WMSM: An Efficient Real-Time Framework for Hebrew Sign Language Recognition and Sentence-Level Translation

Eyal Pasha
College of Management Academic Studies
Faculty of Computer Science
Rishon LeZion, Israel
eyalpasha115@cs.colman.ac.il

Dr. Galit Haim
College of Management Academic Studies
Faculty of Computer Science
Rishon LeZion, Israel
galitha@colman.ac.il

***Abstract*—This paper introduces WMSM (Word Model, Sentence Mechanism), an innovative deep learning approach for real-time Hebrew Sign Language (HSL) recognition, optimized for mobile devices. WMSM features a novel word-level recognition model combined with a sentence-level prediction mechanism, offering an efficient alternative to traditional sequence-to-sequence (Seq2Seq) frameworks. Sign language recognition presents significant challenges, including limited dataset diversity, overfitting, and the difficulty of generalizing across different signers with varying signing styles. These issues are compounded by computational constraints, particularly when deploying models on mobile devices. To address these challenges, the paper details an augmentation pipeline that artificially expands the dataset, enhancing variability and enabling the model to generalize more effectively. Additionally, computational challenges such as packet loss and processing delays were mitigated through optimized model efficiency and the use of CPU-based workflows. The system demonstrates robust translation capabilities, producing coherent sentence-level outputs even under challenging conditions. This approach sets a new benchmark for efficient, high-performance real-time sign language recognition, addressing key challenges in dataset limitations, generalization, and mobile deployment.**

***Index Terms*—Sign language recognition, Sign language translation, Deep learning, Efficient NLP, Hebrew Sign Language, Data augmentation, Human-Centered NLP, Machine Translation**

I. Introduction

Sign language serves as a vital means of communication for Deaf and hard-of-hearing individuals. HSL is a key language used by Deaf communities in specific regions, yet despite its critical role, technology to recognize HSL still lags behind. This delay perpetuates communication barriers between Deaf and hearing individuals.

A. Advancements in Machine Learning and the Need for HSL Research

Advances in machine learning, especially in computer vision and natural language processing, have enabled the development of sign language recognition and translation (SLRT) systems. These systems can translate sign language into text or speech in real time, with significant progress made for widely used sign languages, such as American Sign Language (ASL).

Li et al. [1] developed an SLR system using the Word-Level American Sign Language (WLASL) dataset, which includes 2,000 words and 21,000 videos from 119 signers. Their method combines holistic visual appearance-based models with 2D pose-based techniques, achieving 83.1% top-1 accuracy and 89.5% top-5 accuracy. Despite these advancements, the authors do not detail real-world deployment or practical applications, likely due to the models' size, complexity, and computational demands.

Less-studied sign languages, such as HSL, remain largely unexplored, highlighting a significant gap in SLRT research. Broader studies could deepen our understanding of the field and provide insights into effective recognition techniques across diverse linguistic and cultural contexts.

This paper introduces WMSM, an efficient and real-time HSL recognition system designed for mobile deployment. It addresses the lack of research and practical solutions in this area by focusing on augmentation, model compression, and optimizations. Additionally, it establishes a foundation for practical HSL applications and offers scalability and adaptability to other sign languages, fostering broader adoption of SLRT technologies.

II. Related Work

SLRT faces challenges such as limited data, computational constraints, and capturing linguistic nuances. Recent works have explored segmentation techniques, attention mechanisms, data augmentation, and efficient architectures. Building on these, the proposed WMSM addresses key limitations while optimizing for real-time performance and practical deployment.

Rust et al. [2] introduced SSVP-SLT, a framework combining self-supervised pretraining with masked autoencoding [3] and supervised fine-tuning. Their hierarchical transformer model, SignHiera, processes long video clips of up to 128 frames and achieved a BLEU-4 [4] score of 15.5 on the How2Sign dataset [5], which is a moderate result for large-scale SLRT tasks. Although their approach addressed privacy concerns by blurring facial features, we opted to focus

exclusively on hand landmarks. This reduces data complexity and computational demands, making our model ideal for real-time applications and edge devices.

