International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Real-Time Sign Language Recognition from Webcam Landmarks Using a Multi-Stream Temporal Attention Network

Raunak Dahiwile[1], Aryan Tajane[2], Dolly Rakhde[3], Kaustubh Khode[4], Gautamkumar Mehta[5], Mansi Lanjewar (Guide)[6]

[1–6]Department of Computer Technology, Priyadarshini College of Engineering, Nagpur, India

---

Abstract - Very few hearing people know how to interpret sign languages, and interpreters are rare. In this work, we introduce SignVision - a system capable of recognizing isolated ASL words from videos using only a webcam. MediaPipe Holistic transforms every video frame into hand and body pose landmarks which are normalized and concatenated with frame-to-frame velocities to create a 454-dimensional feature vector. Multi-stream classification model splits hand and pose feature streams independently to pass them through two separate bidirectional LSTMs, merge them using cross-modal attention and aggregate the sequence with masked self-attention pooling. The model has around 0.56 million parameters. Rule-based layer translates recognized glosses into English subtitles to display on a web page. Training was performed on the 67 WLASL100 glosses, whose videos were still accessible online, totaling 402 clips. We also introduce a dataset of 15,000 frames which we gathered ourselves. In a random 80/10/10 split, the model reaches 39.0% top-1 and 70.7% top-5 accuracy on 41 out-of-distribution clips, compared to 1.5% and 7.5% expected by chance. Our pipeline works at a rate of around 32 fps on CPU. Due to the limited test set and lack of stratification and signer independence in the split.

Key Words: sign language recognition, MediaPipe Holistic, BiLSTM, cross-modal attention, WLASL, real-time systems.

1. INTRODUCTION

Over 1.5 billion individuals are estimated to be affected by hearing impairments by the World Health Organization, with about 430 million having a disability causing hearing impairment [27]. For those who communicate through the use of signing language, their means of communication faces another challenge: there is a lack of interpreters to assist the conversation. Normal conversations at a pharmacy, a school office, and even a bus stop often take place without the presence of an interpreter.

The legislation is starting to address this issue. The Rights of Persons with Disabilities Act, 2016 in India requires both the State and organizations to facilitate accessibility and use of sign language [16]. A camera-based translation tool cannot replace an interpreter and this paper is not suggesting that it should be used for that purpose. This tool is nevertheless useful when an interpreter is unavailable as long as it is cheap to use and honest about its mistakes.

Each existing solution suffers from some weaknesses. While instrumented gloves accurately track finger articulations, they are expensive and uncomfortable to wear [5]. Video-to-3D Convolution Network solutions require the presence of a GPU as well as training on how to ignore background, clothing, and lighting [4, 19]. Our chosen method extracts body landmarks from each frame and performs classification on the outputted landmark sequence classification [12, 20, 22, 25, 28]. This classifier is lightweight enough to run on a standard laptop CPU, and no video data leaves the device at all..

Sign language translation is more than simply attaching a label to an image. The system needs to locate the hands and body of the signer in each image whatever the distance and the background; to represent hand shapes, locations and motions efficiently; to differentiate signs which appear similar in isolation; to determine where one sign ends and another begins; and to provide its output in the form of a sentence at a speed which will allow a natural conversation to be maintained. Systems which fail to do at least one of these things tend to fall apart in real-world environments, need massive hardware resources or to produce strings of unrelated words. This project deals directly with the first three problems, and addresses the other two only partially as described in Sections 4 and 6.

SignVision is presented in this work – an ASL word recognition system relying only on webcams, as well as what it was able to do using the available data. This work makes four contributions. First, a 454-dimensional per-frame representation of the normalized hand and upper body landmarks, their velocities, and presence-of-hand flags. Second, a multi-stream classifier known as Multi-Modal Temporal Attention Network for Sign Language Recognition (MTAN-SLR) – it uses two bidirectional LSTM streams for hand and pose features respectively, with cross-modal attention being used to combine them. Third, a real-time web application with REST and WebSocket backend services and ONNX export. Fourth, an evaluation which explicitly describes all the limitations of the experiment. These limitations are substantial – one third of the WLASL100 vocabulary is no longer downloadable, and we mention it whenever it impacts reported numbers.

The rest of the paper is organized as follows. Sec. 2 reviews prior work and Sec. 3 describes the requirements of the design. Sec. 4 describes the system, Sec. 5 explains the dataset and evaluation protocol, and Sec. 6 the experimental results. Privacy, fairness, licensing, and reproducibility are discussed in Sec. 7, future work, including a portable version, in Sec. 8, and concluding remarks in Sec. 9. The source code is available under the MIT license at: https://github.com/ronnycloud/SIGNZ.

2. RELATED WORK

2.1 Sensor-based and vision-based recognition

Cheok et al. [5] divide the literature into two broad categories based on whether sensor-based methods are used – that is, where there is use of data gloves with flex sensors and accelerometers, or vision-based methods, where there is use of cameras. The data glove performs well in capturing the configuration of the hand, but it requires calibration, is expensive and cannot capture the face, making it more appropriate for lab environments rather than service counters in a public office setting. Ong and Ranganath [14] note that the interpretation of a sign goes beyond handshape alone and encompasses both grammar and facial/whole-body language.

Rastgoo et al. [19] survey deep-learning approaches to sign recognition, both static and dynamic.

The vision-based system usually sends the raw frame or optical flow data to 2D/3D convolutional neural networks. The video models such as the I3D [4] have very good results on large datasets, but need the neural network to learn to disregard the background and clothing color, require a huge amount of labeled video data, and are expensive to process at video speed. This is why the landmark-based approach is chosen..

