# OpenFS: Multi-Hand-Capable Fingerspelling Recognition with Implicit Signing-Hand Detection and Frame-Wise Letter-Conditioned Synthesis - Supplementary -

Junuk Cha
KAIST

Jihyeon Kim
KT

Han-Mu Park
KETI

In the supplementary material, we provide additional explanations, extended results, and a discussion of limitations and future work, organized as the table of contents below.



## S1. Pose Extraction and Preprocessing

Following PoseNet [1], we employ the MediaPipe Holistic framework [7] to extract multi-hand pose sequences from RGB video frames, where each hand pose is represented by 21 joints in 2D space. We normalize each pose by translating it so that the origin is set to the midpoint between the minimum and maximum coordinates of all joints. The translated coordinates are then divided by the maximum absolute coordinate value and multiplied by 0.5, resulting in values scaled to the range $[-0.5, 0.5]$. We construct the input multi-hand pose sequence by concatenating the normalized pose sequences of multiple hands along the temporal dimension.

In multi-person scenarios, since MediaPipe [7] does not support multi-person hand pose extraction, we employ YOLOv11 [4] to detect all individuals appearing in a video. For frames containing multiple people, each detected person is cropped and processed independently using MediaPipe to extract hand poses. The extracted hand poses are then normalized and concatenated along the temporal dimension following the same procedure described above.

In addition, MediaPipe [7] provides left–right hand identity, while YOLOv11 [4] provides person identity; we combine these two signals into a unified hand-identity encoding, which is used in both our dual-level positional encoding and the Signing-Hand Focus loss. This unified hand-identity encoding enables our multi-hand-capable recognizer to robustly process hand poses from multiple hands and multiple people simultaneously.

## S2. Loss Function Details

**Signing-hand focus loss for $N$ hands.** While the main paper explains the signing-hand focus loss using the two-hand case (right and left hands) for clarity, the formulation naturally extends to an arbitrary number of hands. In the general setting, the decoder cross-attention tensor is defined as: $\mathbf{A} \in \mathbb{R}^{L_d \times |W| \times T}$, where $L_d$ is the number of decoder layers, $W = \{W_i\}_{i=1}^{|W|}$ is the output letter-token sequence of length $|W|$, and $T = \sum_{n=1}^{N} T_n$ is the total number of pose tokens, with $T_n$ denoting the number of pose tokens for the $n$-th hand and $N$ denoting the total number of hands. The layer-averaged cross-attention is computed as:

$$\widetilde{\mathbf{A}}_{W_i,t} = \frac{1}{L_d} \sum_{\ell=1}^{L_d} \mathbf{A}_{\ell,W_i,t}, \qquad \widetilde{\mathbf{A}} \in \mathbb{R}^{|W| \times T}. \tag{S1}$$

Each pose token is associated with a hand-identity label $h \in \{1, ..., N\}$, and these labels form a one-hot matrix:

$$\mathbf{H} \in \mathbb{R}^{T \times N}, \qquad H_{t,h} = \begin{cases} 1 & \text{if the pose token at position } t \\ & \text{belongs to hand } h, \\ 0 & \text{otherwise.} \end{cases} \tag{S2}$$

The attention contribution from hand $h$ to letter token $W_i$ is computed as:

$$a_{W_i,h} = \sum_{t=1}^{T} \widetilde{A}_{W_i,t} H_{t,h}, \qquad \sum_{h=1}^{N} a_{W_i,h} = 1. \tag{S3}$$

To encourage each letter token $W_i$ to place its attention on a single signing hand, the entropy of the hand-attention distribution is minimized:

$$\mathcal{E}_{W_i,h} = -a_{W_i,h} \log(a_{W_i,h} + \epsilon). \tag{S4}$$

The signing-hand focus loss is defined as:

$$\mathcal{L}_{\text{SF}} = \frac{1}{|W|} \frac{1}{N} \sum_{i=1}^{|W|} \sum_{h=1}^{N} \mathcal{E}_{W_i,h}. \tag{S5}$$

**Monotonic alignment loss.** To enforce a monotonic correspondence between the pose-token sequence and the letter sequence, the cross-attention tensor $\mathbf{A} \in \mathbb{R}^{L_d \times |W| \times T}$ is accumulated along the temporal dimension. For each letter token $W_i$, the cumulative attention is computed as:

$$\mathbf{C}_{\ell,W_i,t} \quad = \quad \sum_{t'=1}^{t} \mathbf{A}_{\ell,W_i,t'}, \tag{S6}$$

The temporal change between $W_{i-1}$ and $W_i$ (for $i \geq 2$) is:

$$\Delta_{\ell,W_i,t} \quad = \quad \mathbf{C}_{\ell,W_i,t} - \mathbf{C}_{\ell,W_{i-1},t}. \tag{S7}$$

A monotonicity violation occurs when $W_i$ places more attention on earlier pose-token positions than $W_{i-1}$. Such violations are defined as:

$$\mathbf{V}_{\ell,W_i,t} \quad = \quad \max(\Delta_{\ell,W_i,t}, 0). \tag{S8}$$

The final monotonic alignment loss is:

$$\mathcal{L}_{\text{MA}} \quad = \quad \frac{1}{(|W|-1)} \frac{1}{T} \sum_{\ell=1}^{L_d} \sum_{i=2}^{|W|} \sum_{t=1}^{T} \mathbf{V}_{\ell,W_i,t}. \tag{S9}$$

### S2.1. Ablation Study on SF and MA Losses

To analyze the individual contributions of the signing-hand focus (SF) loss and the monotonic alignment (MA) loss, we conduct ablation experiments by selectively removing each component from the full objective. Without either auxiliary loss, the baseline model achieves 74.8 letter accuracy. Applying only the SF loss or only the MA loss results in 74.7 letter accuracy in both cases. When both losses are jointly applied, performance improves to 75.4 letter accuracy. These results indicate that the SF and MA losses are complementary and are most effective when applied together.