Zhang et al. [6] presented a multilingual SLRT framework that combines SLT with machine translation (MT). This framework achieved a BLEU-4 score of 21.06 on the How2Sign dataset, reflecting state-of-the-art performance under large-scale multilingual transfer. In contrast, WMSM employs a specialized methodology that reduces computational overhead and avoids the complexities of multilingual MT systems, while still delivering effective and competitive results.

Gueuwou et al. [7] introduced SignMusketeers, a multi-stream SLT framework that processes hand, face, and body posture features to capture linguistic nuances. By utilizing pose estimators and DINOv2 [8] for static features, they significantly reduced computational costs. The framework achieved a BLEU-4 score of 14.3 on How2Sign, a moderate outcome given the complexity of SLRT, while using only 3% of the resources required by Rust [2]. Their finding that temporal information is unnecessary during pretraining supports our decision to treat frames independently during preprocessing and augmentation. While they encode face and body features, we focus solely on hand landmarks, simplifying the process and enhancing scalability for real-world applications.

These studies highlight diverse approaches to SLRT, ranging from multilingual frameworks to multi-stream architectures. Drawing from these insights, we developed the WMSM framework, a streamlined system that integrates advanced augmentation techniques, efficient sequence modeling, and real-time adaptability. The WMSM framework emphasizes practical deployment, ensuring scalability and usability in real-world scenarios.

III. Methodology

A. Data Augmentation

1) Initial Dataset Overview: The initial HSL dataset [9] consisted of 2,342 short videos, with one video per word in HSL. Each video represents a single gesture corresponding to a specific word (see Appendix A for terminology).

2) The Problem: One of the key challenges in sign language recognition is generalization. Models often struggle to perform consistently across different signers, and this problem is made worse by our very limited dataset, as noted earlier. A small dataset restricts the diversity of signing styles, leading to overfitting, where models perform well on training data but fail on new inputs. This lack of variety is a major obstacle, as sign language systems need exposure to diverse gestures and conditions to generalize effectively. To address this, we introduced an innovative augmentation pipeline to artificially expand the dataset and introduce greater variability.

However, other significant challenges persisted. Packet loss during processing was a recurring issue with GPU-based workflows, especially during extended runs, leading to data corruption and delays.

Managing compute resources and processing time presented a significant challenge. The preprocessing pipeline demanded substantial computational effort due to the large volume of augmented data. Balancing the workload with the requirement for high-quality, diverse data proved particularly difficult, highlighting the complexities of scaling data preparation processes effectively.

3) Augmentation Strategy: To expand the dataset and improve generalization, we adopted a three-stage augmentation strategy: an initial controlled step (Section III-A4), a main phase of extensive transformations (Section III-A5), and an on-the-fly procedure during training (Section III-B3). These steps enrich data diversity, addressing the challenges posed by the limited dataset.

One key idea was to skip temporal information in the initial stages, simplifying preprocessing by enabling frame-by-frame augmentation without considering the sequence. This focused the pipeline on spatial features like hand shapes and positions before adding temporal dynamics in later stages.

To avoid data corruption and packet loss during preprocessing, the preprocessing pipeline was run on CPUs. This ensured data integrity, taking 24 hours for the initial phase and 72 hours for the main phase. With more powerful CPUs, the dataset can be scaled substantially. Details of the hardware and software configurations appear in Appendix B. This methodology lays the groundwork for future advancements in sign language recognition, with the potential to support a wide range of applications in accessible communication technologies.

4) Initial Augmentation: The initial phase increased dataset diversity by applying transformations to entire videos, not individual frames. Frame-based transformations were reserved for later stages. Controlled operations included rotation, translation, zoom, and shear (skewing along one axis). Rotation had a 50% probability with $\theta \in [-5°, 5°]$, translations shifted $t_x, t_y \in [-5, 5]$ pixels, zoom adjusted $z \in [0.98, 1.02]$, and shear added $s \in [-0.05, 0.05]$. The initial ranges were heuristically chosen to reflect natural variations in hand pose and camera perspective, such as minor rotations and shifts during real signing. They were later refined through visual inspection to ensure gesture integrity.

