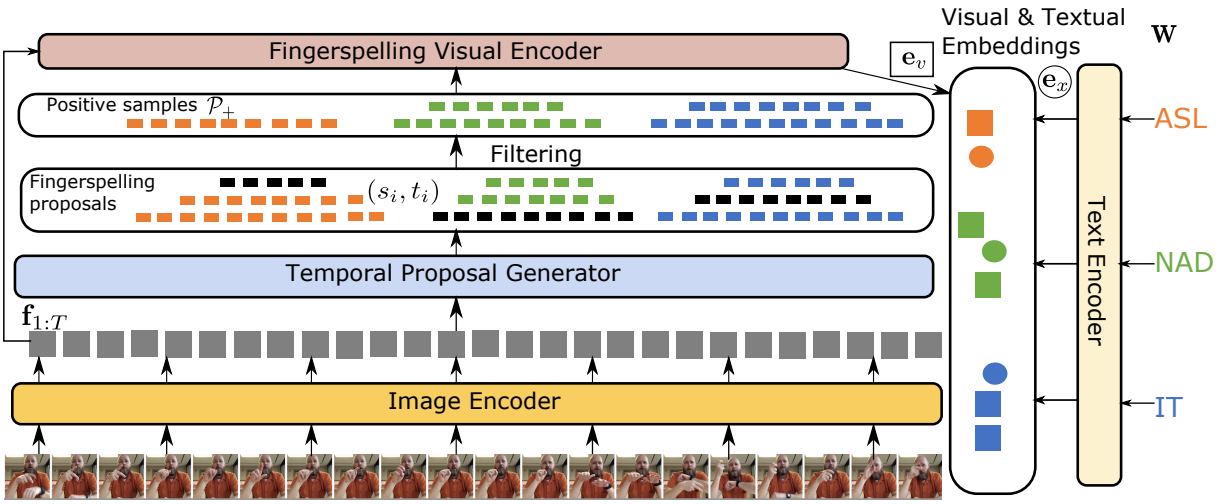


Figure 2: FSS-Net: The proposed model for fingerspelling search and retrieval. The model maps candidate fingerspelling segments and text into a shared embedding space. ○: text embedding, □: visual embedding. The colors correspond to different input fingerspelling sequences. As pictured, this is the training time model, where the pairing between text and video segments is known. At test time, the labels (colors) of the visual embeddings are unknown and we do not filter the proposals.

where d denotes cosine distance $d(\mathbf{a}, \mathbf{b}) = 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$, m is a margin, and $\mathcal{N}_v$ and $\mathcal{N}_w$ are sets of negative samples of proposals and words. To form negative pairs we use semi-hard negative sampling (Schroff et al., 2015):

$$\mathcal{N}_v = \{(s', t') | d(\mathbf{e}_v^{(s',t')}, \mathbf{e}_x^w) > d(\mathbf{e}_v^{(s,t)}, \mathbf{e}_x^w)\}$$
$$\mathcal{N}_w = \{w' | d(\mathbf{e}_v^{(s,t)}, \mathbf{e}_x^{w'}) > d(\mathbf{e}_v^{(s,t)}, \mathbf{e}_x^w)\} \quad (3)$$

For efficiency, negative samples are selected from the corresponding mini-batch.

Overall loss The model is trained with a combination of the detection loss and triplet loss:

$$L_{tot}(\mathbf{I}_{1:T}, \mathcal{P}_g) = \lambda_{det} L_{det}(\mathbf{I}_{1:T}, \mathcal{P}_g) + L_{tri}(\mathbf{I}_{1:T}, \mathcal{P}_+) \quad (4)$$

with the tuned weight λ_{det} controlling the relative importance of detection versus visual-textual matching.

Inference At test time, the model assigns a score $sc(\mathbf{I}_{1:T}, w)$ to a given video clip $\mathbf{I}_{1:T}$ and word w. The word is encoded into the word embedding $\mathbf{e}_x^w$. Suppose the set of fingerspelling proposals generated by the temporal proposal generator is $\mathcal{H}(\mathbf{I}_{1:T})$. We define a scoring function for the proposal $h \in \mathcal{H}(\mathbf{I}_{1:T})$ and word w

$$sc_{word}(h_m, w) = p_{det}(1 - d(\mathbf{e}_v^{h_m}, \mathbf{e}_x^w))^\beta \quad (5)$$

where p_{det} is the probability given by the temporal proposal generator and β controls the relative weight between detection and matching. In other words, in order for a segment and word to receive a high score, the segment should be likely to be fingerspelling (according to p_{det}) and its embedding should match the text. Finally, the score for the video clip $\mathbf{I}_{1:T}$ and the word w is defined as the highest score among the set of proposals $\mathcal{H}(\mathbf{I}_{1:T})$:

$$sc(\mathbf{I}_{1:T}, w) = \max_{h \in \mathcal{H}(\mathbf{I}_{1:T})} sc_{word}(h, w) \quad (6)$$

5 Experimental Setup

5.1 Data

We conduct experiments on ChicagoFSWild (Shi et al., 2018) and ChicagoFSWild+ (Shi et al., 2019), two large-scale publicly available fingerspelling datasets which contain 7,304 and 55,272 fingerspelling sequences respectively. The ASL videos in the two datasets are collected from online resources and include a variety of viewpoints and styles, such as webcam videos and lectures.

