

HandReader: Advanced Techniques for Efficient Fingerspelling Recognition

Pavel Korotaev¹
pkorotaevdm@gmail.com

Petr Surovtsev¹
pysurovtsev@daimld.tech

Alexander Kapitanov¹
aakapitanov@daimld.tech

Karina Kvanchiani¹
karinakvanchiani@gmail.com

Alexander Nagaev¹
aonagaev@daimld.tech

¹Department of Artificial Intelligence and Machine Learning Development, DAIMLD

Abstract

Fingerspelling is a significant component of Sign Language (SL), allowing the interpretation of proper names, characterized by fast hand movements during signing. Although previous works on fingerspelling recognition have focused on processing the temporal dimension of videos, there remains room for improving the accuracy of these approaches. This paper introduces HandReader, a group of three architectures designed to address the fingerspelling recognition task. HandReader_{RGB} employs the novel Temporal Shift-Adaptive Module (TSAM) to process RGB features from videos of varying lengths while preserving important sequential information. HandReader_{KP} is built on the proposed Temporal Pose Encoder (TPE) operating on keypoints as tensors. Such keypoint composition in a batch allows the encoder to pass them through 2D and 3D convolution layers, utilizing temporal and spatial information and accumulating keypoint coordinates. We also introduce HandReader_{RGB+KP} – architecture with a joint encoder to benefit from RGB and keypoint modalities. Each HandReader model possesses distinct advantages: on the ChicagoFSWild and ChicagoFSWild+ datasets, HandReader models achieve the best results among methods trained on a single training set, including a new state-of-the-art result on ChicagoFSWild+.

1 Introduction

Sign Language (SL) facilitates communication among more than 430 million deaf and hard-of-hearing individuals worldwide¹. Due to the limited quantity of SL interpreters, an automatic SL Recognition (SLR) system can bridge the communication barrier between deaf individuals and people without SL knowledge (National Deaf Center, 2022; Nimdzi Insights, 2021; Globo Language Solutions,

2023). Such systems find application in schools, hospitals, and other social environments.

Fingerspelling is an integral component of any sign language, used for parts of speech, such as prepositions, conjunctions, and interjections, proper names, and instances without specific signs, such as addresses, abbreviations, and surnames. For example, to sign the metro station’s name, the SL user should sign it letter by letter, using the corresponding signs for each letter from the SL alphabet. Therefore, fingerspelling is characterized by rapid and swift movements, complicating its recognition by blurry video frames. The motivation to tackle the fingerspelling recognition task is supported by interest from major companies, such as Google, which organized a Kaggle competition (Chow et al., 2023) to address it.

Many prior approaches to fingerspelling recognition are based on neural networks capable of processing the temporal component. To convert a video which consists of a frame sequence $I = \{I_1, I_2, \dots, I_n\}$ to a letter sequence $W = \{W_1, W_2, \dots, W_k\}$, encoder-decoder architectures are typically utilized (Fayyazsanavi et al., 2024; Kumwilaisak et al., 2022; Shi et al., 2018). However, fingerspelling recognition, similar to Automatic Speech Recognition (ASR) and Optical Character Recognition (OCR), is complicated by the lack of alignment between W and I , as multiple frames can interpret a single letter. The Connectionist Temporal Classification (CTC) loss (Graves, 2012) function is widely used in ASR and OCR to address this challenge by facilitating the learning of alignment between input frames and their corresponding output sequences, without the need for explicitly annotated segmentations. So, there is an option to simplify the task and enhance methods’ performance by employing not only RGB frames but also such modalities as depth maps, keypoints, or combinations of video frames and keypoints (Kang et al., 2015; Shi et al., 2019; Fayyazsanavi