## S3. Coarse Frame-Wise Letter Annotation

To obtain frame-wise letter annotations, we leverage the recognizer's decoder cross-attention, which encodes the correspondence between output letter tokens and input pose frames. For each decoder layer $\ell \in \{1,...,L_d\}$, let $\mathbf{A}_\ell \in \mathbb{R}^{|W| \times T}$ denote the cross-attention matrix, where $|W|$ is the number of output letter tokens and $T$ is the number of input frames. We first compute the layer-averaged attention:

$$\widetilde{\mathbf{A}}_{W_i,t} = \frac{1}{L_d} \sum_{\ell=1}^{L_d} \mathbf{A}_{\ell,W_i,t}, \qquad \widetilde{\mathbf{A}} \in \mathbb{R}^{|W| \times T}. \tag{S10}$$

Each row $\widetilde{\mathbf{A}}_{W_i,:}$ represents how strongly $i$-th letter token $W_i$ attends to each frame.

**Token-wise thresholding.** For each token $W_i$, let $\mathbf{a}_i = \widetilde{\mathbf{A}}_{W_i,:} \in \mathbb{R}^T$. We exclude the highest attention value because it frequently shows an overly large peak, making it an unreliable indicator of the true attention distribution. Using the 2nd–4th largest values instead provides a more stable estimate of the typical attention magnitude, and thus yields a more robust threshold:

$$\theta_i \quad = \quad 0.5 \cdot \text{mean}(\text{top-2–4}(\mathbf{a}_i)). \tag{S11}$$

Frames whose attention exceeds this threshold are assigned to token $W_i$:

$$L_{i,t} = \begin{cases} W_i, & \text{if } a_{i,t} \geq \theta_i, \\ -1, & \text{otherwise.} \end{cases} \tag{S12}$$

This yields a temporary label matrix

$$\mathbf{L} \in \mathbb{R}^{|W| \times T}. \tag{S13}$$

An example of this matrix is shown in Fig. 4(a) of the main paper, visualized as a processed cross-attention map.

**Frame-level label consolidation.** Final frame labels are obtained by collapsing $\mathbf{L}$ along the letter-token dimension. For each frame $t$, let

$$U_t = \{L_{i,t} \mid L_{i,t} \neq -1\}. \tag{S14}$$

If $U_t$ contains exactly one unique letter, the frame receives that label; otherwise (when no letter is assigned or when the frame is matched by multiple letter tokens), it is assigned the blank symbol $\phi$:

$$y_t = \begin{cases} u, & \text{if } U_t = \{u\}, \\ \phi, & \text{otherwise.} \end{cases} \tag{S15}$$

The resulting frame-wise label sequence is

$$\mathbf{y} = (y_1,...,y_T), \qquad \mathbf{y} \in \{\phi\} \cup \{1,...,\text{char\_size}\}^T, \tag{S16}$$

which constitutes the coarse frame-wise letter annotation used for training the frame-wise annotation refiner. An example of this label sequence $\mathbf{y}$ is shown in Fig. 4(a) of the main paper, visualized as coarse frame-wise letter labels.

## S4. Implementation Details

All experiments are implemented in PyTorch [9] and conducted on a single NVIDIA A40 GPU. Transformer [16] hyperparameters, training configurations, and the embedding/output head architectures of the recognizer are summarized in Tabs. S1 and S2, while those of the generator are provided in Tabs. S3 and S4. The architecture of the frame-wise annotation refiner are described in Tab. S5. Following PoseNet [1], the character set includes 26 lowercase English letters, several special symbols such as the space character, and two additional tokens, `<start>` and `<end>`, resulting in 33 characters in total.

Table S1. Implementation details of the Transformer and training configurations for the recognizer.

| Component | Recognizer |
|---|---|
| Architecture | Transformer encoder–decoder |
| Layers | $L_{\text{enc}}/L_{\text{dec}}=3/3$ |
| Hidden dimension | 256 |
| Feed-forward dimension | 2,048 |
| Attention heads | 8 |
| Activation | GELU |
| Dropout | 0.1 |
| Positional encoding | Dual-level (hand+time) |
| Optimizer | Adam [5] |
| Learning rate | 1e−4 |
| Epochs | 20 |
| LR schedule | decayed by 0.1 every 10 epochs |
| Batch size | 64 |

Table S2. Architectures of the encoder-side pose embedding, decoder-side character embedding, and decoder head used in the **recognizer**. `pose_dim` corresponds to 21 (joints) × 2 $(x, y)$ = 42. We set `char_size` = 33, and the additional index is reserved for padding.