We follow the setup of (Shi et al., 2021) and split the raw ASL videos into 300-frame clips with a 75-frame overlap between neighboring chunks and remove clips without fingerspelling. The numbers of clips in the various splits can be found in the Appendix. On average, each clip contains 1.9/1.8 fingerspelling segments in the ChicagoFSWild and ChicagoFSWild+ respectively.

5.2 Baselines

We compare the proposed model, FSS-Net, to the following baselines adapted from common approaches for search and retrieval in related fields.

To facilitate comparison, the network architecture for the visual and text encoding in all baselines is the same as in FSS-Net.

Recognizer In this approach, we train a recognizer that transcribes the video clip into text. Specifically, we train a recognizer to output a sequence of symbols consisting of either fingerspelled letters or a special non-fingerspelling symbol <x>. We train the recognizer is trained with a connectionist temporal classification (CTC) loss (Graves et al., 2006), which is commonly used for speech recognition. need to say what the structure of the recognizer model is At test time, we use beam search to generate a list of hypotheses $\hat{\mathbf{w}}_{1:M}$ for the target video clip $\mathbf{I}_{1:T}$. Each hypothesis $\hat{w}_m$ is split into a list of words $\{\hat{w}_m^n\}_{1\leq n\leq N}$ separated by <x>. The matching score between video $\mathbf{I}_{1:T}$ and $\mathbf{w}$ is defined as:

$$sc(\mathbf{I}_{1:T}, w) = 1 - \min_{1\leq m\leq M} \min_{1\leq n\leq N} \text{LER}(\hat{w}_m^n, w) \tag{7}$$

where the letter error rate LER is the Levenshtein edit distance. This approach is adapted from (Saraçlar and Sproat, 2004) for spoken utterance retrieval. Fingerspelling boundary information is not used in training this baseline model.

Whole-clip The whole-clip baseline encodes the whole video clip $\mathbf{I}_{1:T}$ into a visual embedding $\mathbf{e}_v^I$, which is matched to the textual embedding $\mathbf{e}_x^w$ of the query $\mathbf{w}$. The model is trained with contrastive loss as in equation 2. At test time, the score for video clip $\mathbf{I}_{1:T}$ and word $\mathbf{w}$ is:

$$sc(\mathbf{I}_{1:T}, \mathbf{w}) = 1 - d(\mathbf{e}_v^I, \mathbf{e}_x^w) \tag{8}$$

where d is the cosine distance as in FSS-Net. Fingerspelling boundary information is again not used in this baseline.

External detector (Ext-det) This baseline uses the off-the-shelf fingerspelling detectors (Shi et al., 2021) to generate fingerspelling proposals, instead of our proposal generator, and is otherwise identical to FSS-Net. For each dataset (ChicagoFSWild, ChicagoFSWild+), we use the detector trained on the training subset of that dataset.

Attention-based keyword search (Attn-kws) This model is adapted from (Tamer and Saraçlar, 2020b)'s approach for keyword search in sign language. The model employs an attention mechanism to match a text query with a video clip, where each frame is weighted based on the query embedding. The attention mechanism enables the model to implicitly localize frames relevant to the text. The model of (Tamer and Saraçlar, 2020b) is designed for lexical signs rather than fingerspelling. To adapt the model to our open-vocabulary fingerspelling setting, we use the same text encoder as in FSS-Net to map words into embeddings instead of using a word embedding matrix as in (Tamer and Saraçlar, 2020b). Fingerspelling boundary information is again not used in training this model, which arguably puts it at a disadvantage compared to FSS-Net. More details on the formulation of the model can be found in the Appendix.

5.3 Evaluation

For FWS, we use all words in the test set as the test vocabulary $w_{1:n}$. For FVS, all video clips in the test are used as candidates and the text queries are again the entire test vocabulary. We report the results in terms of standard metrics from the video-text retrieval literature (Momeni et al., 2020; Tamer and Saraçlar, 2020a): mean Average Precision (mAP) and mean F1 score (mF1), where the averages are over words for FVS and over videos for FWS. Additional details on data, preprocessing and model implementation can be found in the appendix.

6 Results and analysis

6.1 Main Results

Table 1: FWS/FVS performance on the ChicagoFSWild and ChicagoFSWild+ test sets.The range of mAP and mF1 is [0, 1].