2.2 Landmark-based methods

MediaPipe delivers real-time tracking of 21 three-dimensional landmarks per hand [28], as well as a framework combining the hand, pose and face models [12] using regular CPUs. The landmark detector being trained on the dataset representing general human pose information instead of signs allows it to be reused for different signers and languages and keeps the task of recognizing body parts separate from the task of recognizing a gesture performed by those parts.

The above methodology has been exploited by several papers on Indian sign language. Subramanian et al. [22] utilize MediaPipe landmarks with a GRU, while Velmathi and Goyal [25] utilize MediaPipe Holistic. Rawat et al. [20] use it with LSTMs for dynamic and static gestures. iSign, introduced by Joshi et al. [8], is the Indian sign language processing benchmark. This shows that landmark-based features are highly effective at reducing the input size, while preserving a majority of information which differentiates signs from each other. Nevertheless, almost all of them feed concatenated features to one recurrent stream, while classifying only isolated signs.

2.3 Temporal models and attention

Dynamic signs are determined by the changes in hand shape, location, and movement through time. LSTMs [7] are used for this task traditionally, while bidirectional recurrent networks [21] use information from both past and future frames, making them suitable for isolated signs that are defined by both the initial and final states. Self-attention [24] establishes connections between all frames simultaneously. Cui et al. [6] use a spatial-temporal Transformer for sign recognition, and Camgoz et al. [3] use them for recognition and translation.

Sequential processing of frames in recurrent models limits parallelization, while transformers require more data or regularization in case of limited datasets. Given that the number of available clips was very small, we chose recurrent encoders because of their effectiveness in case of short sequences, while applying attention mechanism to the two places it could be most beneficial – between hand and pose streams, and for pooling over time.

2.4 Datasets and sentence-level output

The WLDSL [9] dataset offers videos of 2,000 glosses made by several signers and is considered as the standard for word-level ASL recognition. The authors mention that they get around 66% top-1 accuracy of I3D on a subset of 100 glosses. In case we want to convert gloss sequence into an English sentence, T5 (Text-to-Text Transfer Transformer) [18], which treats all languages as text generation, is offered by Hugging Face library [26]. Camgoz et al. [3] formulate sign language translation as sequence to sequence problem.

2.5 Positioning

Table -1 places SignVision within one of the major paradigms. Gloves are accurate but costly. Raw pixel systems are cumbersome and are influenced by the environment. Landmark-based single stream models are fast but include the fine grained movements of the fingers and coarse movements of the body together. End-to-end translation systems require large databases and graphics processing units. SignVision lies between these paradigms – it relies on landmarks, two recurrent streams coupled using attention and rules based sentence layer.

Table -1: Comparison of sign language recognition approaches.

Approach	Input	Temporal model	Continuous signing	Sentence output
Data glove [5]	Glove sensors	Varies	Limited	No
CNN or 3D CNN [4, 19]	Raw frames	Recurrent or 3D conv.	Fixed window	No
Landmarks + GRU/LSTM [20, 22, 25]	Landmarks	One recurrent stream	Fixed window	No
Landmarks + Transformer [6]	Landmarks	Self-attention	Fixed window	No
End-to-end translation [3]	Video or features	Transformer	Yes	Yes, with large data
SignVision (this work)	**Landmarks + velocity**	**Two BiLSTM streams + attention**	**Heuristic boundaries**	**Rule-based**

2.6 Research gap

There are three recurring shortcomings in this literature. Firstly, many of the milestone systems use one stream for coding hand and body information, which makes it difficult to disambiguate fine grained hand movements from gross body movements as well as ambiguous signs that differ in location. Secondly, most of the systems analyze isolated gestures or perform segmentation on continuous video through the same fixed-size window, and thus are not good at handling co-articulation. Thirdly, very few systems have tried to integrate recognition and sentence generation into one pipeline and measure its processing cost. SignVision solves the first problem completely but addresses the other two partially; we try to show how.

3. REQUIREMENTS AND DESIGN PRINCIPLES

Requirements arise out of three categories of users. Signer wants accurate recognition without carrying any equipment. Non-signing listener desires reading a correctly generated sentence without much delay. Deployer, which can be a hospital, school or office, desires a system which should run on an existing computer and process video locally. Principles listed in table -2 arise out of these requirements.

Table -2: Design principles and the reason for each.

Principle	Reason
Landmarks, not pixels	Eliminates variation in background, clothing and lighting, and reduces compute and data requirements.
Invariance by normalisation	Re-centring and scaling make features independent of the signer's position and distance.
Motion as a feature	Velocities distinguish signs that share a posture but move in opposite directions, such as YES and NO.
Decoupled modalities	Separate hand and pose streams stop the larger pose stream from overwhelming finger detail.
Prefer silence to a wrong answer	Low-confidence predictions are discarded rather than displayed.
Privacy by design	Frames are processed on the device and never stored or transmitted; only landmarks are used afterwards.

The design process was incremental in nature, starting with requirements analysis and literature survey followed by data collection and pre-processing and landmark detection. These steps were followed by feature and model design and training, and finally the pipeline and interfaces. Every step was validated before proceeding to the next step and feedback was taken from mistakes for feature and model design. The initial prototype consisted of an LSTM model designed in Keras using webcam images of three different gestures, which was later replaced by the PyTorch model discussed below.

4. SYSTEM DESCRIPTION

4.1 Overview

Figure -1 shows the flow diagram. OpenCV [2] collects frames from the camera at about 30 FPS. Each frame is converted into landmarks that create a 454-Dimensional feature vector. These feature vectors are collected in a buffer that can hold up to 90 frames. The classification component gives a mapping between the buffered sequence and the gloss with a confidence score. If the decision gate passes it, then it is translated to an English sentence and sent as a subtitle to the browser.