¹<https://www.who.int/news-room/fact-sheets/deaf-tail/deafness-and-hearing-loss>

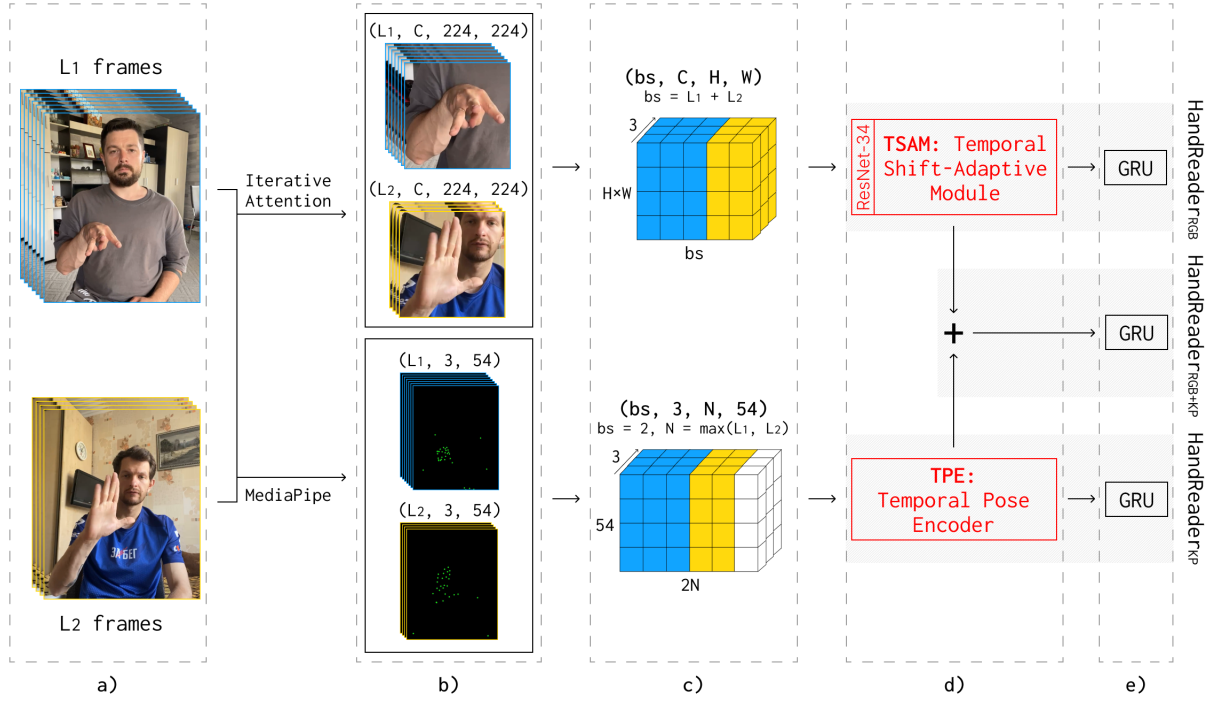


Figure 1: Three HandReader architectures. a) The videos with different lengths b) are processed to get crops and keypoints for HandReader_{RGB} and HandReader_{KP} , respectively, c) form the batches, and d) pass through the encoders. Further, e) the GRU decoder is applied to the encoder outputs. The final output of the $\text{HandReader}_{RGB+KP}$ is obtained by summing the outputs of TSAM and TPE. Red-colored encoders are the paper’s contribution.

et al., 2024; Parelli et al., 2020).

Despite multiple attempts to enhance fingerspelling recognition, the results of prior methods (Kang et al., 2015; Shi et al., 2019; Fayyazsanavi et al., 2024; Parelli et al., 2020; Shi et al., 2018; Pannattee et al., 2024; Kumwilaisak et al., 2022) on two American fingerspelling datasets – ChicagoFSWild (Shi et al., 2018) and ChicagoFSWild+ (Shi et al., 2019) highlight the need for improved accuracy to ensure the applicability of existing approaches in real-world scenarios. Our work aims to enhance letter recognition accuracy while keeping the architectures lightweight.

We aimed to develop an efficient approach for fingerspelling recognition. The contributions of this paper are summarized as follows:

- We introduce the Temporal Shift-Adaptive Module (TSAM), extending the original Temporal Shift Module (TSM) (Lin et al., 2019) to effectively process RGB image features from videos of varying lengths without trimming or padding.
- The Temporal Pose Encoder (TPE) to operate keypoints as three-channel tensors, capturing spatial and temporal information.

- HandReader, a group of three architectures (see Figure 1) that achieved a state-of-the-art result on the ChicagoFSWild+ test set and the best result on ChicagoFSWild among models trained solely on its training set.

We release the pre-trained models and code².

2 Related Work

This overview excludes isolated Sign Language Recognition and Continuous Sign Language Recognition (CSLR) research, as methods for these tasks operate at the sign level: CSLR is further complicated by the need to recognize the independent relationships between signs and letters within words. In contrast, fingerspelling recognition presents a more straightforward one-to-one correspondence.

Recently, the fingerspelling recognition task has been explored using various data modalities, including RGB imagery (Shi et al., 2019, 2018; Pannattee et al., 2024; Kumwilaisak et al., 2022), keypoints (Fayyazsanavi et al., 2024), and depth maps (Kang et al., 2015). Moreover, Parelli et al. (2020) utilized data obtained by combining different modalities. Researchers are motivated to focus on the

²<https://github.com/ai-forever/HandReader>

RGB modality because a fingerspelling recognition system must apply in real-world conditions and resist external changes. Meanwhile, acquiring data from other modalities requires additional equipment, such as extra cameras like Kinect for depth maps or an extra data processing stage for keypoints.

Typical videos for the fingerspelling task contain irrelevant information, such as backgrounds and foreign objects. Therefore, only RGB data requires the model to identify the most relevant areas in the frames, specifically the hands, before recognizing a sign sequence. To simplify locating the signing hand, Shi et al. (2018) developed a hand detection method using Faster R-CNN, then trained a CNN-LSTM model for the recognition task. In their subsequent work (Shi et al., 2019), they replaced the explicit hand detection method with an iterative attention mechanism. The authors of Panattee et al. (2024) and Kumwilaisak et al. (2022) employed representation learning to improve the model’s ability to distinguish different signs by alignment from CTC and incorporated the method from Shi et al. (2019) as a preprocessing stage. Panattee et al. (2024) acquired state-of-the-art results on ChicagoFSWild and ChicagoFSWild+.