	FWS (Video ⟹ Text)		FVS (Text ⟹ Video)	
	ChicagoFSWild			
Method	mAP	mF1	mAP	mF1
Whole-clip	.175	.154	.142	.119
Attn-KWS	.204	.181	.246	.229
Recognizer	.318	.315	.331	.305
Ext-detector	.383	.385	.332	.312
FSS-Net	**.434**	**.439**	**.394**	**.370**
	ChicagoFSWild+			
Method	mAP	mF1	mAP	mF1
Whole-clip	.466	.457	.548	.526
Attn-KWS	.545	.530	.573	.547
Recognizer	.465	.462	.398	.405
Ext-detector	.633	.641	.593	.577
FSS-Net	**.674**	**.677**	**.638**	**.631**

Table 1 shows the performance of the above approaches on the two datasets. First, we notice that embedding-based approaches consistently outperform the word-list baseline in the larger data setting (ChicagoFSWild+) but not the smaller data setting

(ChicagoFSWild), which suggests that embedding-based models generally require more training data. The inferior performance of word-list also shows that explicit fingerspelling recognition is not necessary for the search tasks. In addition, explicit fingerspelling detection (Ext-det, FSS-Net) improves performance over implicit fingerspelling detection (attn-KWS) and detection-free search (whole-clip). Explicit fingerspelling detection requires boundary information during training. Of the models that don't use such supervision, Attn-KWS is the best performer given enough data, but is still far behind FSS-Net. Our model outperforms all of the alternatives. The relative performance of different models is consistent across the various metrics and the two search tasks. For completeness, we also measure the performance of different models in terms of ranking-based metrics (e.g., Precision@N, Recall@N), as in prior work on video-text retrieval (Ging et al., 2020; Ranjay et al., 2017) (see full results in the Appendix). The relative performance of different models is consistent on different metrics. The analysis below is done on ChicagoFSWild for simplicity. The conclusions also hold for ChicagoFSWild+ .

6.2 Model analysis

Does better localization lead to better search? In the previous section we have seen that models that explicitly detect and localize fingerspelling outperform ones that do not. Next we look more closely at how well several models—Ext-det, Attn-KWS and FSS-Net—perform on the task of localizing fingerspelling, which is a byproduct of these models' output. We measure performance via AP@IoU, a commonly used evaluation metric for action detection (Idrees et al., 2016; Heilbron et al., 2015) that has also been used for fingerspelling detection (Shi et al., 2021). AP@IoU measures the average precision of a detector under the constraint that the overlap of its predicted segments with the ground truth is above some threshold Intersection-over-Union (IoU) value. For attn-KWS, the model outputs an attention vector, which we convert to segments as in (Shi et al., 2021).

In general, the models with more accurate localization also have higher search and retrieval performance, as seen by comparing Table 2 with Table 1. However, differences in AP@IoU do not directly translate to differences in search performance. For example, the AP@IoU of ext-detector (0.344) is an order of magnitude higher than that of attn-KWS (0.035) while their FVS mAP results are much closer (0.593 vs. 0.573).

Table 2: Fingerspelling localization performance for detection-based models.

	AP@0.1	AP@0.3	AP@0.5
Attn KWS	0.268	0.104	0.035
Ext detector	0.495	0.453	0.344
Ours	**0.568**	**0.519**	**0.414**

Raw images vs. estimated pose as input Prior work on sign language search (Tamer and Saraçlar, 2020a,b) has used estimate pose keypoints as input, rathan than raw images as we do here. For comparison, we extract body and hand keypoints with OpenPose (Cao et al., 2019) and train a model with the pose skeleton as input.

Table 3: Impact of input type (pose vs. raw RGB images) on search performance.

	FWS (Video $\Longrightarrow$ Text)		FVS (Text $\Longrightarrow$ Video)	
Input	mAP	mF1	mAP	mF1
Pose	.142	.147	.127	.121
RGB	**.434**	**.439**	**.394**	**.370**

As is shown in Table 3, the pose-based model has much poorer search performance than the RGB image-based models. We believe this is largely because, while pose estimation works well for large motions and clean visual conditions, in our dataset much of the handshape information is lost in the estimated pose (see the Appendix for some qualitative examples).

6.3 Ablation Study

Table 4: Effect of various components of FSS-Net on FWS and FVS.

	FWS		FVS	
	mAP	mF1	mAP	mF1
Full model	**.434**	**.439**	**.394**	**.370**
(1) w/o generator	.186	.180	.259	.270
(2) $\lambda_{det} = 0, \beta = 0$	.411	.420	.373	.350
(3) $\lambda_{det} = 0.1, \beta = 0$	.418	.432	.360	.348
(4) w/o $\mathcal{P}_k$	.411	.420	.386	.366

Within our model, the proposal generator produces a subset of all possible fingerspelling proposals, intended to represent the most likely fin-

Figure 3: Examples of FWS predictions. For each example video, the ground truth (GT) is shown along with the top 5 predicted fingerspelling sequences. Top red line: ground-truth fingerspelling segment. Bottom blue line: highest-scoring predicted fingerspelling segment. Segment locations are shown here for qualitative analysis, but they are not part of the task evaluation. Note that many fingerspelling sequences (both ground-truth and predictions) are abbreviations, and some are misspelled; we include all fingerspelling sequences that appear in the test set in the query vocabulary.

gerspelling segments. To measure whether this component is important to the performance of the model, we compare our full model with the proposal generator to one where the proposal generator is removed (see Table 4). When the proposal generator is not used, the model is trained only with ground-truth fingerspelling segments ($\mathcal{P}_G$) and considers all possible proposals within a set of sliding windows. Such a "sliding-window" approach is commonly used in previous sign language keyword search (Albanie et al., 2020; Pfister et al., 2013) and spoken keyword spotting (Chen et al., 2015). As can be seen from Table 4 (top), the proposal generator greatly improves search performance. This is not surprising, since the proposal generator greatly reduces the number of non-fingerspelling segments, thus lowering the chance of a mismatch between the text and video, and also refines the segment boundaries through regression, which should improve the quality of the visual segment encoding.