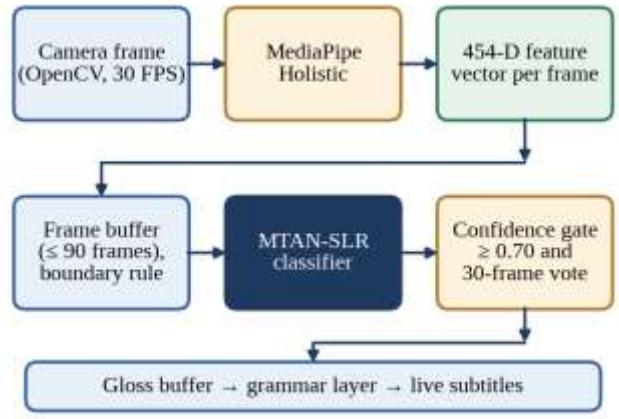


Fig -1: The SignVision pipeline, from camera frame to subtitle.

4.2 Landmark features

Each frame is fed into the MediaPipe Holistic pipeline [12]. We keep 21 three-dimensional landmarks for each hand (126 values) and 25 out of 33 pose landmarks, which have four values (x, y, z, and visibility). We discard the legs' landmarks since they are almost meaningless in ASL and simply introduce noise. We do not keep the 468 face landmarks either to keep the model small, thus, missing some important cues like eyebrows and mouth position, an aspect discussed further in Section 6.7. Raw per-frame vectors contain all 33 pose landmarks (258 values), but only the 226 values described above are used for features.

Raw coordinates depend on the position of the signer relative to the camera. In case of a hand landmark p, where w and s denote the wrist's position and palm span respectively, subtracting the wrist's position and dividing by the palm span cancel these factors,

$$p' = (p - w) / s \quad (1)$$

Pose landmarks are also processed in the same way – they are re-centered using the body reference point and normalized with the body size scale. The motion component is computed as a first-order difference:

$$vt = xt - xt{-}1 \quad (2)$$

for all 226 normalized values. Two additional flags indicate whether hands were detected or not, enabling the model to distinguish a missing hand from a hand at the origin point. Table -3 and Fig -2 show the distribution of 454 dimensions..

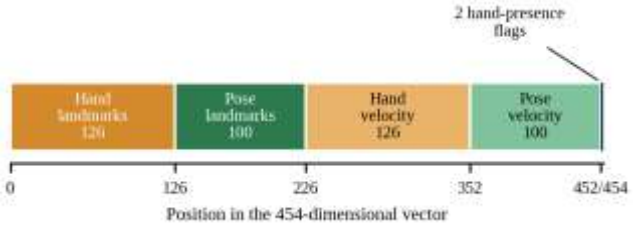


Fig -2: Layout of the 454-dimensional feature vector.

Table -3: Composition of the per-frame feature vector.

Component	Dim.	Content
Hand landmarks	126	2 hands × 21 × (x, y, z)
Pose landmarks	100	25 × (x, y, z, visibility)
Velocities	226	Differences of the 226 values above
Hand-presence flags	2	One per hand
Total	**454**	

4.3 Augmentation

As there are about six instances for each word, the burden on augmentation is considerable. -4 shows the list of augmentations used, which are performed at the time of training. The mirroring transforms the two hands, because a left-handed signer makes the mirror of a right-handed signer.

Table -4: Augmentation settings.

Augmentation	Setting
Temporal jitter	±2 frames skipped or repeated
Gaussian spatial noise	σ = 0.005
Scale variation	0.9 to 1.1
Mirroring	Swap left and right hands
Speed variation	0.8× to 1.2×

4.4 MTAN-SLR model

There are 454 input streams separated by modalities (Fig -3). Hand stream is made up of 254 elements (landmarks on hands, velocities of hands, and two flags), and passes through two stacks of BiLSTMs with 64 units per direction. Pose stream contains 200 elements and passes through one BiLSTM with 32 units per direction. Having two separate encoders is supposed to ensure that the large pose stream does not dominate the early optimization of finger articulation.

Then a cross-modal attention block (4 heads, residual connection, layer normalization) makes it possible for each hand stream to access information about the body context. It is based on scaled dot-product attention [24].

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^{T} / \sqrt{d_k})\, V \quad (3)$$

The queries are generated from the hand stream, and the keys and values from the pose stream. The aim is to differentiate between signs which have similar hand shape, but dissimilar location, for example, THINK at forehead location and FEEL at chest location. The attended hand features and the pose features (192 per frame) are concatenated and fed into a masked temporal self-attention layer, which has a classification token and 4 heads. This allows giving higher weights to the hold phase of the sign than its transitions. Masking is required because the video clips are of different lengths, and therefore the padding should not affect the attention weights. The MLP head, which

has 128 hidden units and dropout of 0.3, gives an output of 100 logits. There are 559,396 parameters in total in the network. The PyTorch checkpoint takes around 6.5 MB in size, while the ONNX model is around 2.2 MB.

