Fingerspelling Detection in American Sign Language

Bowen Shi[1], Diane Brentari[2], Greg Shakhnarovich[1], Karen Livescu[1]

[1]Toyota Technological Institute at Chicago, USA [2]University of Chicago, USA

{bshi,greg,klivescu}@ttic.edu   dbrentari@uchicago.edu

Figure 1: Fingerspelling detection and recognition in video of American Sign Language. The goal of *detection* is to find intervals corresponding to fingerspelling (here indicated by open/close parentheses), and the goal of *recognition* is to transcribe each of those intervals into letter sequences. Our focus in this paper is on detection that enables accurate recognition. In this example (with downsampled frames), the fingerspelled words are PIRATES and PATRICK, shown along with their canonical handshapes aligned roughly with the most-canonical corresponding frames. Non-fingerspelled signs are labeled with their glosses. The English translation is "Moving furtively, pirates steal the boy Patrick."

Abstract

Fingerspelling, in which words are signed letter by letter, is an important component of American Sign Language. Most previous work on automatic fingerspelling recognition has assumed that the boundaries of fingerspelling regions in signing videos are known beforehand. In this paper, we consider the task of fingerspelling detection in raw, untrimmed sign language videos. This is an important step towards building real-world fingerspelling recognition systems. We propose a benchmark and a suite of evaluation metrics, some of which reflect the effect of detection on the downstream fingerspelling recognition task. In addition, we propose a new model that learns to detect fingerspelling via multi-task training, incorporating pose estimation and fingerspelling recognition (transcription) along with detection, and compare this model to several alternatives. The model outperforms all alternative approaches across all metrics, establishing a state of the art on the benchmark.

1. Introduction

Sign languages, such as American Sign Language (ASL), are natural languages expressed via movements of the hands, face, and upper body. Automatic processing of sign languages would assist communication between deaf and hearing individuals, but involves a number of challenges. There is no standard written form for sign languages. Automatic transcription of sign language into a written language such as English is in general a translation task. In addition, sign language gestures are often coarticu-

lated and do not appear in their canonical forms [22, 25].

In this paper, we focus on fingerspelling (Figure 1), a component of sign language in which words are signed letter by letter, with a distinct handshape or trajectory corresponding to each letter in the alphabet of a written language (e.g., the English alphabet for ASL fingerspelling). Fingerspelling is used for multiple purposes, including for words that do not have their own signs (such as many proper nouns, technical terms, and abbreviations) [39] but also sometimes for emphasis or expediency. Fingerspelling accounts for 12% to 35% of ASL, where it is used more than in other sign languages [40]. As important content words are commonly fingerspelled, automatic fingerspelling recognition can enable practical tasks such as search and retrieval in ASL media.

Compared to *translation* between a sign language and a written language, fingerspelling recognition involves *transcription* into a restricted set of symbols, with a monotonic alignment with the written form. Linguistically, fingerspelling is distinct from other elements of ASL, such as lexical signs and classifiers [23, 4], so fingerspelling is likely to benefit from a separate model. The role that fingerspelling transcription is likely to play in ASL to English translation is similar to that of transliteration in written language translation [11]. For all of these reasons, we believe that even as more general ASL processing methods are developed, it will continue to be beneficial to have dedicated fingerspelling detection and recognition modules.

Fingerspelling recognition has been widely studied [42, 17, 45, 37, 26, 46]. However, in most prior work, it is assumed that the input sequence contains fingerspelling only, sometimes extracted from longer sequences of sign-

ing via human annotation. Replacing human annotation with fully automatic *detection* of fingerspelling – identifying time spans in the video containing fingerspelling – is a hurdle that must be cleared to enable truly practical fingerspelling recognition "in the wild".

Fingerspelling detection has not been widely studied before. In principle it can be treated as a special case of action detection [53, 7, 56, 12]. However, in contrast to typical action detection scenarios, the actions in the fingerspelling "class" are highly heterogeneous and many fingerspelling handshapes are also used in non-fingerspelled signs. In addition, considering the goal of using the detector as part of a complete sign language processing system, a fingerspelling detector should be evaluated based on its effect on a downstream recognition model, a step not normally included in evaluation of action recognition. This makes common detection metrics, like average precision (AP) for action detection, less informative for fingerspelling detection.

Our design of a detection model is motivated by two observations. The first is that articulated pose, in particular handshape, plays a role in the distinctiveness of fingerspelling from other types of sign. At the same time, pose estimation, while increasingly successful in some domains, may be insufficiently accurate for directly informing fingerspelling recognition, as shown in [46] and in our experiments. Instead we incorporate pose estimation as part of training our model, but do not rely on explicit pose estimates at test time. The second observation concerns the goal of optimizing fingerspelling detection as a means to an end of improving downstream recognition. We address this by including a fingerspelling recognizer in model training. Our results show that this multi-task learning approach produces a superior detector compared to baselines that omit the pose and/or recognition losses.

Ours is to our knowledge the first work to demonstrate the effect of fingerspelling detection on fully automatic fingerspelling recognition in the wild. Beyond this novelty, our contributions are as follows. First, we propose an evaluation framework for fingerspelling detection that incorporates the downstream recognition task into the metrics, and introduce a benchmark based on extending a publicly available data set. Second, we investigate a number of approaches for fingerspelling detection, adapted from fingerspelling recognition and action detection, and develop a novel multi-task learning approach. Our model outperforms baseline detection approaches across all evaluation metrics, establishing a state of the art for the proposed benchmark.

2. Related Work

Early work on sign language recognition mostly focused on recognition of isolated signs. Recent work has increasingly focused on continuous sign language recognition, which transforms image sequences to glosses (for gen-

eral sign language) or letter sequences (for fingerspelling), and translation. Approaches commonly separate between an image feature extraction component and a sequence modeling component. The former has moved from employing human-engineered features such as HoG [26, 28] to convolutional networks [29, 31, 47]. Approaches for the sequence modeling component have included Hidden Markov Models (HMM) [31, 30, 27], semi-Markov conditional random fields [26], connectionist temporal classification (CTC) [47, 46, 9], and encoder-decoder models [20, 5], which are largely borrowed from speech recognition.

Much of the prior work has been limited to data collected in a controlled environment. There has been a growing interest in sign language recognition "in the wild" (naturally occurring sign language media), which includes challenging visual conditions such as lighting variation, visual clutter, and motion blur, and often also more natural signing styles [15, 16, 3, 24, 32, 33, 47, 46]. Two recently released datasets of fingerspelling in the wild [47, 46] include data from 168 signers and tens of thousands of fingerspelling segments; these are the testbeds used in our experiments.

Prior work on fingerspelling detection [55, 51, 50, 54] employs visual features from optical flow or pre-defined hand keypoints. For sign language data collected in the wild, the quality of pose estimates is usually low, making them a poor choice as input to a detector (as we show in our experiments). Tsechpenakis *et al.* [51, 50] proposed a statistical procedure to detect changes in video for fingerspelling detection; however, these were tested anecdotally, in controlled environments and simplified scenarios. Related work on detection of sign language segments (in video containing both signing and non-signing) uses recurrent neural networks (RNN) to classify individual frames into signing/non-signing categories [38]. We compare our approach to baselines that use ideas from this prior work.