| Layer | Input Dimension | Output Dimension |
|---|---|---|
| **Encoder: Pose embedding** | | |
| Linear | `pose_dim` | 256 |
| LayerNorm | 256 | 256 |
| ReLU | 256 | 256 |
| Linear | 256 | 256 |
| LayerNorm | 256 | 256 |
| ReLU | 256 | 256 |
| **Decoder: Character embedding** | | |
| Embedding | `char_size`+1 | 256 |
| **Decoder: Output head** | | |
| Linear | 256 | 128 |
| ReLU | 128 | 128 |
| Linear | 128 | 33 |

Table S3. Implementation details of the Transformer architecture and training configurations for the generator, and the diffusion process.

| Component | Generator (DDIM) |
|---|---|
| Architecture | Transformer encoder |
| Layers | $L=8$ |
| Hidden dimension | 256 |
| Feed-forward dimension | 1,024 |
| Attention heads | 4 |
| Activation | GELU |
| Dropout | 0.1 |
| Positional encoding | Temporal |
| Optimizer | Adam [5] |
| Learning rate | 1e−4 |
| Epochs | 1,000 |
| Batch size | 20 |
| Diffusion method | DDIM [13] |
| Noise schedule | Cosine |
| Diffusion steps | $T=50$ |

Table S4. Architectures of the embedding layers and output head used in the **generator**. We use `char_size` = 33, and the additional index is reserved for padding. `pose_dim` corresponds to 21 (joints)×3 $(x,y,z)$=63.

| Layer | Input Dimension | Output Dimension |
|---|---|---|
| **Time embedding** | | |
| Sinusoidal | 1 | 256 |
| Linear | 256 | 256 |
| SiLU | 256 | 256 |
| Linear | 256 | 256 |
| **Pose embedding** | | |
| Linear | `pose_dim` | 128 |
| **Letter embedding** | | |
| Embedding | `char_size`+1 | 128 |
| **Output head** | | |
| Linear | 256 | `pose_dim` |

Table S5. Architecture of the frame-wise annotation refiner. We set `char_size` = 33.

| Layer | Input Dimension | Output Dimension |
|---|---|---|
| Linear | 256 | 256 |
| LayerNorm | 256 | 256 |
| ReLU | 256 | 256 |
| Dropout ($p=0.1$) | 256 | 256 |
| Linear | 256 | `char_size` |

Table S6. Ablation on input pose representations on CFSW [11], comparing coordinate dimension (2D vs 3D) and concatenation strategy (frame-wise vs joint-wise).

| Coordinate Dim. | Concatenation | Letter Accuracy |
|---|---|---|
| 3D | frame-wise | 73.4 |
| 2D | joint-wise | 71.6 |
| 2D | frame-wise | **75.4** |

Table S7. FID and Diversity results across different conditioning methods. WC denotes word-conditioned generation, LC denotes letter-conditioned generation, and FWLC denotes frame-wise letter-conditioned generation. Lower FID indicates better fidelity, and Diversity results are better when the values are closer to the real data. All evaluations are repeated 20 times, and $\pm$ indicates the 95% confidence interval.

| Method | FID (↓) | Diversity (→) |
|---|---|---|
| Real | - | 1.1107 ± 0.0115 |
| WC | 0.4036 ± 0.0408 | 0.7803 ± 0.0015 |
| LC | 0.4159 ± 0.0516 | 0.8987 ± 0.0009 |
| FWLC | **0.3695 ± 0.0522** | **0.8996 ± 0.0006** |

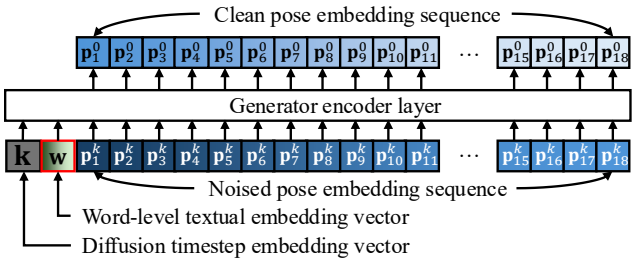
## S5. Comparison of Input Pose Representations

In Tab. S6, we first compare 3D and 2D input pose representations. Although Mediapipe [7] provides 3D hand poses, they are substantially noisier than their 2D estimates, and this noise leads to noticeably lower recognition accuracy. A similar observation was also reported in PoseNet [1]. As a result, the 2D representation yields the best performance.

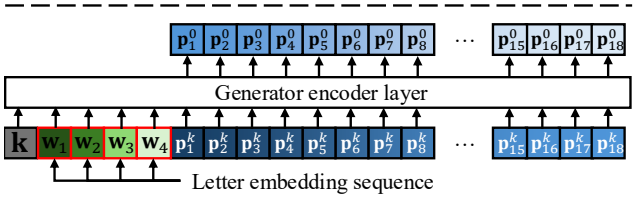
We also evaluate an alternative representation that does not treat the multi-hand pose sequences as a frame-wise concatenation but instead concatenates all joint coordinates within each frame into a single vector (joint-wise concatenation). In our default setting, a pose sequence has the shape $T \times N \times J \times d$, where $T$ is the number of frames, $N$ is the number of hands, $J$ is the number of joints per hand, and $d$ is the coordinate dimension. The *frame-wise* representation preserves the per-hand spatial structure, and the model effectively receives inputs of the form $(T \times N) \times J \times d$. In contrast, the *joint-wise* representation collapses the $N$ hands within each frame, resulting in $T \times (NJ) \times d$, which merges informative and irrelevant hands into the same joint axis, providing a less discriminative input representation and ultimately degrading recognition performance.