The fingerspelling detection component of our model has two aspects that may affect performance: imposing an additional loss during training, and rescoring during inference. We disentangle these two factors and show their respective benefits for our model in Table 4 (middle). The auxiliary detection task, which includes classification between fingerspelling and non-fingerspelling proposals, helps encode more comprehensive visual information into the visual embedding. In addition, the proposal probability output by the detector contains extra information and merging it into the matching score further improves the search performance.

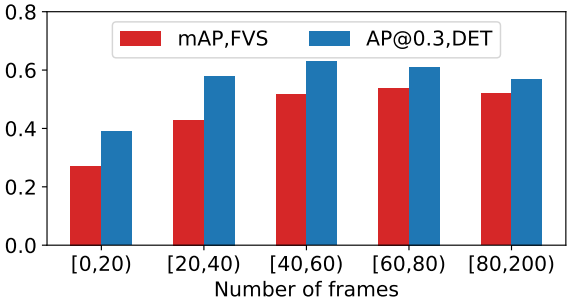
Table 4 (bottom) shows the effect of sampling additional proposals ($\mathcal{P}_g$) in fingerspelling detection. Additional positive samples makes the visual embedding more robust to temporal shifts in the generated proposals, thus improving search performance.

6.4 Result analysis

The performance of our model is worse for short fingerspelled sequences than for long sequences (see Figure 4). may be because shorter words are

harder to spot, as is shown from the trend in fingerspelling detection in the same figure.

Figure 4: Performance as a function of fingerspelled word length. Red: FVS mAP, Blue: detection AP.



The datasets we use are collected from multiple sources, where the video quality varies. To quantify the effect of visual quality on search/retrieval performance, we categorize the ASL videos into three categories according to their source: YouTube, DeafVIDEO, and other miscellaneous sources (misc). YouTube videos are mostly ASL lectures of high resolution. DeafVIDEO videos are vlogs from deaf users of the social media site `deafvideo.tv`, where the style, camera angle, and image quality vary greatly. The visual quality of videos in the miscellaneous category tends to fall between the other two categories. Typical image examples from the three categories can be found in the Appendix. The FWS performance of our model on videos in YouTube, DeafVIDEO, and misc are 0.684, 0.584, 0.629 (mAP) respectively. The results are overall consistent with the perceived relative visual qualities of these categories.

As a qualitative analysis, we examine example words and videos on which our model is more or less successful. Table 5 shows the query words/phrases with the highest/lowest FVS performance, where the best-performing queries tend to be long and drawn from the highest-quality video source.

We also visualize the top FWS predictions made by our model in several video clips (see Appendix). As expected, we see more errors on DeafVIDEO clips. Another common source of error is confusion between letters with similar handshapes (e.g., "i" vs. "j"). A final failure type is fingerspelling detection failure.

7 Conclusion

Our work takes one step toward better addressing the need for language technologies for sign languages, by defining fingerspelling search tasks and developing a model tailored for these tasks. These tasks are complementary to existing work on keyword search for lexical signs, in that it addresses the need to search for a variety of important content that tends to be fingerspelled, like named entities. Fingerspelling search is also more challenging in that it requires the ability to handle an open vocabulary and arbitrary-length queries. Our results demonstrate that a model tailored for the task in fact improves over baseline models based on related work on signed keyword search, fingerspelling detection, and speech recognition. However, there is room for improvement between our results to the maximum possible performance. We make our data sets and code for training and evaluation publicly available, to encourage additional research in this area.[6] One important aspect of our approach is the use of explicit fingerspelling detection within the model. An interesting avenue for future work is to address the case where the training data does not include segment boundaries for detector training. Finally, a complete sign language search system should consider both fingerspelling and lexical sign search.

Table 5: Example words with low/high mAP in FVS. Insde () is the source of the corresponding video

Low	High
script (youtube)	cabol erting (youtube)
agent (misc)	vp ron stern (youtube)
kc (youtube)	co chairs (youtube)
pati (deafvideo)	dr kristin mulrooney (youtube)
mexer (deafvideo)	myles (youtube)
flow (youtube)	camaspace (youtube)
yr (deafvideo)	electronics (youtube)
exalted (misc)	brain (youtube)
poem (youtube)	land (deafvideo)

References

S. Albanie, G. Varol, L. Momeni, T. Afouras, J. Chung, N. Fox, and A. Zisserman. 2020. Bsl-1k: Scaling up co-articulated sign language recognition using mouthing cues. In *ECCV*.