Prior work [38, 55] has largely treated the detection task as frame classification, evaluated via classification accuracy. Though sequence prediction is considered in [51, 50], the model evaluation is qualitative. In practice, a detection model is often intended to serve as one part of a larger system, producing candidate segments to a downstream recognizer. Frame-based metrics ignore the quality of the segments and their effect on a downstream recognizer.

Our modeling approach is based on the intuition that training with related tasks including recognition and pose estimation should help in detecting fingerspelling segments. Multi-task approaches have been studied for other related tasks. For example, Li *et al.* [34] jointly train an object detector and a word recognizer to perform text spotting. In contrast to this approach of treating detection and recognition as two parallel tasks, our approach further allows the detector to account for the performance of the recognizer. In sign language recognition, Zhou *et al.* [57] estimate hu-

man pose keypoints while training the recognizer. However, the keypoints used in [57] are manually annotated. Here we study whether we can distill knowledge from an external imperfect pose estimation model (OpenPose).

3. Task and metrics

We are given a sign language video clip with N frames $I_1, ..., I_N$ containing n fingerspelling segments $\{x_i^*\}_{1 \le i \le n}$ (∗ denotes ground truth), where $x_i^* = (s_i^*, t_i^*)$ and s_i^*, t_i^* are the start and end frame indices of the ith segment. The corresponding ground-truth letter sequences are $\{\mathbf{l}_i^*\}_{1 \le i \le n}$.

Detection task The task of fingerspelling detection is to find the fingerspelling segments within the clip. A detection model outputs m predicted segment-score pairs $\{(\widehat{x}_i, f_i)\}_{1 \le i \le m}$ where $\widehat{x}_i$ and f_i are the ith predicted segment and its confidence score, respectively.

AP@IoU A metric commonly used in object detection [36] and action detection [21, 19] is AP@IoU. Predicted segments, sorted by score f_i, are sequentially matched to the ground-truth segment x_j^* with the highest IoU (intersection over union, a measure of overlap) above a threshold δ_{IoU}. Once x_j^* is matched to a $\widehat{x}_i$ it is removed from the candidate set. Let $k(i)$ be the index of the ground-truth segment $x_{k(i)}^*$ matched to $\widehat{x}_i$; then formally,

$$k(i) = \underset{j: IoU(\widehat{x}_i, x_j^*) > \delta_{IoU}, \, j \neq k(t) \forall t < i}{\arg\max} IoU(\widehat{x}_i, x_j^*). \quad (1)$$

Precision and recall are defined as the proportions of matched examples in the predicted and ground-truth segment sets, respectively. Varying the number of predictions m gives a precision-recall curve $p(\tilde{r})$. The average precision (AP) is defined as the mean precision over $N_r + 1$ equally spaced recall levels $[0, 1/N_r, ..., 1]$[1], where the precision at a given recall is defined as the maximum precision at a recall exceeding that level: $AP = \frac{1}{N_r} \sum_{i=1}^{N_r} \max_{\tilde{r}: \tilde{r} \ge i/N_r} p(\tilde{r})$. AP@IoU can be reported for a range of values of δ_{IoU}.

Recognition task The task of recognition is to transcribe $\widehat{x} = (\widehat{s}, \widehat{t})$ into a letter sequence $\widehat{\mathbf{l}}$. The recognition accuracy of a predicted sequence $\widehat{\mathbf{l}}$ w.r.t. the ground truth $\mathbf{l}^*$ is defined as $\text{Acc}(\mathbf{l}^*, \widehat{\mathbf{l}}) = 1 - D(\mathbf{l}^*, \widehat{\mathbf{l}})/|\mathbf{l}^*|$, where D is the edit (Levenstein) distance and $|\mathbf{l}|$ is the length of a sequence. Note that Acc can be negative.

Prior work has considered recognition mainly applied to ground-truth segments x^*; in contrast, here we are concerned with detection for the purpose of recognition. We match a predicted $\widehat{x}_j$ to a ground-truth x_i^* and then evaluate the accuracy of $\widehat{\mathbf{l}}_j$ w.r.t. $\mathbf{l}_i^*$. Thus, in contrast to a typical action detection scenario, here IoU may not be perfectly correlated with recognition accuracy. For example, a detected

---

[1] N_r is set to 100 as in [36].

---

segment that is too short can hurt recognition much more than a segment that is too long. We propose a new metric, AP@Acc, to measure the performance of a fingerspelling detector in the context of a given downstream recognizer.

AP@Acc This metric uses the letter accuracy of a recognizer to match between predictions and ground-truth. It also requires an IoU threshold to prevent matches between non-overlapping segments. As in AP@IoU, predicted segments are sorted by score and sequentially matched:

$$k(i) = \underset{\substack{j: IoU(\widehat{x}_i, x_j^*) > \delta_{IoU}, \, j \neq k(t) \forall t < i \\ Acc(\mathbf{l}_j^*, Rec(I_{\widehat{s}_i : \widehat{t}_i})) > \delta_{acc}}}{\arg\max} Acc(\mathbf{l}_j^*, Rec(I_{\widehat{s}_i : \widehat{t}_i})), \quad (2)$$

where $Rec(I_{s:t})$ is the output (predicted letter sequence) from a recognizer given the frames $I_{s:t}$. We can report AP@Acc for multiple values of δ_{acc}.

Maximum Sequence Accuracy (MSA) Both AP@IoU and AP@Acc measure the precision of a set of detector predictions. Our last metric directly measures just the performance of a given downstream recognizer when given the detector's predictions. We form the ground-truth letter sequence for the entire video $I_{1:N}$ by concatenating the letters of all ground-truth segments, with a special "no-letter" symbol $\emptyset$ separating consecutive letter sequences:

$$\mathbf{L}^* = \emptyset, \mathbf{l}_1^*, \emptyset, \ldots, \emptyset, \mathbf{l}_n^*, \emptyset. \quad (3)$$

Note that $\emptyset$ is inserted only where non-fingerspelling frames exist. For instance, the video in Figure 1 would yield $\mathbf{L}^* = \text{PIRATES}\emptyset\text{PATRICK}$. We similarly obtain a full-video letter sequence from the predicted segments. We suppress detections with score f_i below δ_f and apply local non-maximum suppression, resulting in a set of non-overlaping segments $\widehat{x}_1, \ldots, \widehat{x}_n$. Each of these is fed to the recognizer, producing $\widehat{\mathbf{l}}_i = Rec(I_{\widehat{s}_i : \widehat{t}_i})$. Concatenating these in the same way as in (3) gives us the full-video predicted letter sequence $\widehat{\mathbf{L}}(\delta_f)$. We can now treat $\mathbf{L}^*$ and $\widehat{\mathbf{L}}$ as two letter sequences, and compute the transcription accuracy. Maximum sequence accuracy (MSA) is defined as

$$MSA = \max_{\delta_f} Acc(\mathbf{L}^*, \widehat{\mathbf{L}}(\delta_f)). \quad (4)$$

Like AP@Acc, MSA depends on both the detector and the given recognizer Rec. By comparing the MSA for a given detector and for an "oracle detector" that produces the ground-truth segments, we can obtain an indication of how far the detector output is from the ground-truth.