Since most RGB modality methods are centered around the hand detection problem, they are limited to exploring different architectures or decoding methods on the final results. In contrast, the keypoint modality avoids this issue because it includes relevant information about hand placement. Fayyazsanavi et al. (2024) trained a Transformer encoder-decoder architecture with a new decoding approach, advancing performance on ChicagoFSWild and ChicagoFSWild+ test sets. Keypoints are usually acquired using MediaPipe (Lugaresi et al., 2019) or another neural network trained for pose estimation. Parelli et al. (2020) combined RGB and keypoint modalities employing linear layers to project the 2D coordinates into 3D space. After extracting all features, they concatenated them and used a decoder based on an attention mechanism. Despite the promise shown by the methods above, recognition accuracy still has potential for improvement. Therefore, our primary objective is to enhance the methodologies employed without compromising the model’s weight and inference speed.

Multi-Stream Sign Language Models. Combining RGB and keypoint streams has also been explored beyond fingerspelling. Zhou et al. (2021)

exploited multiple visual cues, including hands, face, and full-frame appearance, for continuous sign language recognition and translation, while Chen et al. (2022) modeled raw videos and keypoint sequences with a two-stream network for the same tasks. More recently, self-supervised multi-stream approaches, such as SignMusketeers (Gueuwou et al., 2025a) and SHuBERT (Gueuwou et al., 2025b), learn sign language representations from separate face, hand, and body streams. These methods operate at the sign or sentence level and typically involve large-scale pre-training. In contrast, our work addresses the letter-level fingerspelling task, and its contribution lies at the encoder level: TSAM for variable-length RGB processing and TPE for tensor-based keypoint encoding, combined through a deliberately lightweight fusion.

RGB Feature Extraction in the Fingerspelling Task. There are two main ways to extract features in the fingerspelling task. (1) Spatial features can be received by applying a 2D CNN to each frame, then extracting temporal features using 1D convolutions or RNNs. (2) Spatio-temporal features can be obtained with 3D convolution. However, processing frames as a sequence of independent frames lacks temporal information. The TSM (Lin et al., 2019) was developed to address this issue by employing 2D convolutions and shifting information along the temporal dimension. However, TSM processes fixed-size videos, which leads to the loss of valuable information in long videos or negatively affects training time and memory allocation due to the padding in shorter videos. Therefore, this paper presents a modified version of the original TSM to handle videos of varying lengths.

3 Approach

We present a family of three architectures, collectively referred to as HandReader, designed for sign sequence recognition across multiple modalities. The models differ only in the encoder stage (see Figure 1). Thus, HandReader_{RGB} solely utilizes RGB features, HandReader_{KP} relies on keypoints, and HandReader_{RGB+KP} combines both. Each was trained directly using CTC loss, eliminating the need for frame-level letter alignment. The CTC loss accounts for all possible alignments between hand shapes and sign language letters, eliminating the need for explicit alignment supervision. More formally:

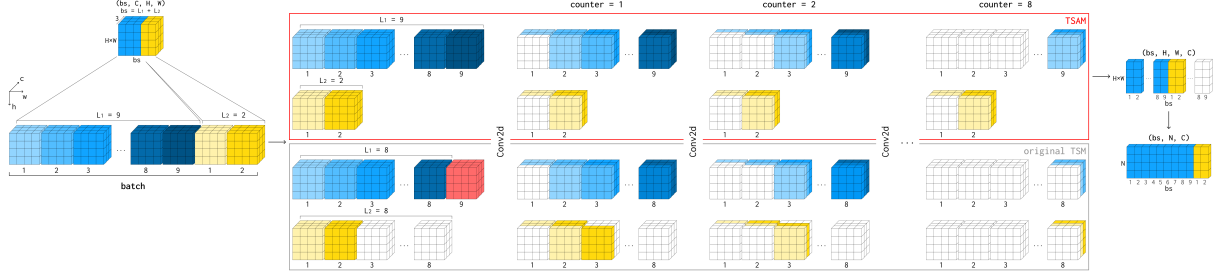


Figure 2: Comparison of the TSAM architecture (outlined in red at the top) with the original TSM (outlined in gray at the bottom). The input batch includes two videos of different lengths, with specific frames highlighted in blue and yellow. While the TSM model requires fixed-length inputs utilizing padding (shown in white) or trimming frames (shown in red) when necessary, the TSAM model employs individual shift counters to manage variable-length sequences without additional padding or trimming. The shift counter displayed at the top of the TSAM block shows how the shifts are activated individually for each video, incrementing the shift counter until it reaches the length of that specific video.