## S6. Comparison of Conditioning Strategies for Generator

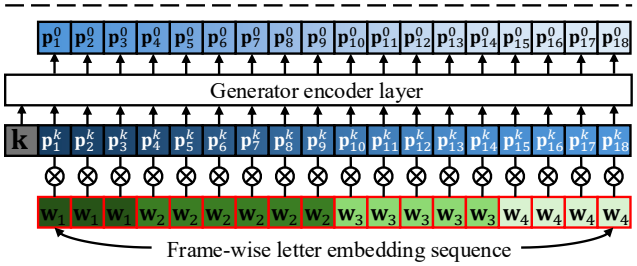
Fig. S1 shows a design comparison of the three conditioning strategies for the generator: word-conditioned (WC), letter-conditioned (LC), and frame-wise letter-conditioned (FWLC). For the WC setting, we adopt the CLIP word-level text embedding [10] as the conditioning input following previous




(a) Word-conditioned (WC)




(b) Letter-conditioned (LC)




(c) Frame-wise letter-conditioned (FWLC)

Figure S1. **Comparison of the three conditioning strategies.** We omit the details of standard positional encoding [16] and the embedding layers. For illustration, we consider a word of length 4. The symbol $\mathbf{w}$ denotes the word-level textual embedding, $\mathbf{w}_i$ denotes the embedding of the $i$-th letter in the word, and $\mathbf{p}_t^k$ denotes a pose embedding token, where $t$ is the frame index and $k$ is the diffusion timestep ($k = 0$ indicates a clean pose). The diffusion timestep embedding vector is represented as $\mathbf{k}$. (a) WC uses a single word-level textual embedding token as input. (b) LC takes a sequence of letter tokens, with one token corresponding to each character. (c) FWLC aligns each pose embedding token with its corresponding letter embedding token and concatenates them in a one-to-one, frame-wise manner. The different conditioning inputs are highlighted with red rectangles.

text-to-motion methods [3, 6, 8, 14, 15, 18]. In contrast, LC and FWLC do not rely on CLIP, instead, each letter is represented by a learnable letter embedding layer that maps character indices to continuous embedding vectors. This difference causes WC to encode word-level semantics, while LC and FWLC provide letter-specific conditioning signals, which are more suitable for fingerspelling pose generation since fingerspelled hand poses are inherently defined at the letter level.

In Tab. S7, we further measure the FID and Diversity of samples produced under different conditioning strategies. Among these strategies, the frame-wise letter-conditioned (FWLC) generator delivers the best overall performance. The word-conditioned (WC) baseline conditions generation solely on a single word-level embedding, ignoring the structure of

Table S8. Model size, number of parameters, and inference speed (FPS) of PoseNet [1] and our models, grouped into the multi-hand-capable (MHC) recognizer and the Frame-Wise Letter-Conditioned (FWLC) generator.

| Method | Model Size (MB) | Params. (M) | FPS |
|---|---|---|---|
| PoseNet [1] | 33.5 | 8.78 | 6 |
| MHC recognizer | 33.7 | 8.81 | 962 |
| FWLC generator | 24.8 | 6.48 | 96 |

Table S9. Comparison of deletion, substitution, and insertion error counts and their corresponding rates across different methods on CFSW [11]. The symbol † denotes methods trained with additional synthetic data.

| Method | Deletions | Substitutions | Insertions |
|---|---|---|---|
| PoseNet [1] | 962(21.8%) | 502(11.4%) | **235(5.3%)** |
| Ours | **458(10.4%)** | **390(8.8%)** | 239(5.4%) |
| PoseNet† [1] | 850(19.2%) | **339(7.7%)** | **173(3.9%)** |
| Ours† | **336(7.6%)** | 344(7.8%) | 304(6.9%) |

individual letters, while the letter-conditioned (LC) baseline provides letter-level cues only at coarse segment boundaries without enforcing frame-wise temporal alignment. In contrast, FWLC conditions the generator on the corresponding letter at every frame, preserving both temporal alignment and fine-grained letter-level structure. This dense conditioning leads FWLC to generate synthetic sequences with higher fidelity and more realistic diversity, achieving the strongest results across all metrics. Additional qualitative comparisons (Fig. 7), along with recognizer accuracy evaluations (Tab. 5) on synthetic data generated under each conditioning strategy, are provided in the main paper.

## S7. Model Efficiency

In Table S8, we report the model size, parameter counts, and inference speed (FPS) of our multi-hand-capable (MHC) recognizer and Frame-Wise Letter-Conditioned (FWLC) generator, compared against PoseNet [1]. Although the MHC recognizer has a similar number of parameters to PoseNet (8.81M vs. 8.78M), it achieves a higher inference speed (962 FPS vs. 6 FPS), demonstrating that our architecture is substantially more efficient while supporting multi-hand inputs. The FWLC generator is even more lightweight, with 6.48M parameters and a model size of only 24.7 MB, and operates at 96 FPS, enabling fast pose-sequence synthesis.