4. Models for fingerspelling detection

Figure 2 sketches several baseline models and our proposed model, described below.

4.1. Baseline models

Baseline 1: Frame-based detector This model classifies every frame as positive (fingerspelling) or negative
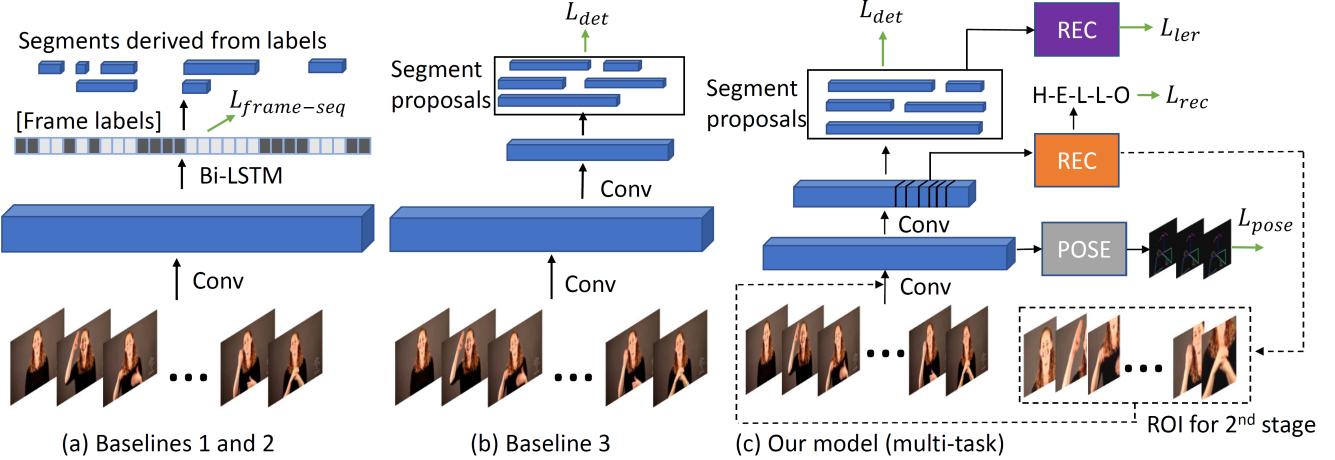
Figure 2: Detection models. (a) Baselines 1 and 2. The frame labels are binary labels and augmented letter labels, respectively, for baselines 1 and 2. $L_{frame-seq}$ is a per-frame cross-entropy loss for baseline 1 and modified CTC loss for baseline 2. (b): Baseline 3, modified R-C3D; L_{det} is the standard detection loss. (c) Our multi-task model trained with a combination of detection loss, recognition loss using a recognizer that takes in the ground-truth segments; letter error loss using a recognizer that takes in the detections; and pose estimation loss computed relative to pose pseudo-labels.

(non-fingerspelling), and is trained using the per-frame cross-entropy loss, weighted to control for class imbalance. Each frame is passed through a convolutional network to extract visual features, which are then passed to a multi-layer bidirectional long short-term memory RNN (LSTM) to take temporal context into account, followed by a linear/sigmoid layer producing per-frame class posterior probabilities.

To convert the frame-level outputs to predicted segments, we first assign hard binary frame labels by thresholding the posteriors by $\bar{p}$. Contiguous sequences of positive labels are taken to be the predicted segments, with a segment score f computed as the average posterior of the fingerspelling class for its constituent frames. We repeat this process for decreasing values of $\bar{p}$, producing a pool of (possibly overlapping) segments. Finally, these are culled using thresholding on f and non-maximum suppression limiting overlap, producing a set of $\widehat{x}$s as the final output.

Baseline 2: Recognition-based detector Instead of classifying each frame into two classes, this baseline addresses the task as a sequence prediction problem, where the target is the concatenation of true letter sequences separated by $\emptyset$ indicating non-fingerspelling segments, as in (3). The frame-to-letter alignment in fingerspelling spans is usually unknown during training, so we base the model on connectionist temporal classification (CTC) [18]: We generate frame-level label softmax posteriors over possible letters, augmented by $\emptyset$ and a special *blank* symbol, and train by maximizing the marginal log probability of label sequences that produce the true sequence under CTC's "label collapsing function" that removes duplicate frame labels and then *blank*s (we refer to this as the CTC loss).

In this case we have a partial alignment of sequences and frame labels, since we know the boundaries between $\emptyset$ and fingerspelling segments, and we use this information to explicitly compute a frame-level log-loss in non-fingerspelling regions. This modification stabilizes training and improves performance. At test time, we use the model's per-frame posterior probability of fingerspelling, $1 - p(\emptyset)$, and follow the same process as in baseline 1 (frame-based) to convert the per-frame probabilities to span predictions.

Baseline 3: Region-based detector This model directly predicts variable-length temporal segments that potentially contain fingerspelling, and is adapted from R-C3D [53], a 3D version of the Faster-RCNN [43]. The model first applies a 2D ConvNet on each frame; additional 3D convolutional layers are applied on the whole feature tensor to capture temporal information. Unlike [53], we did not directly apply an off-the-shelf 3D ConvNet such as C3D [49], since its large stride may harm our ability to capture the delicate movements and often short sequences in fingerspelling.

A region proposal network is applied to the feature tensor, predicting shifts of potential fingerspelling segments with respect to (temporal, 1D) anchors, and a binary label indicating whether the predicted proposal contains fingerspelling. The detector is trained with a loss composed of two terms for (binary) classification and (positional) regression, $L_{det} = L_{cls} + L_{reg}$. See [53] for further details. At test time, we use greedy non-maximum suppression (NMS) on fingerspelling proposals to eliminate highly overlapping and low-confidence intervals. Since fingerspelling detection is a binary classification task, we do not use a second stage classification subnet as in the original RC3D [53].

4.2. Our approach: A multi-task model

Our model is based on the region-based detector, with the key difference being that fingerspelling recognition and pose estimation are incorporated into training the model.

Recognition loss is computed by passing the fingerspelling segments to a recognizer. Our intuition is that including the recognition task may help the model learn richer features for fingerspelling, which may improve its ability to distinguish between fingerspelling and non-fingerspelling. The recognizer here plays a role similar to the classification subnet in RC3D. But since we don't assume that frame-letter alignment is available at training time, we directly build a sub-network for letter sequence prediction (the orange "REC" in Figure 2).

The recognition sub-network follows the attention-based model proposed for fingerspelling recognition in [46], using only the ground-truth segment for the recognition loss. The recognition loss L_{rec} is computed as the CTC loss summed over the proposed regions predicted by the detector:

$$L_{rec} = \sum_{i=1}^{n} L_{ctc}(Rec(I_{s_i^\star : t_i^\star}), \mathbf{l}_i^*) \qquad (5)$$

where n is the number of true fingerspelling segments.