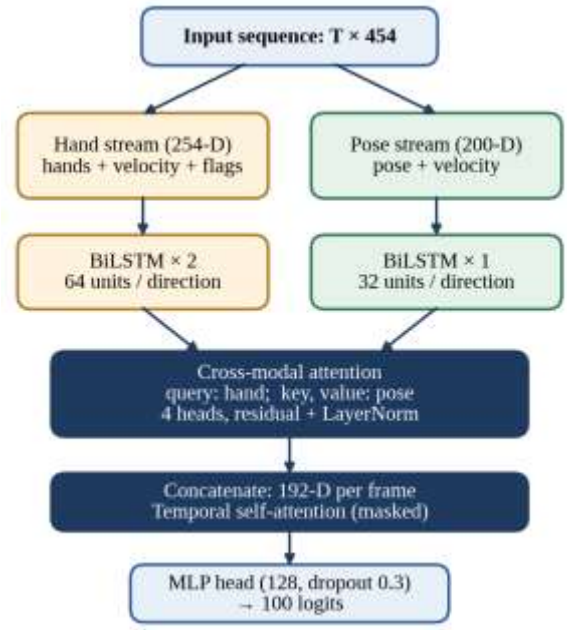


Fig -3: Architecture of MTAN-SLR. The pose representation is also concatenated with the attended hand representation before temporal pooling.

Moreover, there are three additional architectures that can be formed using the same configuration file (Table -5). It means that we can make comparisons using the same pipeline of data, its splitting and training process. Unfortunately, we have not made them yet.

Table -5: Architectures selectable in the model configuration.

Model	Key configuration
bilstm_attention	Primary model, as described above
transformer	d_model 256, 8 heads, 4 layers, feed-forward 512
cnn_lstm	1D CNN [64, 128, 128], kernels [3, 5, 7], then LSTM 128 × 2
baseline_lstm	Stacked LSTM [64, 128, 64], dense [64, 32]

4.5 Training

We use AdamW [10], cosine annealing scheduling [11] and cross entropy loss with label smoothing [23], the numbers are listed in Table -6. Training stops when the validation loss does not decrease in the last 20 epochs and we save the checkpoint with the minimum validation loss. The sequences are trimmed or zero padded to 90 frames and zero padding is masked. The model is implemented using PyTorch [17]. We activate automatic mixed precision [13] in the settings, but it works only on a CUDA device.

Table -6: Training configuration.

Item	Value
Optimiser	AdamW, learning rate 1e-3, weight decay 1e-4
Schedule	Cosine annealing, T_max = 50, floor 1e-5
Loss	Cross-entropy, label smoothing 0.1
Batch size	32
Epochs / patience	Up to 200 / 20
Gradient clipping	Norm 1.0
Sequence length	90 frames, padded and masked
Seed	42

One thing should be mentioned. Given that Tmax = 50 and the upper bound is 200 epochs, the cosine schedule in PyTorch lets the learning rate increase again after epoch 50. It is okay and even can be helpful, but the schedule is thus not one smooth decay as its name would suggest..

4.6 Live inference

The camera loop keeps a buffer of a maximum of 90 frames. The prediction is accepted only if the confidence is greater than 0.70 and wins a majority vote among the last 30 frames. Any segments failing this criterion are considered as transition regions and discarded based on the philosophy that it is better to err on the side of caution and be silent than to provide an incorrect response. Boundaries are obtained using a velocity-energy approach where there is slowing down of the hands between the signs; thus, lower velocity energy marks the likely boundary. There is a segmenter component implemented in the code, but boundary finding in the live system is heuristic currently.

4.7 Language layer

The structure of signed languages is governed by its own word order and lacks function words; thus, a glossed list like I, WATER, WANT cannot form an English sentence yet. Glosses that can be accepted are collected and presented to the rule-based layer which then forms a sentence out of them. We provide a point of reference for our improved T5 model [18], but we have yet to train one. The rule-based layer does not learn ASL grammar but is a readable approximation instead.

4.8 Interfaces and software

The web app is implemented using Flask. The camera stream is served to the browser via MJPEG protocol, while the subtitles, FPS, and status messages are sent using Server-Sent Events. The FastAPI server provides REST and WebSocket endpoints for external access, while the command line tool renders the output in an OpenCV window. Parameters of the model, training, and inference process, such as confidence threshold and smooth window sizes, are defined in YAML configuration files to allow modification without altering the source code. One of the scripts converts the model into ONNX and compares its performance with PyTorch, while the other one measures latency. Table -7 shows the software.

Table -7: Software used.

Layer	Tools
Language	Python 3.11
Vision	OpenCV [2], MediaPipe Holistic [12]
Deep learning	PyTorch 2.x [17], ONNX and ONNX Runtime
Prototype	Keras on TensorFlow [1]
Utilities	scikit-learn, NumPy, tqdm
Back end	Flask (SSE, MJPEG); FastAPI with Uvicorn
Front end	HTML, CSS, JavaScript
Data tools	yt-dlp for the WLASL downloader

Table -8 lists the main files and their roles, as a guide for readers who wish to follow the code.

Table -8: Main parts of the repository.

Part	Role
app.py, templates/, static/	Flask web app: live video, landmark overlay, subtitles, frame rate
api_server.py	FastAPI REST and WebSocket server for remote clients
inference.py	Command-line inference in an OpenCV window
train.py	Training loop with early stopping and checkpointing
scripts/evaluate.py	Accuracy per split and latency, on clean data
scripts/export_onnx.py	ONNX export with a check against PyTorch
scripts/benchmark.py	Latency and throughput benchmark
src/features, src/data	Landmark extraction, normalisation, velocities, datasets, augmentation
src/models, src/nlp	MTAN-SLR and the alternative models; gloss-to-English layer
configs/	Model, training and inference settings in YAML

5. EXPERIMENTAL SETUP

5.1 WLASL100

The 100-gloss subset of WLASL [9] is used, with videos downloaded from sign language websites and YouTube. Most of the links are broken now. We were able to process 402 videos for 67 of the 100 glosses; the rest 33 glosses do not have any videos (see Table -9). The output layer consists of 100 classes, although the empty classes never appear in the training process. The landmarks are detected once and saved in one array per video, thus avoiding the need to re-decode the videos each time during training. WLASL is released under the Computational Use of Data Agreement (C-UDA), and thus it is not an MIT-licensed dataset (Sec. 7.4).