## S8. Stabilized Error-Type Sensitivity

In Tab. S9, Ours† (the symbol † denotes methods trained with additional synthetic data) shows superior performance in the case of *Deletions* and comparable results in *Substitutions* and *Insertions*, while maintaining balanced error rates across all error types. Compared to Ours, Ours† exhibits a slight increase in insertion errors, but achieves a substantial reduction in deletion errors and a modest reduction in substitution errors. Notably, PoseNet and PoseNet† [1] exhibits an extreme imbalance between deletion and insertion errors; deletions occur 4–5 times more frequently than insertions. This indicates that the model has a strongly conservative decoding behavior, often choosing not to output a letter under uncertainty.

Table S10. Letter accuracy across word-length bins (1–4 and 5+) on CFSW [11]. Diff. denotes the accuracy drop between short (1–4) and long (5+) words. Both PoseNet [1] and our model show accuracy drops on longer words, while their synthetic-data variants† markedly reduce this gap.

| Method | 1-4 | 5+ | Diff. |
|---|---|---|---|
| PoseNet [1] | 65.6 | 59.4 | 6.2 |
| Ours | **79.2** | **73.4** | **5.8** |
| PoseNet† [1] | 69.2 | 68.8 | **0.4** |
| Ours† | **80.0** | **76.6** | 3.4 |

Table S11. We compare PoseNet [1], Ours, and their variants† trained with additional synthetic data on CFSW [11], CFSWP [12] and FSNeo, using top-1 accuracy as the evaluation metric. Parenthesized numbers indicate the accuracy improvements achieved by using the additional synthetic data†.

| Method | CFSW | CFSWP | FSNeo |
|---|---|---|---|
| PoseNet [1] | 28.8 | 25.7 | 4.7 |
| Ours | **44.6** | **40.4** | **24.1** |
| PoseNet† [1] | 36.9(**+8.1**) | 36.0(**+10.3**) | 64.2(**+59.5**) |
| Ours† | **50.5**(+5.9) | **48.5**(+8.1) | **82.0**(+57.9) |

## S9. Improving Robustness to Long Words

As shown in Tab. S10, both recognizers exhibit clear performance degradation as word length increases when trained only on the original dataset (PoseNet [1]: 65.6→59.4; Ours: 79.2→73.4). However, incorporating the synthetic data generated by our frame-wise letter-conditioned generator substantially reduces this gap. For PoseNet† [1], the short-to-long difference shrinks from 6.2 to 0.4, effectively eliminating the length-related drop. Similarly, Ours† reduces the gap from 5.8 to 3.4, demonstrating improved robustness on longer words.

## S10. Evaluation on Additional Metric

As shown in Tab. S11, we utilize top-1 accuracy as an additional evaluation metric to complement the letter-accuracy results reported in the main paper. Letter accuracy evaluates character-level correctness by accounting for substitution, deletion, and insertion errors, whereas top-1 accuracy is a word-level metric that counts a prediction as correct only when the entire word is recognized exactly. Under this more challenging metric, our recognizer achieves significantly higher performance than PoseNet [1] on all datasets. Furthermore, the variants trained with our additional synthetic data†, generated by the frame-wise letter-conditioned (FWLC) generator, show substantial improvements in top-1 accuracy. This indicates that

Table S12. On the FSboard dataset [2], our multi-hand-capable recognizer substantially outperforms ByT5-based methods [2, 17] while using 34× fewer parameters. ByT5-s and ByT5-p denote the ByT5 model trained from scratch and the pretrained variant, respectively.

| Method | Letter Acc. | Top-1 Acc. | Params. (M) |
|---|---|---|---|
| ByT5-s [2, 17] | 66.2 | 17.9 | 300M |
| ByT5-p [2, 17] | 88.9 | 52.9 | 300M |
| Ours | **93.7** | **59.4** | 8.81M |

FWLC produces synthetic sequences that are not only realistic at the frame level but also highly effective for enhancing overall word-level recognition performance.

## S11. Evaluation on FSboard

We evaluate our multi-hand-capable recognizer on FSboard [2], a large-scale ASL fingerspelling dataset collected from smartphone recordings. FSboard provides 151K sequences (train: 126K, validation: 12K, test: 13K) totaling 3.2M characters across 147 signers. Unlike existing ASL fingerspelling datasets, FSboard includes diverse content categories such as addresses, URLs, names, and phone numbers, and contains not only alphabetic fingerspelling but also digits and special characters, offering a significantly broader label space.

As shown in Tab. S12, the ByT5 [17]-based method trained on FSboard [2] from scratch (ByT5-s) achieves 66.2 letter accuracy and 17.9 top-1 accuracy, while the pretrained variant (ByT5-p) achieves 88.9 letter accuracy and 52.9 top-1 accuracy. In contrast, our recognizer achieves 93.7 letter accuracy and 59.4 top-1 accuracy, while using 34 times fewer parameters (8.81M vs 300M), demonstrating substantially higher efficiency and effectiveness. Notably, FSboard contains not only alphabetic fingerspelling but also digits and special characters. Some of these symbols share similar hand poses across different character domains (e.g., letters vs. numbers), which introduces additional ambiguity. Despite this challenge, our model remains robust to such cross-domain pose similarity and consistently outperforms the ByT5-based methods. Moreover, while the ByT5-pretrained method benefits from large-scale pretraining on massive text data, our recognizer is trained without such pretraining and still achieves superior accuracy with significantly fewer parameters, highlighting the effectiveness of our architecture.