Letter error rate loss Though L_{rec} should help learn an improved image feature space, the performance of the detector does not impact the recognizer directly since L_{rec} uses only the ground-truth segments. L_{det} encourages the model to output segments that are spatially close to the ground truth. What's missing is the objective of making the detector work well *for the downstream recognition*. To this end we add a loss measuring the letter error rate of a recognizer applied to proposals from the detector:

$$L_{ler} = -\sum_{i=1}^{m} p(\widehat{x}_i) Acc(\mathbf{l}_{k(i)}^*, Rec(I_{\widehat{s}_i : \widehat{t}_i})), \qquad (6)$$

where as before, $k(i)$ is the index of the ground-truth segment matched (based on IoU) to the ith proposal $\widehat{x}_i = (s_i, t_i)$, m is the number of proposals output by the detector, and Acc is the recognizer accuracy. The loss can be interpreted as the expectation of the negative letter accuracy of the recognizer on the proposals given to it by the detector. Since Acc, which depends on the edit distance, is non-differentiable, we approximate the gradient of (6) as in REINFORCE [52]:

$$\nabla L_{ler} \approx -\sum_{i=1}^{M} p(\widehat{x}_i) Acc(\mathbf{l}_{k(i)}^*, Rec(I_{\widehat{s}_i : \widehat{t}_i}) \nabla \log p(\widehat{x}_i), \qquad (7)$$

where the sum is over the M highest scoring proposals, and $p(\widehat{x}_i)$ is the normalized score $f_i / \sum_{i=1}^{M} f_i$.

Pose estimation loss Since sign language is to a large extent based on body articulation, it is natural to consider incorporating pose into the model. However, we can not assume access to ground-truth pose even in the training data. Instead, we can rely on general-purpose human pose estimation models, such as OpenPose [6], trained on large sets of annotated images.

Prior work on sign language has used pose estimates extracted from an external model as input to models for sign language-related tasks [24, 38, 41, 32, 8]. On the controlled studio data used in that work, the quality of extracted keypoints is reliable. On more challenging real-world data like ChicagoFSWild/ChicagoFSWild+ [46], we find the detected keypoints from OpenPose to be much less reliable, presumably in part due to the widespread motion blur in the data (see Figure 5 for examples). Indeed, as we show in our experiments, using automatically extracted poses as input to the model does not significantly improve performance.

Instead of relying on estimated pose at test time, we treat the estimated pose as a source of additional supervision for our model at training time. We use keypoints from Open-Pose as pseudo-labels to help distill knowledge from the pre-trained pose model into our detection model. As pose is used only as an auxiliary task, the quality of those pseudo-labels has less impact on detection performance than does using pose as input at test time.

The pose estimation sub-network takes the feature maps extracted by the model (shared with the detector and recognizer) and applies several transposed convolutional layers to increase spatial resolution, producing for frame I_t a set of heat maps $\mathbf{b}_{t,1}, \ldots, \mathbf{b}_{t,P}$ for P keypoints in the OpenPose model. We also use OpenPose to extract keypoints from I_t; each estimated keypoint p is accompanied by a confidence score $\sigma_{t,p}$. We convert these estimates to pseudo-ground truth heatmaps $\mathbf{b}_{t,1}^*, \ldots, \mathbf{b}_{t,P}^*$. The pose loss is the per-pixel Euclidean distance between the predicted and pseudo-ground truth maps:

$$L_{pose} = \sum_{t=1}^{T} \sum_{p=1}^{P} \|\mathbf{b}_{t,p} - \mathbf{b}_{t,p}^*\|^2 \cdot \mathbf{1}_{\sigma_{t,p} > \tau}, \qquad (8)$$

where the threshold on the OpenPose confidence is used to ignore low-confidence pseudo-labels.

Putting it all together, the loss for training our model is

$$L = L_{det} + \lambda_{rec} L_{rec} + \lambda_{ler} L_{ler} + \lambda_{pose} L_{pose}, \qquad (9)$$

with tuned weights $\lambda_{ler}, \lambda_{rec}, \lambda_{pose}$ controlling the relative importance of different loss components.

Second stage refinement Since we expect reasoning about fine-grained motion and handshape differences to be helpful, we would like to use high-resolution input images. However, since the input video clip covers hundreds of frames, the images need to be downsampled to fit into the memory of a typical GPU.

To mitigate the issue of low resolution in local regions, we "zoom in" on the original image frames using the attention map produced by the reocognizer sub-network. This idea is based on the iterative attention approach of [46]. As large attention values indicate higher importance of the corresponding region for fingerspelling, cropping the ROI surrounding a location with high attention values helps increase the resolution of that part while avoiding other irrelevant areas. To achieve this, we first complete training of the whole model described above. Suppose the image sequence of original resolution is $I^o_{1:N}$, the lower-resolution frame sequence used in the first round of training is $I^g_{1:N}$, and the trained model is $\mathcal{H}$. We run inference on $I^g_{1:N}$ with the recognition sub-network $\mathcal{H}_{rec}$ to produce a sequence of attention maps $\mathbf{A}_{1:N}$. We use $\mathbf{A}_{1:N}$ to crop $I^o_{1:N}$ into a sequence of local ROIs $I^l_{1:N}$. Specifically, at timestep n, we put a bounding box b_n of size $R|I_n|$ centered on the peak of attention map $\mathbf{A}_n$, where R is the zooming factor. We average the bounding boxes of the $2a + 1$ frames centered on the nth frame to produce a smoother cropping "tube" $b^s_{1:N}$:

$$b^s_n = \frac{1}{2a+1} \sum_{i=-a}^{a} (b_{n+i}) \qquad (10)$$

The local ROIs $I^l_{1:N}$ are cropped from $I^o_{1:N}$ with $b^s_{1:N}$.

Finally, we perform a second stage of training with both $I^g_{1:N}$ and $I^l_{1:N}$ as input. At each timestep n, the backbone conv layers are applied on I^g_n and I^l_n to obtain global and local feature maps, respectively. The two feature maps are concatenated for detection and recognition. The pseudo-ground truth pose maps are estimated on $I^{(g)}_t$ and $I^{(l)}_t$ separately. The overall loss for the second-stage training is

$$L^{\text{final}} = L_{det} + \lambda_{rec}L_{rec} + \lambda_{ler}L_{ler} \\ + \lambda_{pose}(L^{(g)}_{pose} + L^{(l)}_{pose}) \qquad (11)$$

The key difference between our approach and the iterative attention of [46] is that we do not drop the input images, but rather use the newly generated ROIs as extra input. In fingerspelling detection, both global context (e.g., upper body position) and local details (e.g., handshape) are important.

5. Experiments

5.1. Setup

We conduct experiments on ChicagoFSWild [47] and ChicagoFSWild+ [46], two large-scale ASL fingerspelling datasets collected in the wild. Though the datasets were introduced for fingerspelling recognition (with the boundaries given), the URLs of the raw ASL videos and the fingerspelling start/end timestamps are provided. We split each video clip into 300-frame chunks (∼12s) with a 75-frame overlap between chunks. The longest fingerspelling

sequence in the data is 290 frames long. We use the same training/dev/test data split as in the original datasets (see additional data statistics in the supplementary material). The image frames are center-cropped and resized to 108×108.