N.C. Camgöz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. 2018. Neural sign language translation. In *CVPR*.

Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *TPAMI*.

[6] Will be available upon acceptance.

G. Chen, C. Parada, and T. N. Sainath. 2015. Query-by-example keyword spotting using long short-term memory networks. In *ICASSP*.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*.

G. Farnebäck. 2003. Two-frame motion estimation based on polynomial expansion. In *SCIA*.

J. Forster, C. Schmidt, O. Koller, M. Bellgardt, and H. Ney. 2016. Extensions of the sign language recognition and translation corpus RWTH-PHOENIX-weather. In *LREC*.

J. Gao, C. Sun, Z. Yang, and R. Nevatia. 2017. Tall: Temporal activity localization via language query. In *CVPR*.

S. Ging, M. Zolfaghari, H. Pirsiavash, and T. Brox. 2020. Coot: Cooperative hierarchical transformer for video-text representation learning. In *NeurIPS*.

A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*.

F.C. Heilbron, V. Escorcia, B. Ghanem, and J.C. Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*.

H. Idrees, A. Zamir, Y.G Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. 2016. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155.

H. Joze and O. Koller. 2019. Ms-asl: A large-scale data set and benchmark for understanding american sign language. In *BMVC*.

J. Keane. 2014. *Towards an articulatory model of handshape: What fingerspelling tells us about the phonetics and phonology of handshape in American Sign Language*. Ph.D. thesis, University of Chicago.

T. Kim, J. Keane, W. Wang, H. Tang, J. Riggle, G. Shakhnarovich, D. Brentari, and K. Livescu. 2017. Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation. *Computer Speech and Language*, pages 209–232.

D. P Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

K. M Knill, M. JF Gales, S. P Rath, P. C Woodland, C. Zhang, and S-X Zhang. 2013. Investigation of multilingual deep neural networks for spoken term detection. In *ASRU*.

O. Koller, S. Zargaran, and H. Ney. 2017. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent cnn-hmms. In *CVPR*.

D. Li, C. Rodriguez-Opazo, X. Yu, and H. Li. 2020. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *WACV*.

J. Mamou, B. Ramabhadran, and O. Siohan. 2007. Vocabulary independent spoken term detection. In *SIGIR*.

A. Martínez, R. Wilbur, Robin Shay, and A. Kak. 2002. Purdue rvl-slll asl database for automatic recognition of american sign language. pages 167–172.

L. Momeni, T. Afouras, T. Stafylakis, S. Albanie, and A. Zisserman. 2020. Seeing wake words: Audio-visual keyword spotting. In *BMVC*.

C. Padden and D.C. Gunsauls. 2003. How the alphabet came to be used in a sign language. *Sign Language Studies*, 4(1):10–33.

T. Pfister, J. Charles, and A. Zisserman. 2013. Large-scale learning of sign language by watching tv (using co-occurrences). In *BMVC*.

K. Ranjay, H. Kenji, F. Ren, F-F.Li, and J.C. Niebles. 2017. Dense-captioning events in videos. *ICCV*.

M. Saraçlar and R. Sproat. 2004. Lattice-based search for spoken utterance retrieval. In *NAACL*.

F. Schroff, D. Kalenichenko, and J. Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.

B. Shi, D. Brentari, G. Shakhnarovich, and K. Livescu. 2021. Fingerspelling detection in american sign language. In *CVPR*.

B. Shi, A. Martinez Del Rio, J. Keane, D. Brentari, G. Shakhnarovich, and K. Livescu. 2019. Fingerspelling recognition in the wild with iterative visual attention. In *ICCV*.

B. Shi, A. Martinez Del Rio, J. Keane, J. Michaux, D. Brentari, G. Shakhnarovich, and K. Livescu. 2018. American Sign Language fingerspelling recognition in the wild. In *SLT*.

K. Simonyan and A. Zisserman. 2015a. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

K. Simonyan and A. Zisserman. 2015b. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

N. Tamer and M. Saraçlar. 2020a. Cross-lingual keyword search for sign language. In *LREC 2020*.

N. Tamer and M. Saraçlar. 2020b. Keyword search for sign language. In *ICASSP*.

A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

H. Xu, A. Das, and K. Saenko. 2017. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*.

H. Xu, K. He, B. A. Plummer, L. Sigal, S. Sclaroff, and K. Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*.

K. Yin and J. Read. 2020. Better sign language translation with STMC-transformer. In *COLING*.

Kayo Yin, Amit Moryossef, Julie Hochgesang, Yoav Goldberg, and Malihe Alikhani. 2021. Including signed languages in natural language processing. In *ACL*.

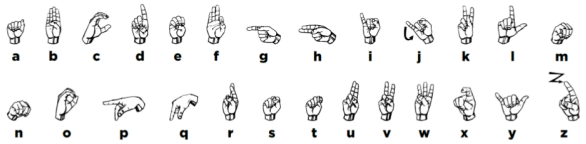
B. Zhang, H. Hu, and F. Sha. 2018. Cross-modal and hierarchical modeling of video and text. In *ECCV*.