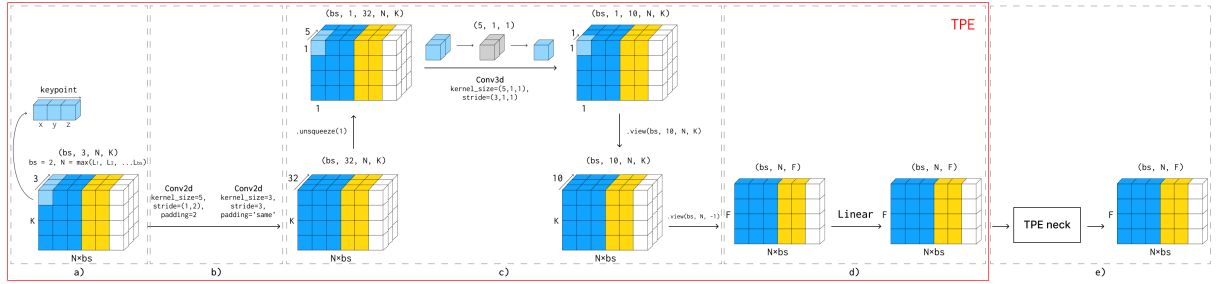


Figure 3: Demonstration of the Temporal Pose Encoder operating on a single batch. a) The encoder receives an input tensor containing N frames and K keypoints. All videos are padded to match the length of the most extended video clip in the batch, allowing them to be stacked. b) 2D convolutions process multiple keypoints across several frames, extracting temporal features. c) Subsequently, a 3D convolution in the form of a “tube” aggregates the extracted motion information for each body part into final feature representations F . d) After applying TPE, features are processed with linear layer, e) followed by MLP with convolution modules from [Gulati et al. \(2020\)](#).

$$L_{CTC} = -\log \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^T p_t(a_t | X), \quad (1)$$

where $\mathcal{A}_{X,Y}$ represents the set of valid alignments between the model input X and the target sequence Y , $A = (a_1, \dots, a_T)$ is one such alignment, and $p_t(a_t | X)$ denotes the probability of symbol a_t occurring at time step t . Each a_t is either a letter of the alphabet or the special blank symbol “_”, introduced by CTC for time steps that emit no letter and to separate repeated letters; blanks and repetitions are removed from an alignment to obtain the final prediction.

A two-layer bidirectional Gated Recurrent Unit (GRU) ([Cho et al., 2014](#)) was chosen as a decoder to generate a sequence of characters, enhancing the model’s capability to capture temporal dynamics. We apply a linear layer with SoftMax to the GRU output to generate posterior probabilities for each

predicted character at every frame. Encoders for each architecture are described below.

3.1 HandReader_{RGB}

The original TSM is a video understanding approach that enables temporal modeling in 2D convolutional networks by shifting portions of channel features along the temporal dimension. However, TSM is constrained to processing a fixed number of frames. Therefore, videos that exceed the frame limit require trimming, while shorter videos need additional empty frames (i.e., padding). Furthermore, the number of shifts is determined by the fixed frame limit. As a result, videos with extra empty frames lose critical sequential information during processing (see Figure 2). We modified the TSM to manage a dynamic number of input frames, avoiding padding and enabling frame-by-frame processing for each video in the batch. Besides, a shift counter is introduced to preserve important sequential information, bypassing shifts on short videos.

Temporal Shift-Adaptive Module. The TSAM was designed to effectively process videos of various lengths. Figure 2 illustrates the distinction between TSAM and TSM operations. The TSAM is incorporated into the ResNet-34 model and operates an input tensor with dimension (bs, C, H, W) , where bs indicates batch size, equal to the sum of the video lengths, C is the number of channels, H and W are the height and width of the frames, respectively. The TSAM also requires an auxiliary list of video lengths, denoted as $L = [L_{S_1}, L_{S_2}, \dots, L_{S_n}]$, where S_i signifies i th video in the batch, and $bs = \sum_i L_{S_i}$. We iterate through L , extracting L_{S_i} frames from the batch to obtain S_i video sequence and apply shifts individually, as detailed in Algorithm 1. Such a method reduces VRAM usage by up to 50% during batch creation, thereby facilitating the processing of longer videos.

The number of shifts is determined by the number of convolutional blocks within the backbone architecture, leading to overwrite temporal features similar to the original TSM. To prevent this loss of information, we introduce a shift counter that stops temporal shifts at the final frame, i.e., no shifting is applied if a video’s length is smaller than the current shift count. Such a method ensures that each video in a batch is treated individually, resulting in more accurate feature extraction.

Each sequence of the received tensor with dimension (bs, H, W, F) , where F represents the number of features, was subsequently padded with zeros to align with the maximum sequence length within the batch. All features are aggregated using 1D convolution, and the resulting tensor is transformed into the format (bs, N, F) , where F is the flattened features across all channels, and N is calculated as $H * W$.

3.2 HandReader_{kp}

We utilized MediaPipe (Lugaresi et al., 2019) to extract body keypoints from every frame. Face keypoints were excluded as stated in Section 5.2; we used 21 keypoints for each hand and 12 body pose keypoints (6 per arm), resulting in 54 (x, y, z) coordinates per frame. The proposed feature encoder is built on a new way to structure keypoints explicitly developed for it, which organizes keypoints within tensors. Each video is treated as a tensor with dimension $(3, N, 54)$, where 54 (x, y, z) vectors are stacked into matrices for each of N video frames. Such a configuration enables the spatial representation of temporal data, allowing the application

Algorithm 1 Applying temporal shifts in TSAM.