## S12. More Qualitative Results

In Figs. S4 to S6, we present the qualitative recognition results of PoseNet [1], PoseNet†, Ours, and Ours† on ChicagoFSWild [11]. The symbol † denotes models trained with additional synthetic data generated by our frame-wise letter-conditioned generator. We also report the letter accuracy for each prediction. Letter accuracy is the standard metric defined as the proportion of correctly predicted letters after accounting for substitutions, deletions, and insertions.

In addition, Fig. S7 illustrates qualitative examples of our coarse-to-fine frame-wise letter annotation process, highlighting the alignment between refined labels and human annotations.

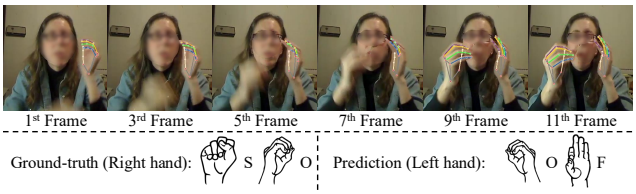



Figure S2. **The only failure case of implicit signing-hand detection.** Although the signer uses the right hand for fingerspelling, the short video and motion blur in early frames cause the model to incorrectly identify the left hand as the signing hand. The ground-truth is "so" with the right hand, whereas the prediction is "of" with the left hand.

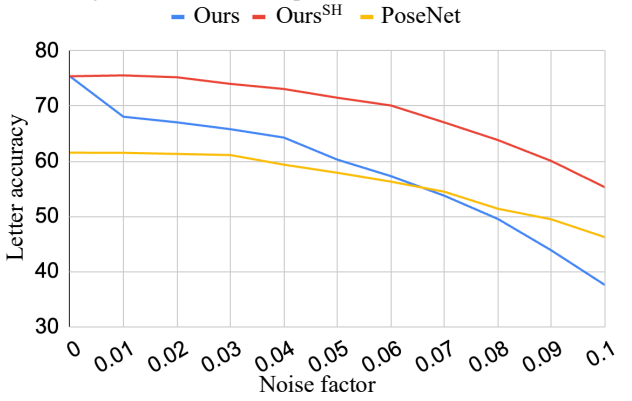



Figure S3. Robustness to pose noise. We increment the noise factor by 0.01 at each step. Ours$^{SH}$ refers to the variant that first identifies the signing hand via cross-attention and then performs recognition using only the selected hand. This single-hand variant maintains substantially higher robustness than both our full model (Ours) and PoseNet across all noise factors.

## S13. Failure Case of Implicit Signing-Hand Detection

While our implicit signing-hand detection module performs reliably in most cases, we identify a single failure case in the test set. In this example, the signing-hand pose appears only in the last four frames out of twelve after the motion blur subsides, as shown in Fig. S2. Under such severe blur, even humans may struggle to correctly determine the signing hand.

## S14. Sensitivity Analysis under Pose Noise

Fig. S3 shows the robustness comparison under pose noise among PoseNet [1], Ours, and Ours$^{SH}$. Ours$^{SH}$ denotes the variant of our model that first identifies the signing hand via cross-attention and then performs recognition using only the selected hand. We add Gaussian noise to the original pose $P$ as $\tilde{P} = P + \sigma_{\text{noise}}\mathcal{N}(0,I)$, where the noise factor $\sigma_{\text{noise}}$ increases from 0 to 0.10 in increments of 0.01 (each step corresponding to 1% of the pose value range $[-0.5, 0.5]$).

Ours exhibits clear vulnerability to pose noise. Its performance drops sharply even at $\sigma_{\text{noise}} = 0.01$, and it falls below

PoseNet starting from $\sigma_{\text{noise}} = 0.07$. This degradation occurs because the non-signing hand introduces additional noise that interferes with correct recognition. To validate this hypothesis, we evaluate a single-hand variant. Ours$^{\text{SH}}$ shows substantially better robustness to pose noise than both Ours and PoseNet across all noise levels.

## S15. Limitations and Future Work

The publicly released dataset exposes multiple challenges: annotation-pose mismatch and articulation-induced label distortion.

**Annotation–Pose Mismatch.** As shown in Fig. S2, the ground-truth label for this sample is "so", but the actual hand configuration resembles "o". This type of mismatch arises from manual annotation ambiguity and highlights the difficulty of accurate labeling when frames are noisy or blurred. Such mismatches degrade recognizer performance when present in the training data, and when they appear in the test set, they impose an upper bound on achievable accuracy.

**Articulation-Induced Label Distortion.** The dataset also contains cases where the signer does not fully articulate certain letters. The annotations faithfully follow the visible evidence, yet the resulting labels may form non-existent words. For instance, "ASL Companion Volume" becomes "ASL Companion Volme" because the signer omits the letter "u" when producing "volume", as illustrated in Fig. S6. In this case, the phenomenon itself does not directly harm recognizer performance. However, depending on the annotator's judgment, such example may be labeled as "volume" despite the missing articulation, thereby risking a transition into an annotation–pose mismatch.

Furthermore, because our method relies on MediaPipe [7] for hand-pose extraction, the resulting pose sequences inherit estimator-induced uncertainty, which may introduce additional noise for downstream models, as shown in Fig. S3. This dependency also makes the system vulnerable to jittering and motion blur, especially in low-quality or fast-moving video.