We take the convolutional layers from VGG-19 [48] pretrained on ImageNet [10] as our backbone network, and fine-tune the weights during training. For baselines 1 and 2, an average pooling layer is applied on the feature map, giving a 512-dimensional vector for each frame, which is fed into a one-layer Bi-LSTM with 512 hidden units. In baseline 3 and our model, the feature map is further passed through a 3D conv + maxpooling layer (with temporal stride 8). In the region proposal network, the lengths of the anchors are fixed at 12 values ranging from 8 to 320, which are chosen according to the typical lengths of fingerspelling sequences in the data. The IoU thresholds for positive/negative anchors are respectively 0.7/0.3. The predicted segments are refined with NMS at a threshold of 0.7. The magnitude of optical flow is used as a prior attention map as in [46]. We use the same recognizer (Rec) for (5) and (6). The pose sub-net is composed of one transposed convolutional layer with stride 2. We use OpenPose [6] to extract 15 body keypoints and 2×21 hand keypoints (both hands) as pseudo pose labels. Keypoints with confidence below $\tau = 0.5$ are dropped. For second-stage refinement, the moving averaging of bounding boxes in (10) uses 11 frames ($a = 5$) and the frame sequence is downsampled by a factor of 2 to save memory. The loss weights (λs) are tuned on the dev set.

To evaluate with AP@Acc and MSA, we train a reference recognizer with the same architecture as the recognition-based detector on the fingerspelling training data. The recognizer achieves an accuracy (%) of 44.0/62.2 on ground-truth fingerspelling segments on ChicagoFSWild/ChicagoFSWild+. The accuracy is slightly worse than that of [46] because we used a simpler recognizer. In particular, we skipped the iterative training in [46], used lower-resolution input, and did not use a language model. We consider $\delta_{IoU} \in \{0.1, 0.3, 0.5\}$ and $\delta_{acc} \in \{0, 0.2, 0.4\}$; δ_{IoU} is fixed at 0 for AP@Acc.

5.2. Results

Main results Table 1 compares models using the three proposed metrics. The values of AP@Acc and MSA depend on the reference recognizer. For ease of comparison, we also show the oracle results when the same recognizer is given the ground-truth fingerspelling segments. Overall, the relative model performance is consistent across metrics. Methods that combine detection and recognition outperform those that do purely detection (baseline 2 vs. 1, our model vs. baseline 3). In addition, region-based methods (baseline 3 and our model) outperform frame-based methods (baseline 1 & 2), whose segment predictions lack

Table 1: Model comparison on the ChicagoFSWild and ChicagoFSWild+ test sets. BMN refers to boundary matching network [35]. The right column (GT) shows results when the "detections" are given by ground-truth fingerspelling segments.

ChicagoFSWild		Base 1	Base 2	Base 3	BMN	Ours	GT
AP@ IoU	AP@0.1	.121	.310	.447	.442	**.495**	1.00
	AP@0.3	.028	.178	.406	.396	**.453**	1.00
	AP@0.5	.012	.087	.318	.284	**.344**	1.00
AP@ Acc	AP@0.0	.062	.158	.216	.209	**.249**	.452
	AP@0.2	.028	.106	.161	.157	**.181**	.349
	AP@0.4	.006	.034	.069	.070	**.081**	.191
MSA		.231	.256	.320	.307	**.341**	.452
(b). ChicagoFSWild+		Base 1	Base 2	Base 3	BMN	Ours	GT
AP@ IoU	AP@0.1	.278	.443	.560	.580	**.590**	1.00
	AP@0.3	.082	.366	.525	.549	**.562**	1.00
	AP@0.5	.025	.247	.420	.437	**.448**	1.00
AP@ Acc	AP@0.0	.211	.323	.426	.433	**.450**	.744
	AP@0.2	.093	.265	.396	.401	**.415**	.700
	AP@0.4	.029	.152	.264	.260	**.277**	.505
MSA		.267	.390	.477	.470	**.503**	.630

smoothness. We can see this also by examining the frame-level performance of the three baselines.[2] Their frame-level average precisions are 0.522 (baseline 1), 0.534 (baseline 2), and 0.588 (baseline 3), which are much closer than the segment-based metrics, showing how frame-level metrics can obscure important differences between models. More details on precision and recall can be found in the supplementary material. The trends in results are similar on the two datasets. For the remaining analysis we report results on the ChicagoFSWild dev set.

In addtion to baselines adapted from related tasks, Table 1 includes a comparison to a recent method developed for action detection: the boundary matching network [35]. We also compared to the multi-stage temporal convolutional network [13] (details in the supplementary). Neither approach is better than ours on the fingerspelling benchmark.

Analysis of evaluation metrics The AP@Acc results are largely invariant to δ_{IoU} (see supplementary material) and we report results for $\delta_{IoU} = 0$ (i.e., matching ground truth segment to detections based on accuracy, subject to non-zero overlap). We also examine the relationship between sequence accuracy and the score threshold of each model (Figure 3). Our model achieves higher sequence accuracy across all thresholds. The threshold producing the best accuracy varies for each model. The average IoUs of the four models for the optimal threshold δ_f are 0.096, 0.270, 0.485, and 0.524 respectively.

Ablation study Our model reduces to baseline 3 when all loss terms except the detection loss are removed. Table 2

---

[2]For the region-based model (baseline 3) we take the maximum probability of all proposals containing a given frame as that frame's probability.
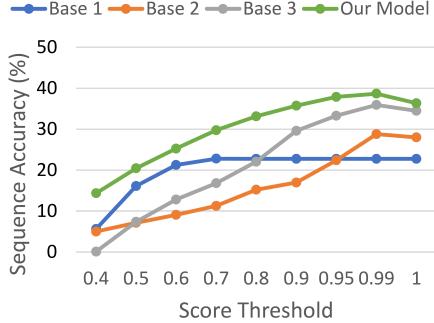

Figure 3: Dependence of sequence accuracy on score threshold δ_f.

shows how model performance improves as more tasks are added. The gain due to the recognition loss alone is smaller than for the frame-based models (base 1 vs. base 2). The recognition sub-network contributes more through the LER loss, which communicates the recognition error to the detector directly. The second-stage refinement does not always boost AP@IoU; the gain is more consistent in the accuracy-based metrics (AP@Acc, MSA). The zoom-in effect of the second-stage refinement increases the resolution of the hand region and improves recognition, though the IoU may remain similar. The downsampling in the second-stage refinement also leads to some positioning error, reflected in the slight drop in AP at IoU=0.5, though a minor temporal shift does not always hurt recognition.

Examples Figure 4 shows examples in which our model correctly detects fingerspelling segments while baseline 3 fails. Fingerspelling can include handshapes that are visually similar to non-fingerspelled signs (Figure 4a). Fingerspelling recognition, as an auxiliary task, may improve detection by helping the model distinguish among fine-grained handshapes. The signer's pose may provide additional information for detecting fingerspelling (Figure 4b). Figure 4c shows an example where the signing hand is a small portion of the whole image; baseline 3 likely fails due to the low resolution of the signing hand. The second-stage
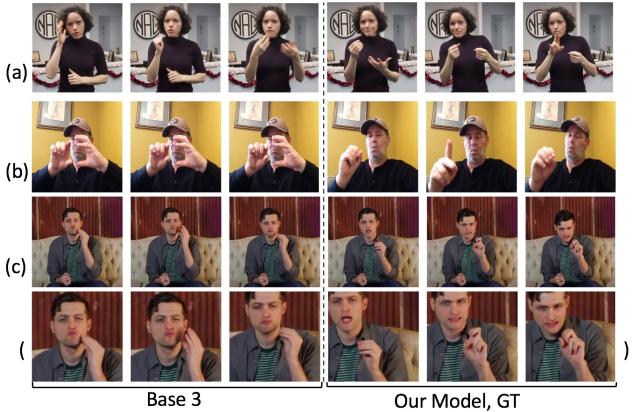

Figure 4: Example segments detected by different models. Bottom row: ROIs used in second-stage refinement. GT: Ground-truth segment. The sequence is downsampled.