Additional visual augmentations—brightness, contrast, color manipulation, Gaussian blur, noise, cutout, motion blur, edge detection, grayscale, inversion, flipping, and crop-resize—further diversified the dataset, all applied at the video level to preserve temporal coherence. Figure 1 shows an augmented frame with transformations such as brightness, contrast, and rotation adjustments.

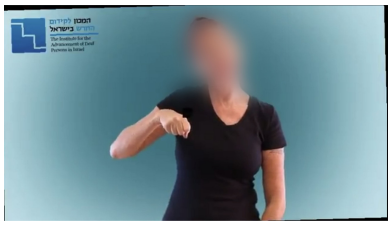

Fig. 1. Augmented Frame.

To balance hand dominance, 11% of samples were flipped to represent left-handed signers, addressing the original dataset's lack of left-handed examples. Each video was labeled with its gesture (e.g., the action for the Hebrew word "cat" was labeled as "cat.mp4"), and ten augmentations per original video produced $25,762$ videos. We personally observed most samples to ensure data integrity and minimize artifacts affecting training.

5) Main Augmentation: The main augmentation phase introduced extensive transformations at both the frame and landmark levels. Up to 40 frames were extracted per video and standardized, while shorter ones were padded. Frame augmentations included isotropic scaling, brightness, and contrast adjustments to mimic diverse lighting conditions. Using MediaPipe's Hand Tracking [10], we captured the spatial dynamics of hand gestures while preserving dataset consistency.

Hand landmarks were extracted using a detection confidence of 0.6 and a tracking confidence of 0.5, optimized to balance accuracy and efficiency. MediaPipe detected up to two hands per frame, generating 21 landmarks per hand. These landmarks, comprising normalized 2D coordinates and depth relative to the hand's centroid, captured essential hand movements while simplifying input to focus on subtle gestures. An example of this process is shown in Figure 2, which illustrates a regular frame (top) and the detected hand landmarks drawn on the same frame (bottom).

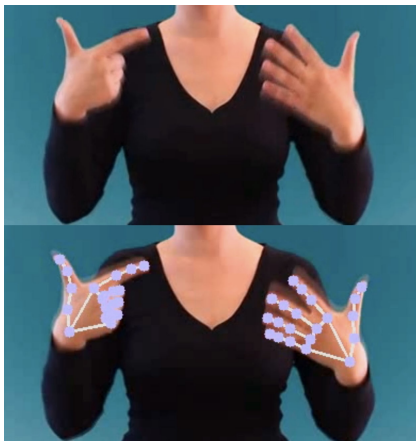

Fig. 2. Detection Example.

Ten augmented versions per video were generated, enriching the dataset while maintaining a consistent input shape. Normalized sequences were stored with gesture labels as numerical classes and frame counts in a structured format. Sequences were standardized to 40 frames, excluding videos with fewer than 10 valid frames to ensure quality.

Skipping temporal information in the initial stages streamlined preprocessing by enabling frame-by-frame augmentation without considering the sequence. This simplification reduced both complexity and processing time, allowing the pipeline to focus on general spatial features like hand shapes and positions.

a) Final Dataset - : The final dataset comprised $252,712$ samples spanning $2,342$ unique gesture classes.