Require: Input tensor S of size (bs, C, H, W) , list of sequence lengths $L = [L_{S_1}, L_{S_2}, \dots, L_{S_n}]$
Ensure: Output tensor with temporal shifts applied

- 1: **for** each sequence S_i in batch from input tensor **do**
- 2: $counter \leftarrow 0$
- 3: **for** each convolutional block in backbone **do**
- 4: **if** $counter < L_{S_i}$ **then**
- 5: Apply temporal shift for sequence S_i
- 6: $counter \leftarrow counter + 1$
- 7: **else**
- 8: Skip temporal shift for sequence S_i
- 9: **end if**
- 10: **end for**
- 11: Apply 1D Convolution for S_i
- 12: **end for**

of processing techniques typically used for standard images. We passed the received tensor to the feature extractor described below.

Temporal Pose Encoder. We present a Temporal Pose Encoder based on the convolution layers. TPE operates an input tensor of dimension $(bs, 3, N, K)$, where bs is the batch size, indicating the number of videos in the batch (in contrast to TSAM, where bs denotes the total number of frames across the videos in the batch), N denotes the maximum length of videos in the batch, while shorter videos were padded, and K is the number of keypoint coordinates. The tensor is processed by two 2D convolution layers, followed by a 3D one (see Figure 3). 2D convolutions serve to extract both temporal and spatial features, while transforming keypoint coordinates, enhancing the model’s understanding of dynamic finger movements. At the same time, 3D convolution tries to accumulate coordinate information within a single video with the kernel-tube size of $(5, 1, 1)$, treating each coordinate independently. In this context, the hyperparameter of 5 defines the size of the receptive window along the coordinate dimension, which captures local patterns across five neighboring coordinate features. We insert a dummy dimension in the second position, enabling 3D convolution to treat the original channels as temporal dimensions. The process can be repre-

Dataset	Lang	Sequences	Chars	Hours	Signers	Source
ChicagoFSWild	ASL	7K	38K	2	160	Web
ChicagoFSWild+	ASL	55K	0.3M	14	260	Web

Table 1: Open fingerspelling recognition datasets used in this work.

sented as follows: $(bs, 32, N, K)$ is unsqueezed to $(bs, 1, 32, N, K)$ and processed by 3D convolution to $(bs, 1, 10, N, K)$. A linear layer is then applied to project the features of the keypoints (bs, N, F) , where F is the final encoder’s features. We additionally use multi-layer perceptron (MLP) with convolution modules from Gulati et al. (2020) as a neck between the encoder and the GRU decoder.

3.3 HandReader_{RGB+KP}

We combine features from the HandReader_{RGB} and HandReader_{KP} encoders to take advantage of both modalities. Two output tensors of features from TSAM and TPE are summed and passed to the GRU (see Figure 1).

4 Experiments

This section outlines the experimental setup and data preprocessing procedures used to assess the proposed HandReader architectures and compare them with other approaches. Table 1 shows the open datasets used in this work.

4.1 Experimental Setup

All three architectures were trained from scratch using an AdamW optimizer (Loshchilov and Hutter, 2019) with an initial learning rate of 0.0001. The MultiStepLR modifies the learning rate with a gamma of 0.1 using milestones at 20 and 40 for the HandReader_{RGB} and HandReader_{RGB+KP} models and 25 and 40 for the HandReader_{KP}. We trained the HandReader_{RGB} for 60 epochs, while the other two models were trained for 100 epochs each. A single Nvidia H100 with 80GB RAM was utilized for training.

4.2 Data Preprocessing

The ChicagoFSWild and ChicagoFSWild+ datasets are commonly used to evaluate approaches to the fingerspelling task.

The RGB frames for each dataset were cropped using the model described in Shi et al. (2019), augmented with horizontal flipping and random rotation (see parameters in Table 2), then resized to 224

Augmentation	Parameters	Probability	Modality
Resample	rate=(0.5, 1.5)	0.8	KP, RGB+KP
SpatialAffine	scale=(0.8, 1.2), shear=(-0.15, 0.15), shift=(-0.1, 0.1), degree=(-30, 30)	0.75	KP
TemporalMask	size=(0.2, 0.4), value=0.0	0.5	KP
SpatialMask	size=(0.05, 0.1), value=0.0, mode="relative"	0.5	KP
HorizontalFlip	—	0.5	RGB, RGB+KP, KP
Rotation	angle=(-10, 10)	0.5	RGB, RGB+KP

Table 2: Augmentations used during training.

and normalized. Keypoints were extracted by MediaPipe from raw frames rather than cropped images. We utilized only the hand keypoints (21 per hand) and 12 body pose keypoints (6 per arm), resulting in 54 keypoints. We replaced the NaN values with zeros, normalized the rest, and applied augmentations specified in Table 2. Only resampling, horizontal flipping, and random rotation augmentations are applied when combining two modalities in HandReader_{RGB+KP}.