H. Zhang, A. Sun, W. Jing, and T. Zhou. 2020. Span-based localizing network for natural language video localization. In *ACL*.

A Appendix

A.1 Fingerspelling alphabet

Figure 5: The ASL fingerspelling alphabet, from (Keane, 2014)



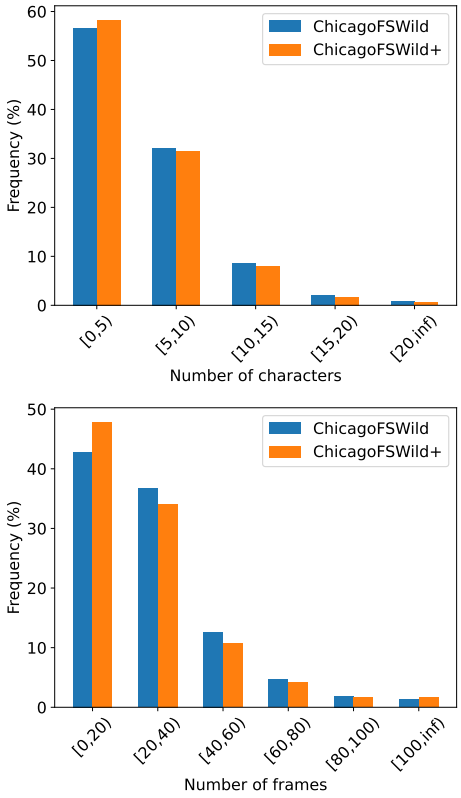
A.2 Data

Figure 6 shows the distribution of fingerspelling length in the two datasets. Figure 7 shows image examples from the following three categories: Youtube, deafvideo, misc.

Table 6: Numbers of 300-frame video clips in ChicagoFSWild and ChicagoFSWild+.

Dataset	Train	Dev	Test
ChicagoFSWild	3,539	691	673
ChicagoFSWild+	13,011	867	885

Figure 6: Distribution of fingerspelling length in ChicagoFSWild and ChicagoFSWild+



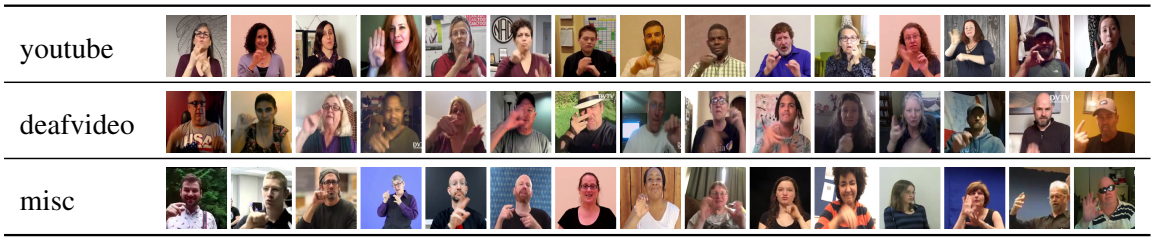
A.3 Implementation Details

Pre-processing The raw images in ChicagoFSWild and ChicagoFSWild+ datasets contain diverse visual scenes which can involve multiple persons. We adapt the heuristic approach used in (Shi et al., 2019) to select the target signer. Specifically, we use an off-the-shelf face detector to detect all the faces in the image. We extend each face bounding boxes by 1.5 times size of the bounding box in 4 directions and select the largest one with highest average magnitude of optical flow (Farnebäck, 2003). We further use the bounding box averaged in the whole sequence to crop the ROI area, which roughly denotes the signing region of a signer. Each image is resize to 160×160 before feeding into the model.

Model implementation The backbone convolutional layers are taken from VGG-19 (Simonyan and Zisserman, 2015b). We pre-train the convolutional layers with a fingerspelling recognition task using the video-text pairs from the corresponding dataset. In pre-training, the VGG-19 layers are first pre-trained on ImageNet (Deng et al., 2009) and the image features further go through a 1-layer Bi-LSTM with 512 hidden units per direction. The model is trained with CTC loss (Graves et al., 2006). The output unit include English alphabet plus the few special symbols: <space>, ', &, ., @ as well as the blank symbol for CTC. The model is trained with SGD with batch size 1 at the initial learning rate of 0.01. The model is trained for 30 epochs and the learning rate is decayed to 0.001 after 20 epochs. The recognizer achieved 52.5%/64.4% lettter accuracy on ChicagoFSWild/ChicagoFSWild+ test sets. The VGG-19 convolutional layers are frozen in FSS-Net training.