As future work, we aim to develop methods that leverage context beyond the fingerspelling clip itself. Certain failure cases such as motion blur, occlusions, incomplete articulations, and pose estimator noise remain challenging to handle within the scope of fingerspelling only. This motivates the development of a broader sign language understanding framework that incorporates surrounding signs and semantic context. By integrating fingerspelling with standard sign language, the model can reason over the full communicative context and produce more robust and reliable predictions under challenging visual conditions.

## References


[1] Pooya Fayyazsanavi, Negar Nejatishahidin, and Jana Košecká. Fingerspelling posenet: Enhancing fingerspelling translation with pose-based transformer models. In *WACV*, 2024. 1, 3, 4, 5, 6

[2] Manfred Georg, Garrett Tanzer, Esha Uboweja, Saad Hassan, Maximus Shengelia, Sam Sepah, Sean Forbes, and Thad Starner. Fsboard: Over 3 million characters of asl fingerspelling collected via smartphones. In *CVPR*, 2025. 6

[3] Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. Momask: Generative masked modeling of 3d human motions. In *CVPR*, 2024. 4

[4] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024. 1

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[6] Junfan Lin, Jianlong Chang, Lingbo Liu, Guanbin Li, Liang Lin, Qi Tian, and Chang-wen Chen. Being comes from not-being: Open-vocabulary text-to-motion generation with wordless training. In *CVPR*, 2023. 4

[7] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 1, 4, 7

[8] Zichong Meng, Yiming Xie, Xiaogang Peng, Zeyu Han, and Huaizu Jiang. Rethinking diffusion for text-driven human motion generation: Redundant representations, evaluation, and masked autoregression. In *CVPR*, 2025. 4

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NIPS*, 2019. 3

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 4

[11] 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 *IEEE Spoken Language Technology Workshop (SLT)*, 2018. 4, 5, 6, 8, 9, 10

[12] 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. 5

[13] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3

[14] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *ECCV*, 2022. 4

[15] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *ICLR*, 2023. 4

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS*, 2017. 3, 4

[17] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 2022. 6

[18] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE TPAMI*, 2024. 4

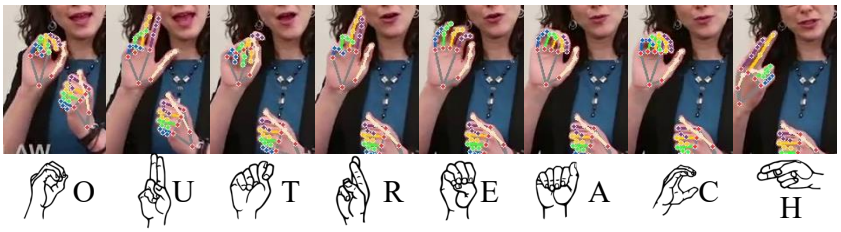
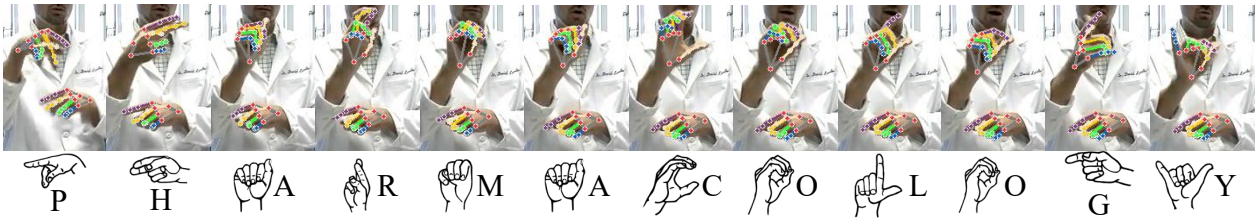




Figure S4. Qualitative recognition results on ChicagoFSWild [11]. For each example, we show the input frames, the ground-truth letters, and the predictions from PoseNet, PoseNet†, Ours, and Ours†. The symbol † denotes models trained with additional synthetic data generated by our frame-wise letter-conditioned generator. We also report the letter accuracy for each prediction. Colored blocks indicate different types of prediction outcomes: blue for ground-truth letters, green for correct predictions, purple for substitution errors, red for deletion errors, and yellow for insertion errors.




Ground-truth: p h a r m a c o l o g y
pharmacology

PoseNet: d p a r m a c a i n
dparmacain (Letter accuracy: 41.7%)

PoseNet†: d h a r m a c o g y n
dharmacogyn (Letter accuracy: 66.7%)

Ours: p h a r m a c o g y n t
pharmacogynt (Letter accuracy: 66.7%)

Ours†: p h a r m a c o g y
pharmacogy (Letter accuracy: **83.3%**)

: Ground-truth letter
: Correct pred
: Substitution error
: Deletion error
: Insertion error


Ground-truth: j e f f ^ d a n i e l s
jeff^daniels

PoseNet: j ^ f ^ f ^ d e i l l e s n
j^f^f^deillesn (Letter accuracy: 41.7%)

PoseNet†: j a f f ^ d a n i l e s
jaff^daniles (Letter accuracy: 75.0%)

Ours: j a f ^ f d a n i e l s
jaf^fdaniels (Letter accuracy: 75.0%)

Ours†: j a f f ^ d a n i e l s
jaff^daniels (Letter accuracy: **91.7%**)

Figure S5. Qualitative recognition results on ChicagoFSWild [11]. For each example, we show the input frames, the ground-truth letters, and the predictions from PoseNet, PoseNet†, Ours, and Ours†. The symbol † denotes models trained with additional synthetic data generated by our frame-wise letter-conditioned generator. We also report the letter accuracy for each prediction. Colored blocks indicate different types of prediction outcomes: blue for ground-truth letters, green for correct predictions, purple for substitution errors, red for deletion errors, and yellow for insertion errors. The symbol ^ denotes a space character.