Table -9: Data actually available.

Property	Value
Glosses in WLASL100	100
Glosses with at least one clip	67
Clips processed	402
Clips per populated gloss	1–10 (about 6)
Clip length (frames)	19–195 (mean 68.8)
Total frames (clips × mean length)	about 27,700
Train / val / test clips	321 / 40 / 41

5.2 Our own dataset

Apart from WLASL, our vocabulary was recorded using the recording utility in the repository. Our vocabulary comprises 100 words categorized into ten topics (Table -10). Each word has five sequences with 30 frames each, giving us 500 sequences and 15,000 frames in total, each frame being a 30x454 features matrix. The utility shows countdown and progress screens, and the signer can skip, replay or pause recording. We have created the prototype for recording 30 sequences of 30 frames each of three gestures (hello, thanks, iloveyou).

Table -10: Our own dataset: 100 words, 5 sequences per word, 30 frames per sequence.

Topic group	Words	Frames
Common signs	18	2,700
Home	6	900
Food	8	1,200
People	10	1,500
Questions	6	900
School	9	1,350
Time	11	1,650
Family	11	1,650
Colours and animals	9	1,350
Descriptions and conversation	12	1,800
Total	**100**	**15,000**

Accuracy is not included for this dataset. With 5 samples per word, a test split of 10 percent will give around 50 samples out of 100 words, where the samples taken in the same session are very similar to each other. Getting good accuracy using a split like this won't tell us anything about how well it performs with a new signer, since we haven't made a signer independent split yet..

5.3 Protocol and metrics

In WLASL100, clips are randomly partitioned with 80% for training, 10% for validation, and 10% for test under seed 42, resulting in 321, 40, and 41 clips. This split is not stratified by classes nor separated by signers. The validation set is employed for early stopping and checkpointing. Each of the accuracies listed below has been calculated by an evaluation script that recreates the above split, turns off data augmentation, and calculates top-1 and top-5 accuracies per split. Let there be N clips with labels y_i and class probabilities pi,

$$\mathrm{Acc@}k = (1/N)\,\Sigma_i\, 1[y_i \in \text{Top-}k(p_i)] \qquad (4)$$

With 67 active classes, chance level is 1.5% for top-1 and 7.5% for top-5. As the held-out sets are very small, we additionally give 95% Wilson score intervals for a proportion p measured on n clips, with $z = 1.96$,

$$p' \pm z\sqrt{(p(1-p)/n + z^2/4n^2)} \text{ over } (1 + z^2/n),\ p' = (p + z^2/2n)/(1 + z^2/n) \qquad (5)$$

We chose the Wilson interval over the plain normal approximation since it is always valid regardless of the sample size and even when p is near 0 or 1.

5.4 Measurement setup

Latency testing was performed on a laptop (see Table -11). Timing is performed using the script that runs the classifier with random data having the required format, and MediaPipe is timed using WLASL video frames resized to 640 x 480.

Table -11: Latency measurement setup.

Item	Value
Processor	Intel 11th-generation Core, Tiger Lake-H (laptop)
Accelerator	None; CPU only
Batch size, window	1, 30 frames
Timed runs	200 after warm-up
Models timed	PyTorch and ONNX Runtime

6. RESULTS AND DISCUSSION

6.1 Recognition accuracy

Results are provided in Table -12 and Fig -4. The model matches the training clips perfectly. For the test clips, it scores 69.0% for top-1 and 70.7% for top-5, which is roughly 26 times that of random guessing for top-1. The validation score is similar (45.0% and 87.5%). The difference in performance on training and validation data is the main point – the network memorized its training set, using about six clips per word.

Table -12: Accuracy by split, with 95% Wilson intervals.

Split	Clips	Top-1 (%)	Top-5 (%)
Train	321	100.0	100.0
Validation	40	45.0 (30.7–60.2)	87.5 (73.9–94.5)
Test	41	69.0 (54.3–70.9)	70.7 (55.5–82.4)
Chance	–	1.5	7.5

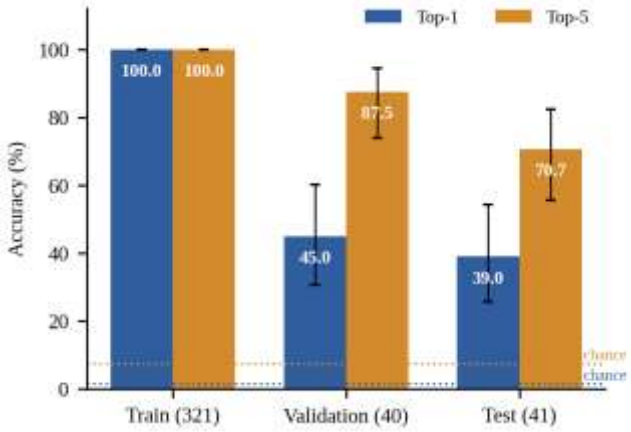


Fig -4: Top-1 and top-5 accuracy by split. Error bars are 95% Wilson intervals; dotted lines mark chance.

Top-5 is the more forgiving approach and the more meaningful one for a system which can present multiple candidates and allow the user to choose. Top-5 seems more promising compared to top-1: the right answer was among the five top-ranked words for around seven video samples out of ten. There is no confusion matrix yet, and so it is impossible to say what signs are being confused with each other.