B. The Model

1) Model Architecture: The single-gesture model serves as the foundational *word-level* component of WMSM. It incorporates several key elements, including TimeDistributed one-dimensional Convolutional Neural Networks (Conv1D [11]), Bidirectional Long Short-Term Memory (Bi-LSTM [12]) networks, a custom attention mechanism, and a custom activation function for confidence scaling. It also features a carefully designed multi-term loss function that balances accuracy, confidence, and temporal consistency. Each component plays a distinct role in processing sequences of hand landmarks from video frames.

a) TimeDistributed Conv1D Layers - : The model begins with TimeDistributed Conv1D layers, each configured with 512 filters and a kernel size of 3. The filter count provides ample capacity to learn nuanced spatial features, while the kernel size was chosen to effectively capture local relationships between neighboring hand landmarks—typically spaced closely in sequential frames. This setting was informed by domain-specific considerations and refined through empirical analysis. ReLU is used as the activation function [13].

b) Rationale for Using Conv1D Over Conv2D - : During experimentation, we found that depth information (the z-coordinate) was not crucial for recognizing most gestures. As a result, it is excluded from this part of the model but remains utilized in the augmentation pipeline. The key factor was the spatial arrangement and relationships between the landmarks in the x-y plane—the sequential patterns formed by connecting the points. This observation guided our focus on Conv1D layers, which are specifically suited for capturing such patterns. Although hand landmarks are three-dimensional (x, y, z), we found that the defining features of hand shapes are primarily determined by their two-dimensional arrangement (x and y coordinates). Most gestures retain their unique characteristics regardless of depth variations, as the relative positions of landmarks in the x-y plane remain consistent. Depth changes had little impact on the essential shapes required for accurate gesture recognition. This conclusion contrasts with the initial assumption that 3D information is necessary for accurate recognition. By treating the sequence of landmarks within each frame as a one-dimensional array, Conv1D layers effectively capture the spatial relationships among landmarks without the added complexity of depth.

c) Batch Normalization and MaxPooling Layers - : Batch Normalization layers [14] are applied twice. The first normalization follows the TimeDistributed Conv1D layers, stabilizing activations, reducing covariate shift, and enabling higher learning rates. Following this, TimeDistributed MaxPooling1D [11] layers are used to downsample the output, retaining essential features while reducing dimensionality. A second Batch Normalization layer follows MaxPooling, stabilizing the feature space before the attention layer.

d) Attention Layer - : Following those applications and pooling operations, the output is passed through the attention layer. Refer to Section III-B2.

e) Flattening Layer - : The output from the attention layer is passed through a TimeDistributed Flattening operation, which flattens the output of each time step independently. This results in a two-dimensional tensor suitable for sequence processing while preserving the temporal structure of the input sequence.

f) Bidirectional LSTM Layer - : A Bidirectional LSTM layer is used to process the sequences, capturing the temporal dependencies essential for sign language gestures. This yields temporal feature embeddings that represent the motion and order of gestures, as shown in Equation 1:

$$\overleftrightarrow{h}_t = \text{LSTM}\big(\overleftrightarrow{h}_{t-1}, \mathbf{y}_t\big). \tag{1}$$

Here, $\mathbf{y}_t$ is the input at time step t, $\overleftrightarrow{h}_{t-1}$ is the previous hidden state, and $\overleftrightarrow{h}_t$ represents the updated hidden state.

g) Fully Connected Layers and Output - : Following the LSTM layer, a Dense layer with 512 units and ReLU activation further processes the features. A 0.5 dropout rate reduces overfitting. The final Dense layer has $2,342$ units, using CustomConfidenceActivation III-B4 for class probability distribution. Also, refer to Appendix D for a layers type breakdown.

2) Attention Layers: To focus on the most informative frames, a custom attention mechanism is introduced. The attention mechanism first applies a linear transformation followed by a non-linear activation ($\tanh$) to each time step, as shown in Equation 2:

$$v_t = \tanh\big(P_t W + b\big), \tag{2}$$

where P_t is the input vector at time step t for the attention mechanism, W is a trainable weight matrix initialized using Glorot uniform initialization [15], and b is a bias term initialized to zeros.