In FSS-Net, the visual features output from convolutional layers are passed through a 1-layer Bi-LSTM with 256 hidden units per direction to capture temporal information. To generate proposals, we first transform the feature sequence via a 1D-CNN of following architecture: conv layer (512 output dimension, kernel width 8), max pooling (kernel width 8, stride 4), conv layer (256 output dimension, kernel width 3) and conv layer (256 output dimension, kernel width 3). The scale of anchors fixed from the range: $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 24, 32, 40, 60, 75\}$, chosen according to the typical fingerspelling lengths in the two

Figure 7: Example image frames from different sources in ChicagoFSWild and ChicagoFSWild+



datasets. The positve/negative threshold of the anchors are 0.6/0.3 respectively. The δ_{IoU}/δ_{IS} are 1.0/0.8. The FS-encoder and text encoder are 3-layer/1-layer BiLSTM with 256 hidden units respectively. The margin m, number of negative samples in $\mathcal{N}_v$ and $\mathcal{N}_w$ are tuned to be 0.45, 5 and 5. The model is trained for 25 epochs with Adam (Kingma and Ba, 2015) at initial learning rate of 0.001 and batch size of 32. The learning rate is halved if the mean average precision on dev set does not improve for 3 epochs. λ_{det} in equation 4 is 0.1. At test time, we generate $M = 50$ proposals after NMS with IoU threshold of 0.7. β is tuned to be 1.

Implementation of Attn-KWS The model assigns score to video clip $\mathbf{I}_{1:T}$ and word w as equation 9, where $\mathbf{e}_v^{1:T}$ is the visual feature sequence of $\mathbf{I}_{1:T}$ and $\mathbf{e}_x^w$ is the text feature of w, $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters. The model is trained with cross-entropy loss.

$$
\begin{aligned}
s(\mathbf{e}_v^t, \mathbf{e}_x^w) &= \beta\left(\frac{\mathbf{e}_v^t \cdot \mathbf{e}_x^w}{||\mathbf{e}_v^t|| \cdot ||\mathbf{e}_x^w||}\right)^2 + \theta \\
a(t) &= \frac{\exp(s(\mathbf{e}_v^t, \mathbf{e}_x^w))}{\sum_t \exp(s(\mathbf{e}_v^t, \mathbf{e}_x^w))} \\
sc(\mathbf{I}_{1:T}, \mathbf{e}_x^w) &= \sigma\left(\mathbf{W} \sum_{t=1}^{T} a(t)\mathbf{e}_v^t + \mathbf{b}\right)
\end{aligned}
\tag{9}
$$

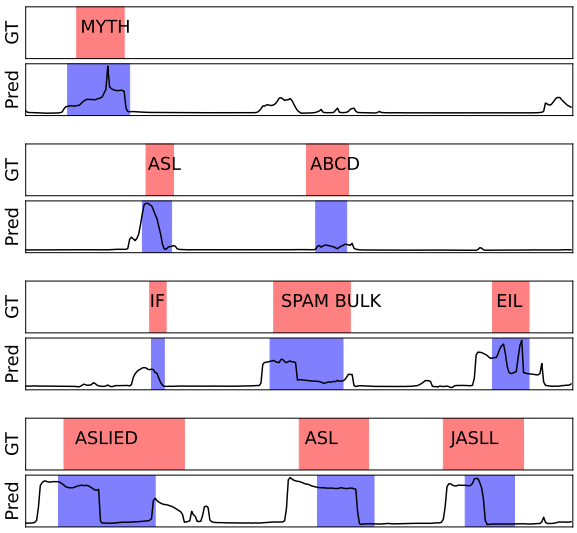
A.4 Full results

In addition to mAP and mF1, we also report ranking-based metrics: Precision@N and Recall@N (N=1, 10). For top-N retrieved X, we compute the percentage of correct X among N retrieved X as precision@N and the percentage of correct X among all correct X as recall@N, where X is text for FWS and video for FVS. Note the maximum value of R@1 and P@10 can be less than 1 as there are clips with multiple fingerspelling sequences and clips with fewer than 10 fingerspelling sequences. The performance of different models measured by all the above metrics is shown in table 7.

A.5 Examples of fingerspelling localization

Figure 8 shows examples fingerspelling localization produced by different methods.

Figure 8: Examples of fingerspelling localization produced by different methods. Upper: Ground-truth, Bottom: Attention weight curve and proposals generated by our model.



A.6 Precision-recall curve in FVS

Figure 9 shows the precision-recall curves of the most common words in ChicagoFSWild+ test set. Overall the performance of our model on frequent words is higher than average.

A.7 Qualitative examples of pose estimation

Figure 10 shows typical failure cases of pose estimation on ChicagoFSWild test set. The estimated hand pose is of low quality due to the motion blur and hand occlusion.

Table 7: FWS/FVS performance on the ChicagoFSWild and ChicagoFSWild+ test sets. Below each metric: (maximum). The minimum value of each metric is 0.