6.2 How much can 41 clips tell us?

Very little, at least concerning the exact number. The 95% confidence interval for test top-1 accuracy ranges between 25.7% and 54.3%, i.e., it has a half-width of about 14 percentage points, and the same goes for the validation interval. Hence, the difference between 45.0% and 69.0% falls well within the noise range, and a change of one or two clips will likely revert it. Figure -5 demonstrates how fast this interval narrows down as the test set size increases. Given that we have 39% as a constant value, a half-width of 10 points implies that we need about 88 test clips, and a half-width of 5 points implies about 362. The total dataset size consists of 402 clips, so we need more clips or cross-validation for this vocabulary.

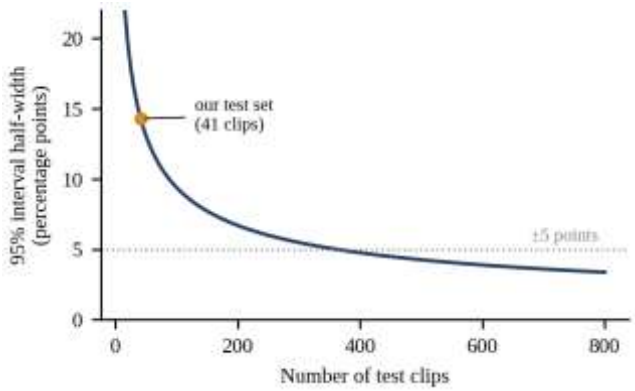


Fig -5: Width of the 95% Wilson interval for a measured accuracy of 39%, against the number of test clips.

6.3 Speed

Latency is found in Table -13, while Fig -6 presents its comparison with the time available per frame. Latency is mainly due to landmark extraction: MediaPipe Holistic requires 27-28 ms per frame while the classifier requires 1 ms or less with ONNX Runtime and 2-4 ms with PyTorch. The pipeline thus needs 30 ms to process a frame, which equals 32 frames per second. ONNX Runtime is 2.5 to 5 times faster than PyTorch and yields logits similar to those yielded by PyTorch up to 2.4 × 10−6.

Table -13: CPU latency per stage.

Stage	Mean (ms)	p95 (ms)
Classifier, ONNX Runtime	0.7–1.0	1.1–1.7
Classifier, PyTorch	2.3–3.5	3.0–5.1
MediaPipe Holistic	27–28	29–32
Full pipeline per frame	29–31	32–35

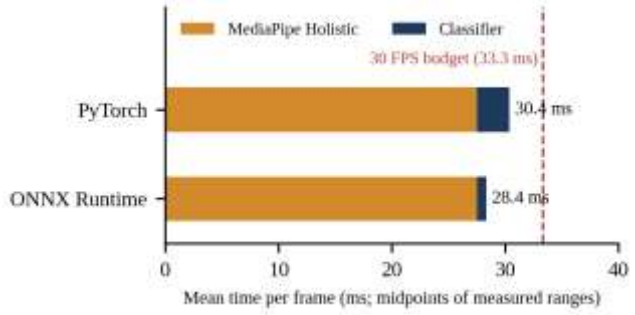


Fig -6: Mean time per frame compared with the 33.3 ms available at 30 FPS.

The 30 FPS camera gives the pipeline 33.3 ms per frame. The average time taken fits into this budget with some margin, although the 95th percentile of the pipeline (32-35 ms) is close to the deadline. Consequently, there will be some frames that come in too late on this laptop. The landmark extraction needs to get significantly faster, while the classifier can be improved less, because it is only a small part of the whole pipeline. There are two issues worth mentioning: the classifier timing assumes synthetic input, and we did not measure the lag from the end of the sign to the sentence appearance.

6.4 Model size

The classifier has 0.56 million parameters, an ONNX file size of 2.2 MB, and takes around a millisecond to execute. For a program that might run on low-end hardware one day, this is encouraging, but it simply demonstrates the efficiency of the algorithm, not its efficacy.

6.5 Threats to validity

Some features of the protocol increase the accuracy, while one feature decreases it (Table -14). The splitting is done by clip, so clips by the same signer or clips from the same video can go to either side of the split. The validation dataset is employed both for stopping training and choosing a checkpoint, so the validation metrics tend to be optimistic. Moreover, augmentation is performed on the validation clips, too, which means that early stopping becomes more noisy; the evaluation script does not do this. On the other hand, if there is a class with only one clip, it can go to just one split, and if this clip ends up in the test set, it cannot be classified.

Table -14: Threats to validity and their likely effect.

Threat	Effect	Remedy
Random split by clip; signers can overlap	Optimistic	Signer-independent split
Validation drives early stopping and selection	Optimistic (validation)	Keep a test set untouched; use cross-validation
Augmentation applied to validation clips in training	Noisier selection	Augment training clips only
Classes with a single clip	Pessimistic	Stratify, or drop classes with too few clips
Only 41 test clips	Wide interval	More clips; k-fold evaluation
Classifier timed on random input	Small	Time on real clips

6.6 What the results support

The authors of WLASL report around 66% of top-1 accuracy on WLASL100 [9] using I3D. Our 69.0% is inferior, yet the conditions are different as they use all 100 glosses and the official split, while we use only 67 glosses and our random split of 402 clips. We don't claim any comparison in accuracy. What our claim supports is the following: the classifier is compact, and the entire process works at the camera rate without a GPU.

There were no ablation studies conducted. The benefits we expect from separate streams, cross-modal attention, and velocity features are just design rationales rather than the results. We didn't evaluate the abilities of boundary detection, sentence creation, resistance to different lightings, backgrounds, or distances, or the performance on novel signers. The confidence gate and consensus vote were designed to limit false positives, but we don't have any measurements of its impact.