Next, the transformed vectors v_t are projected onto a learnable vector q to obtain scalar logits r_t. These logits are scaled by scale $= 5.0$ and normalized via the exponential function. Numerical stability is ensured by subtracting the sequence's maximum value before exponentiation. The attention weights β_t are computed as shown in Equation 3:

$$\beta_t = \frac{\exp\big(\text{scale} \cdot r_t - \max_k(r_k)\big)}{\sum_{j=1}^{T} \exp\big(\text{scale} \cdot r_j - \max_k(r_k)\big)}, \tag{3}$$

where r_t is the scalar logit for time step t, $\max_k(r_k)$ represents the maximum logit value across all time steps $k \in [1, T]$, and T is the total number of time steps. This formulation ensures that larger logits r_t receive proportionally higher weights while preventing numerical overflow during computation.

Finally, the layer computes a weighted sum of the transformed inputs P_t using the attention weights β_t, as described in Equation 4:

$$c = \sum_{t=1}^{T} \beta_t P_t, \tag{4}$$

where c is the context vector summarizing the sequence, β_t is the attention weight for time step t, P_t is the transformed input vector at time step t, and T is the total number of time steps. This operation directs the model to focus on frames most relevant to the task, assigning higher weights to time steps with greater predictive importance.

The attention layer introduces a small number of additional trainable parameters—W, b, and q—and leverages efficient tensor operations to keep computational overhead minimal.

This custom mechanism was chosen for its simplicity, efficiency, and strong performance, aligning with the framework's real-time, lightweight design goals while avoiding the complexity of multi-head attention.

3) On-The-Fly Landmark Augmentation: The final augmentation stage operated on hand landmark coordinates during training, introducing temporal deformations to mimic signing tempo variations and enable the model to handle varying gesture speeds. Next, the geometric center was estimated using the average position of the wrist and palm base landmarks. This palm-centric approach allowed for a natural pivot point, around which a small rotation angle $\theta \sim \mathcal{U}(-0.1, 0.1)$ radians (±6 degrees) was applied, reflecting realistic wrist motion without distortions. Small translations and scaling simulated natural hand movement and size variations. These added shifts and subtle adjustments, reflecting posture or position changes.

To enhance realism, Gaussian noise was added to finger (x, y)-coordinates, simulating natural tremors and sensor imprecision. This helped the model adapt to minor variations in finger placement effectively. Figure 3 shows original landmarks (top) and their augmented counterparts (bottom), highlighting transformations such as rotations, translations, and added Gaussian noise to simulate realistic hand configurations.

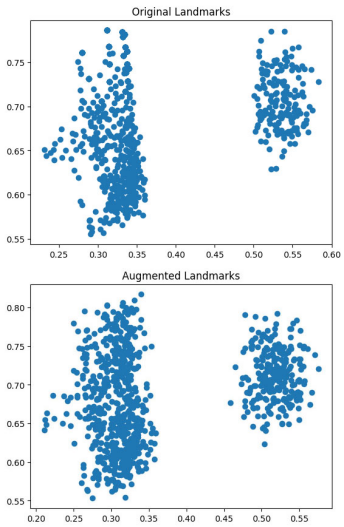

Fig. 3. Original and Augmented Landmarks.

By incorporating timing refinements, palm-centered rotations, translations, scaling, finger perturbations, and controlled noise, the pipeline introduced diverse and realistic hand configurations.

4) Custom Loss and Activation Functions:

a) Custom Confidence Activation - : This layer scales the logits by a scale factor (scale $= 5.0$) before normalizing them into class probabilities. The scaling factor was empirically chosen to enhance class contrast in fine-grained gesture recognition without causing instability. By amplifying the differences between high and low logits, the network produces more definitive classifications for subtle hand poses and motions.