4.3 Results

We assessed HandReader architectures using the letter accuracy metric: $1 - \frac{S+D+I}{N}$, where S , D , and I denote substitutions, deletions, and insertions, respectively, N is the length of ground-truth sequence. Metric values for other methods were obtained from the original papers for comparison purposes. Each HandReader version outperformed all prior methods trained on a single training set on both benchmarks, and HandReader_{RGB+KP} achieved a new state-of-the-art result on ChicagoFSWild+ (see Table 3). Note that Conformers (CFW+CFW+) (Shi, 2023) reaches a higher score on ChicagoFSWild; however, it was trained on the union of the ChicagoFSWild and ChicagoFSWild+ training sets, whereas each HandReader model was trained only on the training set of the corresponding benchmark. We also present prediction comparison between PoseBased Transformer (Fayyazsanavi et al., 2024) and our models in Table 4. For decoding, all models employed the beam search strategy with a beam width of 5, which explores multiple candidate sequences at each step and selects the most probable transcription.

Table 5 provides qualitative examples of predictions produced by the three HandReader variants on the ChicagoFSWild dataset. We observe that the keypoint-based model (**KP**) tends to generate shorter and less stable outputs, frequently dropping characters (e.g., “youtube” → “outube”) or producing heavily distorted strings. The RGB-based model (**RGB**) is generally more consistent

in capturing the global temporal structure, but still suffers from confusions between visually similar finger configurations, resulting in character-level substitutions (e.g., “organic” → “orgadic”). The multimodal model (**RGB+KP**) demonstrates the most robust behavior across the examples, often correcting errors present in the unimodal variants, which indicates that RGB and keypoint cues provide complementary information.

To further analyze the error patterns, Table 6 reports the SDI (Substitution, Deletion, Insertion) breakdown for each model under greedy and beam-search decoding. Across all models, **deletions** constitute the dominant error type, highlighting the difficulty of maintaining correct character continuity in fast fingerspelling sequences. Beam search consistently reduces the number of deletions for all variants (e.g., from 677 to 628 for RGB, and from 857 to 789 for KP), at the cost of a small increase in substitutions and insertions. This trade-off suggests that beam search improves sequence completeness but may introduce additional character-level confusions. Overall, the **RGB+KP** model achieves the best balance between the three error types, reinforcing the benefit of multimodal fusion for fingerspelling recognition.

5 Ablation

This section provides an ablation study to demonstrate the impact of each component on the HandReader’s overall performance. Note that HandReader_{RGB+KP} was excluded from the study since it relies solely on the heuristics of the other two architectures. We maintained a consistent experimental setup described in Section 4 and changed only one component at a time to analyze its influence. The ChicagoFSWild dataset was utilized in all experiments.

5.1 HandReader_{RGB}

Encoder. Table 7 indicates that integrating TSM into the baseline model improves the performance metric by 1.69. Furthermore, replacing TSM with TSAM leads to an additional 1.4 improvement. To isolate the effect of the count shift mechanism, we implemented a variant of TSAM without the timely stopping of shifts, which led to a 1.49 performance degradation.

Decoder. We experimented with the LSTM (Hochreiter and Schmidhuber, 1997) decoder instead of the GRU (Cho et al., 2014) one, which

resulted in a difference of 1.69. Replacing RNN with a linear layer reduces the metric by 11.0. Such observations underlined that high-level features of each frame cannot be used as independent features for final predictions.

Others. Table 7 highlights the significant impact of augmentations, as a model trained without rotations and horizontal flips reached a final metric of 70.49. Such influence is explained by the dataset containing signers with different dominant hands and videos where hands are misaligned due to camera angles or the signer’s position. Additionally, we tested different batch sizes and showed that a bigger batch size is not always optimal. It is especially noticeable in ablation studies with RGB. This may happen because the batch contains videos of varying lengths, complicating the model’s decoding process.

5.2 HandReader_{KP}

Keypoints. We tested possible combinations of different parts of keypoints and provided results in Table 8. The metric was reduced by adding face keypoints since sign language users can express the same phrases differently without verbal language skills. The model performs better with pose and hand keypoints than with just hand keypoints, as the pose coordinates carry information about the position of the whole hand, which may help the model better understand the hand’s relative position throughout the video.

Encoder and Decoder. The same procedure of RNN impact assessment is provided in Table 8. In contrast to Section 5.1, the usage of the linear layer does not show a significant metric drop, as the HandReader_{KP} encoder contains more modules for processing temporal dimensions. We studied the impact of the number of convolution modules utilized in the encoder. As the number of convolution blocks increases, the final metric declines.

Others. The augmentations have a crucial effect on model performance. We also examined various batch sizes: 4, 6, and 8. As the batch size increases, there are noticeable metric drops.