6.7 Limits of the approach

These limitations are an intrinsic part of the process used. As there are no face landmarks, it cannot see the markers for grammar in the eyebrows and mouth. It classifies words one at a time and requires a trained segmenter to handle the problem of continuous signing. Errors made in landmarking due to occlusions, motion blur, and crossing of hands are fed into the classifier, and no information can be retrieved from the landmarks. The vocabulary is limited to the glosses available in the training data, and the gloss layer is not a representation of the grammar of ASL. Lastly, the evaluation focuses on the technical aspect only, and there is nothing said about the usability and acceptance of the method.

6.8 Objectives against outcomes

Our project proposal listed more goals than this paper reports on, so it is fairer to state which of them the system meets. Table -15 sets them against the evidence we can show.

Table -15: Project objectives and where they stand.

Objective	Status	Evidence
Recognise signs in real time	Partly met	Isolated words at about 32 FPS on a CPU
Translate signs into text	Partly met	Rule-based gloss layer; not evaluated
Convert text to speech	Not built	Not described in the repository
Support multilingual output	Not built	Planned with a language-model service
Enable talk between signers and non-signers	Not tested	No user study yet
Achieve high recognition accuracy	Not yet	69.0% top-1 on held-out clips
Be affordable and portable	Partly met	CPU only; 0.56 million parameters
Provide a wearable device	Not built	See Sec. 8.2

The pattern is clear. The elements that depend only on a laptop, a webcam and a small model are in place; those that depend on more data, more hardware or real users are not.

6.9 Lessons from the experiments

There were three lessons learned. The first was that the data was more important than the model. A third of the vocabulary was lost since the videos had been deleted, and no architectural modification would have fixed it. When relying on publicly available video links for your project, it would be prudent to store all your downloads in a file when you collect it. Secondly, the evaluation script became indispensable since the training script reports metrics based on the validation set with augmentation applied, but only the script that rebuilds the split with augmentation switched off is valid for quoting the numbers. Thirdly, having a limited test set is indeed a constraint.

7. PRIVACY, FAIRNESS, LICENCES AND REPRODUCIBILITY

7.1 Privacy

Processing using the camera occurs locally. The frames are never saved nor sent, and after the landmark phase only the coordinates are used. This adheres to the intentions of the Digital Personal Data Protection Act, 2023 [15] but it does not offer privacy, as the landmark sequence still constitutes movement information from a recognizable individual. An actual implementation would need to get consent and a storage policy for sequences saved.

7.2 Fairness and intended use

The model trained on just a handful of signers may fail in its performance among other individuals, and our splitting of the training and testing sets cannot help us ascertain whether that happens. WLASL contains American Sign Language while the end users of the system developed in India will primarily be using Indian Sign Language. This means that ISL samples will be needed for deployment. Our system is a communication tool and not an alternative to the services of a professional translator, especially in matters that involve legal and medical procedures where the risk involved is much higher. No confidence gate can make a 39% classifier safe.

7.3 Reproducibility

This is where you can find the configuration files, the landmark pipeline, and the evaluation and export scripts. To reproduce the experiments yourself, a reader will need to download WLASL100 using the provided downloader, extract landmarks into one array per clip, train with the default configuration, and execute the evaluation script. There are two notes. Firstly, a large portion of WLASL links are broken, which

means that your download will produce a different amount of clips compared to 402 we have, which, in turn, leads to different results. Secondly, the ONNX export package needs a newer version of protobuf than what MediaPipe 0.10.14 works with.

7.4 Licences

Three types of licences are applicable in this project (Table –16). Our code is licensed using the MIT Licence. WLASL is not; it is available with C-UDA as per the repository's report itself, and the MIT Licence for our code does not apply to it. As the landmark arrays have been created using the WLASL videos, we assume that the same applies to them unless stated by the authors of the dataset. Our recordings, on the other hand, belong to the people who created them, and before releasing them, their authors must choose a licence.

Table -16: Licences and terms.

Item	Terms	Note
SignVision code	MIT	LICENSE file in the repository
WLASL videos and labels	C-UDA	Academic and computational use only
Landmarks derived from WLASL	Treated as C-UDA	Check before redistributing
Our own dataset	Authors' choice	MIT if released with the code

8. FUTURE WORK

8.1 Near-term fixes

A more robust evaluation is of foremost importance. Our intention is to retrieve more WLASL clips, use stratified and signer independent cross validation, add data only to the training set, and repeat ablations and architectures from Table -5 in Table -4. Only then will the accuracy numbers become meaningful. The next step would be the improvements indicated by our limitations in Section 6.7, Table -17.

Table -17: Planned work.

Horizon	Work item	Purpose
Near	More clips; stratified, signer-independent k-fold	Trustworthy accuracy
Near	Ablations and alternative architectures	Test design claims
Medium	About 40 lip and eyebrow landmarks	Non-manual markers
Medium	Pre-training on larger pose datasets; graph encoders	Less data hunger
Medium	Trained segmenter; T5 gloss-to-text	Continuous signing, fluent text
Longer	Indian Sign Language data such as iSign [8]	Intended users
Longer	Wearable version (Sec. 8.2)	Hands-free use

8.2 A wearable version

Another project proposal related to the same concept involves designing such a system with smart goggles, including a camera attached to the frame, a mini computer such as Raspberry Pi 5 or ESP32-S3 board, micro OLED screen, speech generation using text-to-speech library, and a language modeling service that can generate sentences in other languages. This is not included in the described system, but our findings suggest the challenges that may be encountered..