Specifically, let z_i be the logit for class i. Each logit is scaled by 5.0, and the maximum scaled logit is subtracted for stability, as shown in Equation 5:

$$\tilde{z}_i = 5.0 \times z_i - \max_k\big(5.0 \times z_k\big), \tag{5}$$

where $\max_k\big(5.0 \times z_k\big)$ represents the maximum scaled logit across all classes k. The probabilities $\hat{p}_i$ are then obtained by exponentiating and normalizing these adjusted logits, as defined in Equation 6:

$$\hat{p}_i = \frac{\exp(\tilde{z}_i)}{\sum_j \exp(\tilde{z}_j)}. \tag{6}$$

Here, $\hat{p}_i$ is the predicted probability for class i, ensuring that the model gives higher confidence to its most likely predictions.

b) Custom Loss Function - : The custom loss function combines three components, each weighted based on its importance for gesture recognition:

1) **Cross-Entropy Loss (CE), $\lambda_1 = 1.0$:** A standard categorical cross-entropy driving correct classification, as shown in Equation 7:

$$\mathcal{L}_{\text{CE}} = -\sum_i y_i \, \log\big(\hat{p}_i\big), \tag{7}$$

where y_i is the true one-hot label for class i, and $\hat{p}_i$ is the predicted probability for class i. Setting $\lambda_1 = 1.0$ fully integrates this objective.

2) **Confidence Maximization (CM), $\lambda_2 = 1.0$:** This term favors peaked distributions, penalizing uncertain predictions and encouraging the model to commit to a single gesture interpretation, as described in Equation 8:

$$\mathcal{L}_{\text{CM}} = -\sum_i \hat{p}_i \, \log\big(\hat{p}_i + 10^{-10}\big). \tag{8}$$

The small constant 10^{-10} avoids undefined logarithms. A weight of $\lambda_2 = 1.0$ balances this with cross-entropy.

3) **Temporal Consistency (TC), $\lambda_3 = 0.5$:** This regularizer discourages large fluctuations in predictions across frames, smoothing transitions in gestures while allowing responsiveness to real changes, as defined in Equation 9:

$$\mathcal{L}_{\text{TC}} = \sum_{t=2}^{T} \sum_i \Big(\hat{p}_{t,i} - \hat{p}_{t-1,i}\Big)^2. \tag{9}$$

Here, $\hat{p}_{t,i}$ and $\hat{p}_{t-1,i}$ represent the predicted probabilities for class i at time steps t and $t-1$, respectively, and T is the total number of time steps. Weighting it by $\lambda_3 = 0.5$ balances smoothness and adaptability.

The total loss is a weighted sum of these components, as shown in Equation 10:

$$\mathcal{L} = \lambda_1 \, \mathcal{L}_{\text{CE}} \; + \; \lambda_2 \, \mathcal{L}_{\text{CM}} \; + \; \lambda_3 \, \mathcal{L}_{\text{TC}}, \tag{10}$$

where $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, and $\lambda_3 = 0.5$. These coefficients reflect an empirical balance among accurate classification, high-confidence outputs, and the inherent smoothness of gestural sequences.

5) Dual Validation Strategy: To evaluate the model and monitor its generalization, a dual validation strategy was employed. This involved validating on both unaugmented and augmented data at the end of each epoch. Unaugmented validation assessed standard performance, while augmented validation measured generalization to data variations. Validation loss and accuracy were logged for both datasets, providing detailed feedback. This strategy helped the model balance accuracy across both data types and served as a fallback when on-the-fly augmentations introduced distortions, ultimately enhancing robustness and generalization.

6) Training Strategy: The model's performance was evaluated using three metrics: accuracy, which measures correct predictions; GAP [16] for class ranking; and MSE [17] for prediction error magnitude. Training was conducted with the Adam optimizer, gradient clipping, and a scheduled learning rate. Regularization techniques such as EarlyStopping and ReduceLROnPlateau ensured convergence and prevented overfitting. Detailed training configurations are provided in Appendix E.