	FWS (Video ⟹ Text)						FVS (Text ⟹ Video)					
	ChicagoFSWild											
Method	mAP (1)	mF1 (1)	P@1 (1)	P@10 (.16)	R@1 (.75)	R@10 (1)	mAP (1)	mF1 (1)	P@1 (1)	P@10 (.17)	R@1 (.86)	R@10 (1)
Whole-clip	.175	.154	.116	.043	.092	.293	.142	.119	.106	.039	.070	.251
Attn-KWS	.204	.181	.158	.059	.108	.358	.246	.229	.238	.061	.179	.411
Recognizer	.318	.315	.352	.072	.284	.465	.331	.305	.323	.071	.220	.474
Ext-detector	.383	.385	.334	.085	.268	.529	.332	.312	.296	.079	.205	.510
FSS-Net	**.434**	**.439**	**.384**	**.093**	**.300**	**.591**	**.394**	**.370**	**.370**	**.091**	**.255**	**.580**
	ChicagoFSWild+											
Method	mAP (1)	mF1 (1)	P@1 (1)	P@10 (.16)	R@1 (.76)	R@10 (1)	mAP (1)	mF1 (1)	P@1 (1)	P@10 (.18)	R@1 (.84)	R@10 (1)
Whole-clip	.466	.457	.416	.100	.326	.626	.548	.526	.546	.101	.421	.711
Attn-KWS	.545	.530	.485	.112	.392	.727	.573	.547	.541	.111	.408	.748
Recognizer	.465	.462	.470	.094	.390	.620	.398	.405	.394	.090	.292	.617
Ext-detector	.633	.641	.589	.118	.491	.769	.593	.577	.568	.114	.419	.786
FSS-Net	**.674**	**.677**	**.637**	**.123**	**.530**	**.796**	**.638**	**.631**	**.596**	**.123**	**.442**	**.825**

Figure 9: FVS precision-recall curve of common words in ChicagoFSWild+ test set. Inside (): mAP

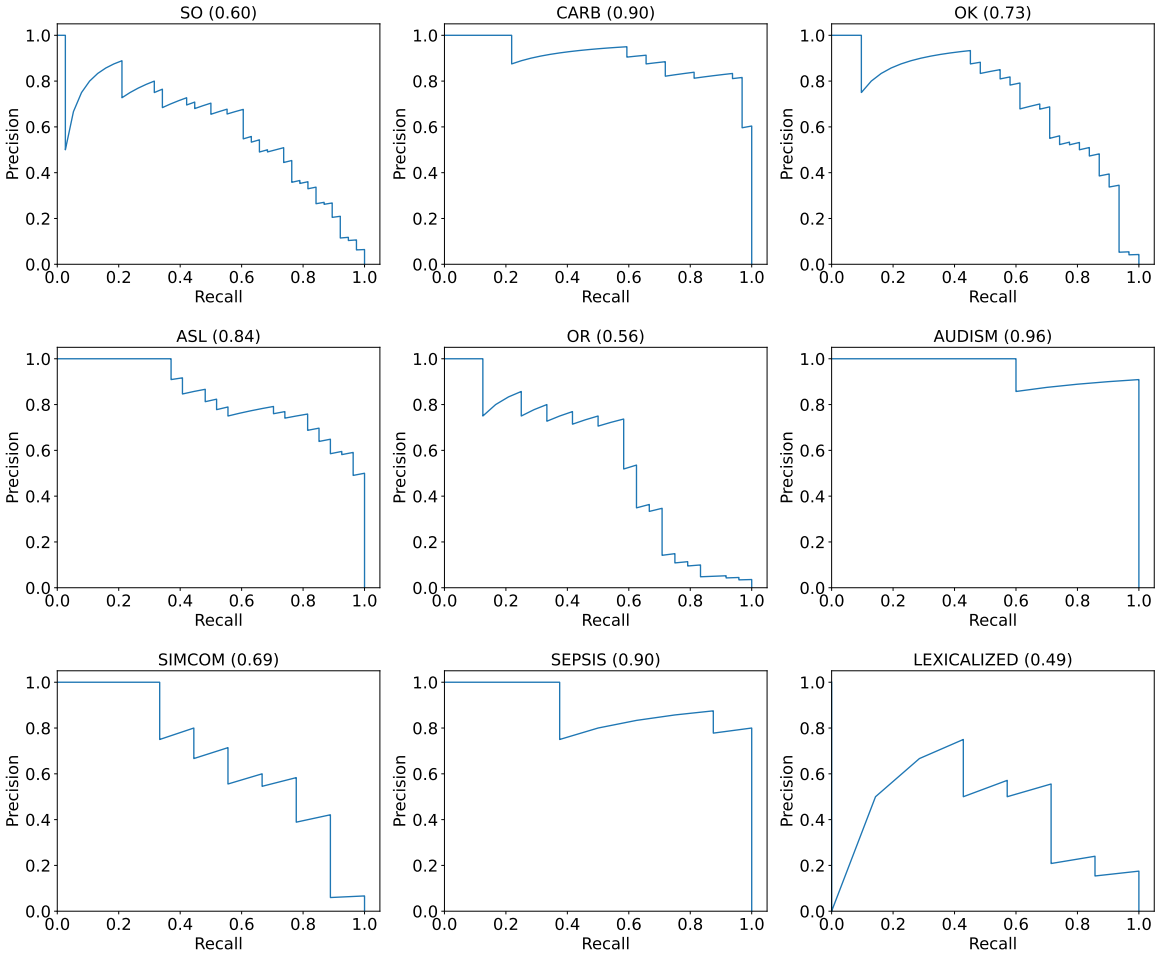


Figure 10: Estimated signer pose using Openpose on ChicagoFSWild test set