First, there is viewpoint. WLASL videos are viewed from the front of the signer. A camera mounted on the wearer's glasses captures the signer's hands in a top-down view from behind and not showing the face. What is being signed is captured rather than the signing of the conversation partner. Frontal viewpoints will not transfer into a classifier that has been trained on such a viewpoint, so a wearable system will require its own training data. Second, there is computation. MediaPipe Holistic requires 27-28 ms per frame on a laptop CPU, and this has not been timed on a Raspberry Pi. ESP32-S3 is a microcontroller and would never run Holistic; it might work better as a camera that can stream frames to a phone or small computer. Third, there is privacy. The transmission of gloss text to a language-model service is quite distinct from transmitting video, and the design should ensure that only the former occurs. Speech and multilingual output are, by contrast, the easy part.

9. CONCLUSIONS

SignVision processes webcam frames to predict ASL words based on landmarks, a two-stream attention classifier, and a subtitles user interface. It processes at an estimated 32 frames per second on a laptop CPU. With respect to the 67 glosses for which we obtained results, it has a top-1 accuracy of 69.0% and a top-5 accuracy of 70.7% on held-out clips. With only 41 clips and a random split available, this is our preliminary guess, with a reasonable top-1 accuracy range of 26% to 54%.

Our contribution is a small, inspectable, and privacy-respectful pipeline from camera frames to subtitles, together with an upfront discussion of its limitations. The difference between perfect training accuracy and low test accuracy means that data, not architecture, is limiting here. The steps that need to be taken first are improved evaluation and data collection because all of Sec. 8 relies on them.

ACKNOWLEDGEMENT

We acknowledge Dr. (Mrs.) A. V. Dehankar, Head of the Computer Technology Department, Dr. G. M. Asutkar, Vice Principal, and Dr. S. A. Dhale, Principal of Priyadarshini College of Engineering, Nagpur, for their assistance and facilities provided during this project.

REFERENCES

1. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. In: Proc. USENIX Symp. Oper. Syst. Des. Implement. (OSDI), pp. 265–283 (2016)
2. Bradski, G.: The OpenCV library. Dr. Dobb's J. Softw. Tools (2000)
3. Camgoz, N.C., Koller, O., Hadfield, S., Bowden, R.: Sign language transformers: joint end-to-end sign language recognition and translation. In: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 10023–10033 (2020)
4. Carreira, J., Zisserman, A.: Quo vadis, action recognition? A new model and the Kinetics dataset. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 6299–6308 (2017)
5. Cheok, M.J., Omar, Z., Jaward, M.H.: A review of hand gesture and sign language recognition techniques. Int. J. Mach. Learn. Cybern. 10(1), 131–153 (2019)
6. Cui, Z., Zhang, W., Li, Z., Wang, Z.: Spatial–temporal transformer for end-to-end sign language recognition. Complex Intell. Syst. (2023)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997)
8. Joshi, A., et al.: iSign: a benchmark for Indian sign language processing. arXiv:2407.05404 (2024)
9. Li, D., Rodriguez, C., Yu, X., Li, H.: Word-level deep sign language recognition from video: a new large-scale dataset and methods comparison. In: Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), pp. 1459–1469 (2020)
10. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Int. Conf. Learn. Represent. (ICLR) (2019)
11. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: Int. Conf. Learn. Represent. (ICLR) (2017)

12. Lugaresi, C., et al.: MediaPipe: a framework for building perception pipelines. arXiv:1906.08172 (2019)
13. Micikevicius, P., et al.: Mixed precision training. In: Int. Conf. Learn. Represent. (ICLR) (2018)
14. Ong, S.C., Ranganath, S.: Automatic sign language analysis: a survey and the future beyond lexical meaning. IEEE Trans. Pattern Anal. Mach. Intell. 27(6), 873–891 (2005)
15. Parliament of India: The Digital Personal Data Protection Act, 2023. Act No. 22 of 2023
16. Parliament of India: The Rights of Persons with Disabilities Act, 2016. Act No. 49 of 2016
17. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Adv. Neural Inf. Process. Syst. (NeurIPS), pp. 8024–8035 (2019)
18. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. 21(140), 1–67 (2020)
19. Rastgoo, R., Kiani, K., Escalera, S.: Sign language recognition: a deep survey. Expert Syst. Appl. 164, 113794 (2021)
20. Rawat, P., Kumar, P., Tamta, V.K., Kumar, A.: A comprehensive approach to Indian sign language recognition: leveraging LSTM and MediaPipe Holistic for dynamic and static hand gesture recognition. EAI Endorsed Trans. AI Robot. (2025)
21. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45(11), 2673–2681 (1997)
22. Subramanian, B., Olimov, B., Naik, S.M., Kim, S., Park, K.-H., Kim, J.: An integrated MediaPipe-optimized GRU model for Indian sign language recognition. Sci. Rep. 12, 11964 (2022)
23. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception architecture for computer vision. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 2818–2826 (2016)
24. Vaswani, A., et al.: Attention is all you need. In: Adv. Neural Inf. Process. Syst. (NeurIPS), pp. 5998–6008 (2017)
25. Velmathi, G., Goyal, K.: Indian sign language recognition using MediaPipe Holistic. arXiv:2304.10256 (2023)
26. Wolf, T., et al.: Transformers: state-of-the-art natural language processing. In: Proc. Conf. Empirical Methods Nat. Lang. Process.: Syst. Demonstrations, pp. 38–45 (2020)
27. World Health Organization: World Report on Hearing. WHO, Geneva (2021)
28. Zhang, F., et al.: MediaPipe Hands: on-device real-time hand tracking. arXiv:2006.10214 (2020)