5.3 HandReader_{RGB+KP}

Feature reduction. We evaluated various feature reduction techniques, including simple summation, feature concatenation, element-wise multiplication, and weighted summation, obtaining 72.95, 72.95, 72.55, and 72.57 letter accuracy, respectively. We preferred the simple summation due to the smaller

Model	ChicagoFSWild	ChicagoFSWild+	Modality
Hand Det + CNN + RNN (Shi et al., 2018)	41.9	41.2	RGB
Iterative Attention (Shi et al., 2019)	45.1	46.7	RGB
3d hand pose + RGB (Parelli et al., 2020)	47.9	–	RGB + KP
SiameseNet+Iterative model (Kumwilaisak et al., 2022)	49.6	–	RGB
Iter train + SupCon (Pannattee et al., 2024)	64.9	71.7*	RGB
Pose-based transformer (Fayyazsanavi et al., 2024)	66.3	71.1	KP
Conformers (CFW) (Shi, 2023)	68.4	60.2	RGB
Conformers (CFW+CFW+) (Shi, 2023)	77.5*	73.0*	RGB
HandReader_{KP}	69.3	72.4	KP
HandReader_{RGB}	72.0	73.8	RGB
HandReader_{RGB+KP}	72.9**	75.6	RGB + KP

Table 3: Results comparison on ChicagoFSWild and ChicagoFSWild+ test sets. * indicates the model was trained on both ChicagoFSWild and ChicagoFSWild+ training sets. ** means two convolution modules were used for HandReader_{KP} encoder, while only one was employed for other metrics.

PoseBased (Fayyazsanavi et al., 2024)*	KP	RGB+KP	RGB	Reference
macah	mach	mach	mach	mach
meach	meah	beack	fealc	beach
hospital	hospital	hospital	hospital	hospital
-	outube	iutube	iutube	youtube
-	organic	organic	orgadic	organic

Table 4: Results comparison of predictions between previous SotA solution and our models, predictions for PoseBased model were taken from the original paper.

Gt	KP	RGB	RGB+KP
youtube	outube	iutube	iutube
issue	isue	isue	isue
organic	or ganic	orgadic	organic
pharmacology	opharmacoint	opharocolohy	opharmacolgi
pharmacology	harmcogiy	haroscothy	harmacotgy
physical	phisical	phylsical	phistical
audio	uio	ardio	atudio
biochemistry	biochemitry	biockemistry	biochemiry

Table 5: Result of comparison of predictions between different models on ChicagoFSWild dataset.

dimension of the sum output than concatenation one. The evaluation was conducted on the test set of the ChicagoFSWild dataset.

6 Conclusion

This paper presents HandReader, a suite of three distinct architectures that significantly advance the fingerspelling recognition task. The Temporal Shift-Adaptive Module and Temporal Pose Encoder were developed to capture RGB and keypoint features effectively. We achieved state-of-the-art performance on the ChicagoFSWild+ dataset and the best performance on ChicagoFSWild among models trained solely on its training set. The ablation study revealed that the proposed techniques and training pipelines are essential for achieving optimal results. The presented findings offer valuable insights for enhancing fingerspelling recognition

Model	D	S	I
RGB	677 / 628	462 / 478	122 / 130
KP	857 / 789	382 / 406	148 / 161
RGB+KP	683 / 621	386 / 409	160 / 164

Table 6: SDI breakdown on ChicagoFSWild. Each cell shows Greedy / Beam decoding.

systems.

Limitations

Architectures. Although the HandReader_{KP} and HandReader_{RGB} effectively recognize fingerspelling, extra models are required to extract keypoints and hand region crops, respectively. Such necessities add computational overhead, decreasing inference speed and limiting the model’s real-time usability, particularly on devices with limited processing power. There is an opportunity to replace an additional model with center cropping augmentation in the HandReader_{RGB}, provided the input focuses on the hands’ region in the center of the frame. The HandReader_{RGB+KP} is subject to the limits of other HandReader architectures, but it can utilize only the keypoint prediction model and obtain crops operating received points.

References

- Yutong Chen, Ronglai Zuo, Fangyun Wei, Yu Wu, Shujie Liu, and Brian Mak. 2022. [Two-stream network for sign language recognition and translation](#). In *Advances in Neural Information Processing Systems*, volume 35.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. [Learn-](#)

Experiment	Packed sequence	RNN	Count shift	Shift type	Image Augs.	BS	Letter accuracy
best	✓	GRU	✓	TSAM	✓	4	72.0
Packed sequence	✗	GRU	✗	TSM	✓	4	70.6 _{-1.4}
RNN	✓	LSTM	✓	TSAM	✓	4	70.31 _{-1.69}
	✓	✗	✓	TSAM	✓	4	61.0 _{-11.0}
Count shift	✓	GRU	✗	TSAM	✓	4	70.51 _{-1.49}
Image Augs.	✓	GRU	✓	TSAM	✗	4	70.49 _{-1.51}
BS	✓	GRU	✓	TSAM	✓	2	71.46 _{-0.54}
	✓	GRU	✓	TSAM	✓	6	70.1 _{-1.9}
Baseline*	✗	GRU	✗	None	✓	4	69.63 _{-2.37}

Table 7: Ablation study of HandReader_{RGB} components. * denotes the baseline setup with a ResNet34 configuration (without TSM or TSAM shift operations), which already outperforms previous state-of-the-art solutions. Here, we define BS as batch size and RNN as the type of recurrent neural network used, which may include LSTM, GRU, or a linear layer without recurrence. “Packed sequence” refers to a modification of the TSM mechanism that enables processing each video independently during the shift operation, as described in Section 3.1.