Figure S6. Qualitative recognition results on ChicagoFSWild [11]. For each example, we show the input frames, the ground-truth letters, and the predictions from PoseNet, PoseNet†, Ours, and Ours†. The symbol † denotes models trained with additional synthetic data generated by our frame-wise letter-conditioned generator. We also report the letter accuracy for each prediction. Colored blocks indicate different types of prediction outcomes: blue for ground-truth letters, green for correct predictions, purple for substitution errors, red for deletion errors, and yellow for insertion errors. The symbol ^ denotes a space character. For the first example, earlier input frames are omitted for space.

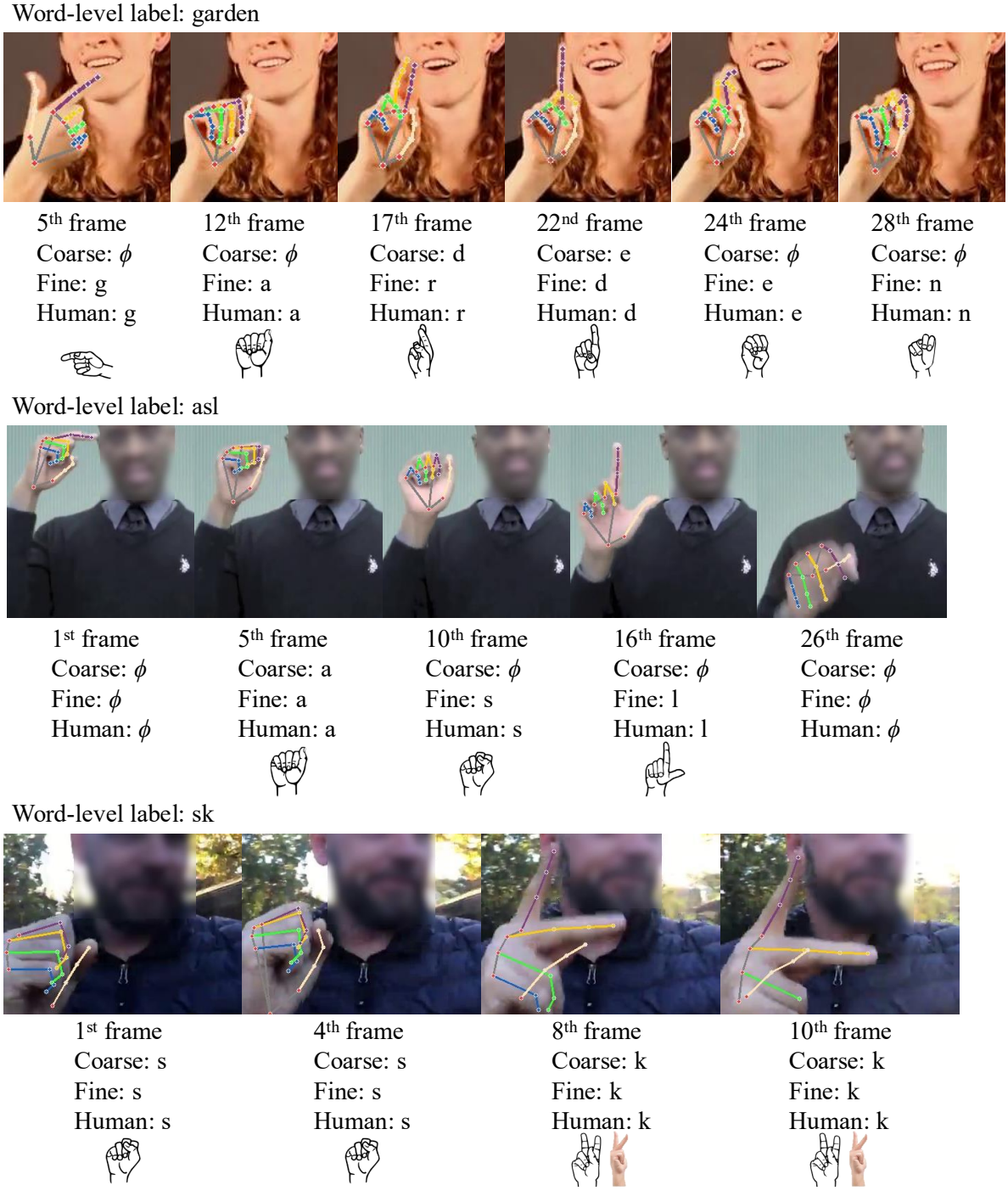



Figure S7. Qualitative examples of coarse-to-fine frame-wise letter annotation. For each word-level label (top: garden, middle: asl, bottom: sk), selected frames are shown with extracted hand keypoints. We compare coarse pseudo labels, fine-grained refined labels, and human annotations. The symbol $\phi$ denotes the absence of a letter label for the corresponding frame. The results show that the refined fine labels align more closely with human annotations, especially in frames where the coarse labels fail to assign a valid letter.