Table 2: Impact of adding loss components to our model training. Results are on the ChicagoFSWild dev set.

		Base 3	+ rec	+LER	+pose	+2nd stage
AP@ IoU	AP@0.1	.496	.505	.531	.539	.551
	AP@0.3	.466	.479	.498	.500	.505
	AP@0.5	.354	.362	.392	.399	.393
AP@ Acc	AP@0.0	.285	.290	.299	.302	.313
	AP@0.2	.222	.231	.245	.255	.267
	AP@0.4	.094	.097	.103	.107	.112
MSA		.359	.365	.378	.383	.386

Table 3: Comparison among modalities for the region-based detector (baseline 3) on the ChicagoFSWild dev set. RGB+Pose(in): both RGB and pose image used as input. RGB+Opt(in): both RGB and optical flow used as input. RGB+Pose(out): detector trained jointly with pose estimation (our model).

AP@IoU	RGB	Pose	Opt	RGB+Opt (in)	RGB+Pose (in)	RGB+Pose (out)
AP@0.1	.496	.368	.476	.503	.501	**.505**
AP@0.3	.466	.332	.438	.472	.470	**.478**
AP@0.5	.354	.237	.315	.357	.346	**.366**

refinement enables the model to access a higher-resolution ROI, leading to a correct detection.

Error analysis Qualitatively, we notice three common sources of false positives: (a) near-face sign, (b) numbers, and (c) fingerspelling handshape used in regular signing (see Figure 6). Such errors can potentially be reduced by incorporating linguistic knowledge in the detector, which is left as future work. We also observe that short fingerspelling segments are harder to detect. See the supplementary material for the performance breakdown according to length.

Other input modalities Our model is trained on RGB images. Motion and pose are two common modalities used in sign language tasks [24, 38, 55], so it is natural to ask whether they would be helpful instead of or in addition to RGB input. We use magnitude of optical flow [14] as the motion image and the pose heatmap from OpenPose as the pose image. Raw RGB frames as input outperform the other
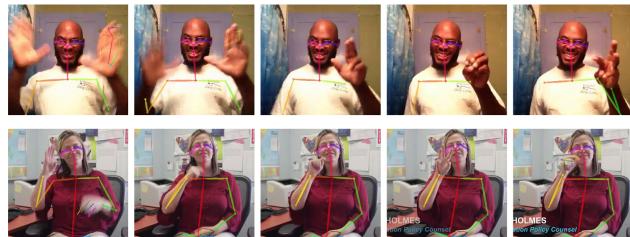
two modalities, although each of them can be slightly helpful when concatenated with the RGB input (see Table 3). The pose image is less consistently helpful, likely because pose estimation is very challenging on our data and Open-Pose often fails to detect the keypoints especially in the signing hand (see Figure 5). However, treating pose estimation as a secondary task, as is done in our model, successfully "distills" the pose model's knowledge and outperforms the use of additional modalities as input. We note that using all three modalities can boost performance further. Using both RGB and motion images as input while jointly estimating pose, the detector achieves .523/.495/.367 for AP@IoU(0.1/0.3/0.5), improving over the best model in Table 3. However, optimizing performance with multiple modalities is not our main focus, and we leave further study of this direction to future work.

Figure 5: Pose estimation failure cases.

Figure 6: False positive detections. The glosses in (a), (b), (c) are respectively "PEOPLE", "2018", "THAT-[C..}".

6. Conclusion

We study the task of sign language fingerspelling detection, motivated by the goal of developing it as an upstream component in an automatic recognition system. We propose a benchmark, based on extending a previously released fingerspelling recognition dataset, and establish a suite of metrics to evaluate detection models both on their own merits and on their contribution to downstream recognition. We investigate approaches adapted from frame classification, fingerspelling recognition and action detection, and demonstrate that a novel, multi-task model achieves the best results across metrics. Beyond standard detection loss, this model incorporates losses derived from recognition and pose estimation; we show that each of these contributes to the superior performance of the model.

Our results provide, for the first time, a practical recipe for fully automatic detection and recognition of fingerspelling in real-world ASL media. While our focus has been on fingerspelling, we expect that the techniques, both in training and in evaluation of the methods, will be helpful in the more general sign language detection and recognition domain, which remains our goal for future work.

8

References

[1] https://github.com/JJBOY/BMN-Boundary-Matching-Network. 11

[2] https://github.com/yabufarha/ms-tcn. 11

[3] Samuel Albanie, Gul Varol, Liliane Momeni, Triantafyllos Afouras, Joon Son Chung, Neil Fox, and Andrew Zisserman. BSL-1K: Scaling up co-articulated sign language recognition using mouthing cues. In *ECCV*, 2020. 2

[4] Diane Brentari and Carol Padden. A language with multiple origins: Native and foreign vocabulary in American Sign Language. *Foreign vocabulary in sign language: A cross-linguistic investigation of word formation*, pages 87–119, 2001. 1

[5] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. In *CVPR*, 2020. 2

[6] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 5, 6

[7] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster R-CNN architecture for temporal action localization. In *CVPR*, 2018. 2

[8] James Charles, Tomas Pfister, Mark Everingham, and Andrew Zisserman. Automatic and efficient human pose estimation for sign language videos. *Intl. Journal of Computer Vision (IJCV)*, 110:70–90, 2013. 5

[9] Runpeng Cui, Hu Liu, and Changshui Zhang. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *CVPR*, 2017. 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 6

[11] Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*, 2014. 1

[12] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. DAPs: Deep action proposals for action understanding. In *ECCV*, 2016. 2

[13] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019. 7, 11

[14] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *SCIA*, 2003. 8

[15] Jens Forster, Christoph Schmidt, Thomas Hoyoux, Oscar Koller, Uwe Zelle, Justus Piater, and Hermann Ney. RWTH-PHOENIX-Weather: A large vocabulary sign language recognition and translation corpus. In *LREC*, 2012. 2

[16] Jens Forster, Christoph Schmidt, Oscar Koller, Martin Bellgardt, and Hermann Ney. Extensions of the sign language recognition and translation corpus RWTH-PHOENIX-Weather. In *LREC*, 2014. 2

[17] Paul Goh and Eun-Jung Holden. Dynamic fingerspelling recognition using geometric and motion features. In *ICIP*, 2006. 1

[18] Alex Graves, Santiago Fernandez, Faustino Gomez, and Schmidhuber J¨urgen. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006. 4

[19] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 3, 11

[20] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, and Weiping Li. Video-based sign language recognition without temporal segmentation. In *AAAI*, 2018. 2

[21] Haroon Idrees, Amir Zamir, Yu-Gang Jiang, Alexander Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155, 2016. 3