Experiment	Keypoints			RNN	Conv. modules	Image Augs.	BS	Letter accuracy
	hand	pose	face					
best	✓	✓	✗	GRU	1	✓	4	69.31
Keypoints	✓	✗	✗	GRU	1	✓	4	68.54 _{-0.77}
	✓	✓	✓			✓		69.08 _{-0.23}
	✓	✗	✓			✓		67.59 _{-1.72}
	✗	✓	✓			✓		10.02 _{-59.29}
	✗	✓	✗			✓		10.36 _{-58.95}
	✗	✗	✓			✓		11.08 _{-58.23}
RNN	✓	✓	✗	LSTM	1	✓	4	67.79 _{-1.52}
	✓	✓	✗	✗		✓		63.88 _{-5.43}
Conv. modules	✓	✓	✗	GRU	2	✓	4	68.09 _{-1.22}
	✓	✓	✗		3	✓		68.11 _{-1.2}
Image Augs	✓	✓	✗	GRU	1	✗	4	62.43 _{-6.88}
BS	✓	✓	✗	GRU	1	✓	2	69.04 _{-0.27}
	✓	✓	✗			✓	6	69.26 _{-0.05}
	✓	✓	✗			✓	8	68.45 _{-0.86}

Table 8: Ablation study of HandReader_{KP} components. Here, “Conv Modules” denotes the number of convolutional modules from the Conformer architecture that are used inside the encoder. Other terms are defined in Table 7.

ing phrase representations using rnn encoder-decoder for statistical machine translation. *Preprint*, arXiv:1406.1078.

Ashley Chow, Glenn Cameron, Manfred Georg, Mark Sherwood, Phil Culliton, Sam Sepah, Sohier Dane, and Thad Starner. 2023. Google – american sign language fingerspelling recognition. <https://kaggle.com/competitions/asl-fingerspelling>. Kaggle.

Pooya Fayyazsanavi, Negar Nejatishahidin, and Jana Košecká. 2024. Fingerspelling posenet: Enhancing fingerspelling translation with pose-based transformer models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1120–1130.

Globo Language Solutions. 2023. Why are asl interpreters so hard to find? <https://www.helloglobo.com/blog/why-are-asl-interpreters-so-hard-to-find>.

Alex Graves. 2012. Connectionist temporal classification. *Supervised sequence labelling with recurrent neural networks*, pages 61–93.

Shester Gueuwou, Xiaodan Du, Greg Shakhnarovich, and Karen Livescu. 2025a. **SignMusketeers: An efficient multi-stream approach for sign language translation at scale**. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22506–22521.

Shester Gueuwou, Xiaodan Du, Greg Shakhnarovich, Karen Livescu, and Alexander H. Liu. 2025b. **SHuBERT: Self-supervised sign language representation learning via multi-stream cluster prediction**. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28792–28810.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. **Conformer: Convolution-augmented**

- transformer for speech recognition. *Preprint*, arXiv:2005.08100.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural computation*, 9:1735–80.
- Byeongkeun Kang, Subarna Tripathi, and Truong Q Nguyen. 2015. Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 136–140. IEEE.
- Wuttipong Kumwilaisak, Peerawat Pannattee, Chatchawarn Hansakunbuntheung, Nattanun Thatphithakkul, et al. 2022. American sign language fingerspelling recognition in the wild with iterative language model construction. *APSIPA Transactions on Signal and Information Processing*, 11(1).
- Ji Lin, Chuang Gan, and Song Han. 2019. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). *Preprint*, arXiv:1711.05101.
- Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. 2019. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.
- National Deaf Center. 2022. The asl interpreter shortage and its impact on accessibility in college settings. <https://nationaldeafcenter.org/news-items/the-asl-interpreter-shortage-and-its-impact-on-accessibility-in-college-settings/>.
- Nimdzi Insights. 2021. American sign language interpreting: The size and state of the market. <https://www.nimdzi.com/asl-interpreting/>.
- Peerawat Pannattee, Wuttipong Kumwilaisak, Chatchawarn Hansakunbuntheung, Nattanun Thatphithakkul, and C-C Jay Kuo. 2024. American sign language fingerspelling recognition in the wild with spatio temporal feature extraction and multi-task learning. *Expert Systems with Applications*, 243:122901.
- Maria Parelli, Katerina Papadimitriou, Gerasimos Potamianos, Georgios Pavlakos, and Petros Maragos. 2020. Exploiting 3d hand pose estimation in deep learning-based sign language recognition from rgb videos. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 249–263. Springer.
- Bowen Shi. 2023. [Toward american sign language processing in the real world: Data, tasks, and methods](#). *Preprint*, arXiv:2308.12419.
- Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Jonathan Michaux, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. 2018. [American sign language fingerspelling recognition in the wild](#). In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 145–152.
- Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. 2019. Fingerspelling recognition in the wild with iterative visual attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. 2021. [Spatial-temporal multi-cue network for sign language recognition and translation](#). *IEEE Transactions on Multimedia*, 24:768–779.