[22] Thomas E Jerde, John F Soechting, and Martha Flanders. Coarticulation in fluent fingerspelling. *Journal of Neuroscience*, 23(6):2383–2393, 2003. 1

[23] Trevor Johnston and Adam C Schembri. On defining lexeme in a signed language. *Sign language & linguistics*, 2(2):115–185, 1999. 1

[24] Hamid Reza Vaezi Joze and Oscar Koller. MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. In *BMVC*, 2019. 2, 5, 8

[25] Jonathan Keane, Diane Brentari, and Jason Riggle. Handshape and coarticulation in ASL fingerspelling. conference presentation, January 2012. Linguistic Society of America 2012 Annual Meeting. 1

[26] Taehwan Kim, Jonathan Keane, Weiran Wang, Hao Tang, Jason Riggle, Gregory Shakhnarovich, Diane Brentari, and Karen Livescu. Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation. *Computer Speech and Language*, pages 209–232, 2017. 1, 2

[27] Oscar Koller, Necati Cihan Camgoz, Hermann Ney, and Richard Bowden. Weakly supervised learning with multistream CNN-LSTM-HMMs to discover sequential parallelism in sign language videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2

[28] Oscar Koller, Jens Forster, and Hermann Ney. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding*, 141:108–125, 2015. 2

[29] Oscar Koller, Hermann Ney, and Richard Bowden. Deep hand: How to train a CNN on 1 million hand images when your data is continuous and weakly labelled. In *CVPR*, 2016. 2

[30] Oscar Koller, Sepehr Zargaran, and Hermann Ney. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs. In *CVPR*, 2017. 2

[31] Oscar Koller, Sepehr Zargaran, Hermann Ney, and Richard Bowden. Deep sign: Hybrid CNN-HMM for continuous sign language recognition. In *BMVC*, 2016. 2

[32] Dongxu Li, Cristian Rodriguez, Xin Yu, and Hongdong Li. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020. 2, 5

[33] Dongxu Li, Xin Yu, Chenchen Xu, Lars Petersson, and Hongdong Li. Transferring cross-domain knowledge for video sign language recognition. In *CVPR*, 2020. 2

[34] Hui Li, Peng Wang, and Chunhua Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *ICCV*, 2017. 2

[35] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. BMN: Boundary-matching network for temporal action proposal generation. In *ICCV*, 2019. 7, 11

[36] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 3

[37] Stephan Liwicki and Mark Everingham. Automatic recognition of fingerspelled words in british sign language. In *2nd IEEE Workshop on CVPR for Human Communicative Behavior Analysis*, 2009. 1

[38] Amit Moryossef, Ioannis Tsochantaridis, Roee Yosef Aharoni, Sarah Ebling, and Srini Narayanan. Real-time sign language detection using human pose estimation. In *ECCV SLRTP workshop*, 2020. 2, 5, 8

[39] Carol A. Padden. The ASL lexicon. *Sign Language and Linguistics*, 1:33–51, 1998. 1

[40] Carol A. Padden and Darline C. Gunsals. How the alphabet came to be used in a sign language. *Sign Language Studies*, 4(1):10–13, 2003. 1

[41] Maria Parelli, Katerina Papadimitriou, Gerasimos Potamianos Georgios Pavlakos, and Petros Maragos. Exploiting 3d hand pose estimation in deep learning-based sign language recognition from rgb videos semantic. In *ECCV SLRTP workshop*, 2020. 5

[42] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time ASL fingerspelling recognition. In *Proceedings of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, jointly with ICCV*, 2011. 1

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 4

[44] Katrin Renz, Nicolaj Stache, Samuel Albanie, and Gül Varol. Sign language segmentation with temporal convolutional networks. In *ICASSP*, 2021. 11

[45] Susanna Ricco and Carlo Tomasi. Fingerspelling recognition through classification of letter-to-letter transitions. In *ACCV*, 2009. 1

[46] Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. Fingerspelling recognition in the wild with iterative visual attention. In *ICCV*, 2019. 1, 2, 5, 6, 11

[47] Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Jonathan Michaux, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. American Sign Language fingerspelling recognition in the wild. In *SLT*, 2018. 2, 6, 11

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[49] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 4

[50] Gabriel Tsechpenakis, Dimitris Metaxas, and Carol Neidle. Learning-based dynamic coupling of discrete and continuous trackers. *Computer Vision and Image Understanding*, 104(2-3):140–156, 2006. 2

[51] Gabriel Tsechpenakis, Dimitris N Metaxas, Carol Neidle, and Olympia Hadjiliadis. Robust online change-point detection in video sequences. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, pages 155–161, 2006. 2

[52] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, pages 229–256, 1992. 5

[53] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3D network for temporal activity detection. In *ICCV*, 2017. 2, 4

[54] Hee-Deok Yang and Seong-Whan Lee. Simultaneous spotting of signs and fingerspellings based on hierarchical conditional random fields and boostmap embeddings. *Pattern Recognition*, 43(8):2858–2870, 2010. 2

[55] Polina Yanovich, Carol Neidle, and Dimitris Metaxas. Detection of major ASL sign types in continuous signing for ASL recognition. In *LREC*, pages 3067–3073, 2016. 2, 8

[56] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017. 2

[57] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. Spatial-temporal multi-cue network for continuous sign language recognition. In *AAAI*, 2020. 2, 3
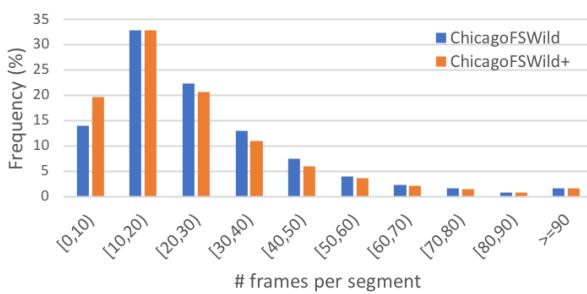
Supplementary Material

A. Statistics of the datasets

Table 4: Numbers of 300-frame raw ASL clips in the ChicagoFSWild and ChicagoFSWild+ data subsets. The number of fingerspelling segments in each subset is given in parentheses.

	train	dev	test
ChicagoFSWild	3539 (6927)	691 (1246)	613 (1102)
ChicagoFSWild+	13011 (44861)	867 (2790)	885 (1531)

Table 4 provides the numbers of clips and of fingerspelling segments in the datasets used in our work. Note that the number of fingerspelling segments is not exactly same as in [46, 47] due to the 75-frame overlap when we split raw video into 300-frame clips. On average there are 1.9/1.8 fingerspelling segments per clip for ChicagoF-SWild/ChicagoFSWild+. The distributions of durations are shown in Figure 7.

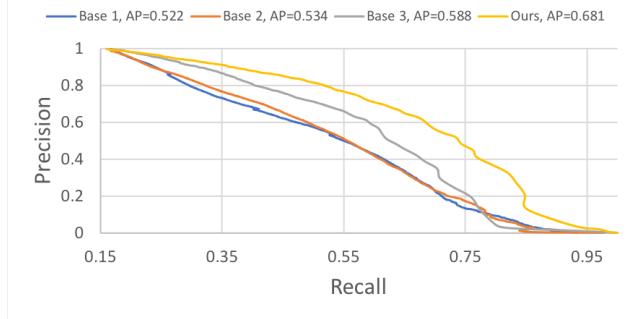Figure 7: Distribution of length of fingerspelling segments.

B. Precision-recall curves for frame classification

Figure 8 shows the precision-recall curves for frame classification of the three baseline models and our proposed model. On the one hand, our approach dominates the others in terms of these frame-level metrics as well. In addition, the differences in frame-level performance among the three baselines are much smaller than the differences in sequence-level performance reported in the main text.

C. Comparison with state-of-the-art models for related tasks

In addition to the baseline models, we compare against two additional approaches—boundary matching network

Figure 8: Precision-recall curves for frame classification of three baselines and our approach.

(BMN) [35] and multi-stage temporal convolutional network (MS-TCN) [13]—on our task of fingerspelling detection. Those two methods are state-of-the-art on temporal action proposal generation in ActivityNet1.3 [19] and sign language segmentation [44]. The implementations are based on [1, 2]. For fair comparison, we use the same backbone network as in the other methods. We use the same network architecture for the individual submodules of the two models and tune hyperparameters on our datasets. As MS-TCN does frame classification in principle, we follow the same post-processing steps as in baseline 1 and 2 to convert frame probabilities into sequence predictions for evaluation.

As is shown in Table 5, these two approaches do not outperform our approach. Comparing BMN and our baseline 3, we notice that the size of the training set has a large impact. The more complex modeling choices in BMN, which searches over a wider range of proposals, leads to better performance mostly when using the larger training set of ChicagoFSWild+. The discrepancy in performance of these two models as measured by different metrics (e.g., AP@IoU vs. AP@Acc) also shows that a model with lower localization error does not always enable more accurate downstream recognition. The MS-TCN model is generally better than other frame-based approaches (baseline 1, 2) but remains inferior to region-based approaches including baseline 3 and ours. Our post-processing steps lead to inconsistency between the training objective and evaluation. Similarly in [44], it is noted that the model sometimes over-segments fingerspelled words.
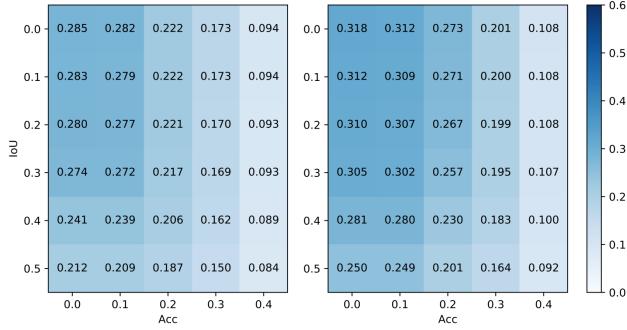
D. Analysis of AP@Acc

Figure 9 shows how varying δ_{IoU} and δ_{acc} impacts the value of AP@Acc. The accuracy threshold δ_{acc} has a much larger impact on AP than does δ_{IoU}. This is primarily because a large overlap between predicted and ground-truth segments is often necessary in order to achieve high accuracy. Therefore, we set the default value of δ_{IoU} to 0.

Table 5: Performance of BMN and MS-TCN on finger-spelling detection using our evaluation metrics on the (a) ChicagoFSWild and (b) ChicagoFSWild+ test sets.

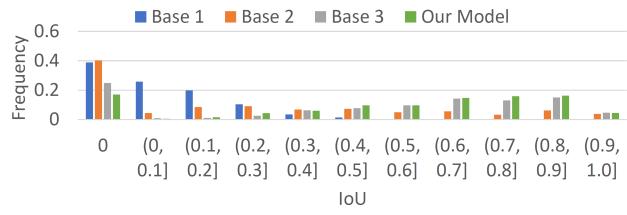|		AP@IoU			AP@Acc			MSA
		AP@0.1	AP@0.3	AP@0.5	AP@0.0	AP@0.2	AP@0.4	
(a)	BMN	.442	.394	.284	.209	.157	.070	.307
	MS-TCN	.282	.177	.095	.141	.093	.036	.319
(b)	BMN	.580	.549	.437	.433	.401	.260	.470
	MS-TCN	.429	.345	.179	.350	.299	.147	.414

Figure 9: AP@Acc with different IoU thresholds on ChicagoFSWild dev set. Left: baseline 3. Right: our model.

E. Histogram of IoU

Figure 10 shows histograms of IoU of predicted segments with respect to the ground truth at peak thresholds used in the MSA computation. Our model has overall higher IoU than the three baselines. The average IoUs of the three baselines and our model for the optimal (peak) threshold δ_f are 0.096, 0.270, 0.485, and 0.524 respectively. The average IoUs of baseline 3 and our model suggest that for AP@IoU, AP@0.5 is more meaningful to compare in terms of recognition performance for those two models.

Figure 10: Histogram of IoU at peak thresholds.

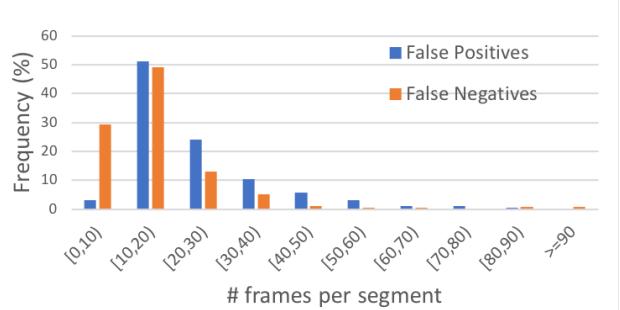## F. Performance breakdown over durations

We separate raw video clips into three categories based on the duration of the fingerspellng segments: short (<20 frames), medium (20-80 frames), and long (≥80 frames). This division is based on the statistics of the dataset. The

performance of our model for the three categories is shown in Table 6, and can be compared to the overall performance in Table 1 of the paper. Shorter fingerspelling segments are harder to spot within regular signing. The typical finger-spelling pattern (relatively static arm and fast finger motion) is less obvious in short segments. In addition, Figure 11 shows the length distribution of false positive and false negative detections from our model. The length distribution of false positives roughly matches that of ground-truth segments in the dataset.

Table 6: Performance on segments of different durations.

	AP@IoU			AP@Acc			MSA
	AP@0.1	AP@0.3	AP@0.5	AP@0.0	AP@0.2	AP@0.4	
Short	.411	.346	.235	.149	.140	.051	.357
Medium	.675	.671	.623	.476	.361	.156	.435
Long	.781	.703	.420	.704	.362	.130	.393

Figure 11: Distribution of lengths of false positives and false negatives.

G. Speed test

The inference time per video clip is shown in Table 7. The speed test is conducted on one Titan X GPU. Inference times for all models are under 1 second. Baselines 1 and 2 are faster as the model architecture is simpler. Our model takes roughly twice the time of baseline 3, which is mainly due to the second-stage refinement.

Table 7: Inference time per 300-frame video clip

	Base 1	Base 2	Base 3	Ours
Inference time (ms)	10.9	11.6	284.5	511.1

H. Detection examples

Figure 12 shows various detection examples from the ChicagoFSWild dev set.

Figure 12: Detection examples. Red: ground-truth segment, green: predicted segment. The sequences are downsampled.