C. Prediction Mechanism

To extend the single-gesture prediction model to sentence-level predictions and complete the WMSM framework, we developed a sentence prediction mechanism to avoid the complexity of traditional sequence-to-sequence models. This design ensures efficiency, making the system practical for real-world applications. The approach feeds overlapping segments of input landmarks into the single-gesture model (the "word" component) and aggregates the results through a "sentence-level" mechanism. This process ensures temporal consistency and accurate sentence generation within a simple computational framework. The prediction process consists of the following steps:

1) **Input Segmentation**: The landmark sequence is split into 40-frame segments with a 5-frame overlap.
2) **Gesture-Level Predictions**: Each segment is analyzed by the single-gesture model, yielding a gesture prediction and confidence score.
3) **Consensus-Based Temporal Aggregation**: Predictions from the latest 5 segments are stored in a sliding window, identifying the most frequent gesture and its average confidence score. A gesture is deemed reliable if it appears at least 3 times ($x\_predictions = 3$), or 60% of the window, balancing stability and responsiveness. Experimentation showed $x\_predictions = 3$ minimizes noise sensitivity while maintaining prompt responses.

Lower thresholds increased noise, while higher thresholds reduced responsiveness.

4) **Dynamic Confidence Thresholding**: A base confidence threshold of 0.65 decides if the consensus gesture is added to the sentence, with adjustments including:
   - **Repetition Count**: Threshold increases by 0.05 for each repeated gesture, reducing overprediction.
   - **Overpredicted Gestures**: For prone gestures, stricter criteria raise the confidence threshold by 0.25 and require 4 occurrences ($x\_predictions = 4$).

This mechanism bridges gesture-level predictions to sentence-level outputs, completing the WMSM framework with a lightweight, adaptable design. Dynamic thresholds and aggregation enhance accuracy in noisy conditions, while avoiding seq2seq overhead enables real-time performance on edge devices.

IV. Results

The WMSM framework achieves state-of-the-art performance in recognizing and translating HSL, particularly excelling in sentence-level tasks even under challenging conditions. Evaluations were conducted using $1,350$ curated videos (2–7 words each), generated via a custom sentence-building mechanism that combines single-word clips from the test set into smooth, coherent sequences. These videos were captured at 40 FPS, introducing additional complexity to gesture detection and translation accuracy.

For evaluation, we used NLTK for BLEU-1 to BLEU-4 scores with smoothing and SacreBLEU for corpus-level BLEU to ensure standardized, reproducible reporting. Despite the complexity of the task, WMSM achieved a BLEU-4 score of 21.15, with BLEU-1, BLEU-2, and BLEU-3 scores of 55.66, 41.32, and 30.16, respectively. A SacreBLEU score of 25.40 confirmed system robustness.

At the word level, WMSM's recognition module demonstrated strong standalone performance, achieving 97.29% accuracy, 97.57% precision, 97.24% F1-score, and a remarkable Top-3 accuracy of 99.72%.

The system's computational efficiency adds to these impressive outcomes. Data augmentation took 72 hours on a CPU, and model training was completed in only 24 hours on a single GPU, demonstrating both scalability and practicality. Full BLEU score analysis under confidence thresholds appears in Appendix F.

Table I offers a comparative view of BLEU-4 scores across various SLRT frameworks. Zhang [6], reached 21.06 on How2Sign using an MT-based strategy, while Rust [2] reported 15.5 with pre-trained features, Gueuwou [7] obtained 14.3 via a multi-stream method, and Uthus [18] achieved 12.4 using pose estimation. In contrast, WMSM recorded 21.15 on HSL, demonstrating a similarly high level of translation quality with a lightweight, real-time architecture. This result is especially notable given HSL's relatively unexplored status, underscoring WMSM's adaptability and efficiency.

As shown in the Dataset column, Rust [2], Gueuwou [7], and Uthus [18] methods utilized additional datasets like YouTube-ASL in combination with How2Sign, with Uthus [18] leveraging over $1,000$ hours of content. In contrast, WMSM was trained on $2,342$ single-gesture videos (augmented to $252,712$) and validated on $1,350$ sentence-level videos, yet achieved strong results. This highlights the method's adaptability and precision.

A key distinction of WMSM is its efficiency. Rust [2] required $32,000$ GPU-hours, and Gueuwou [7] used 880 GPU-hours. In contrast, WMSM completed training in just 24 GPU-hours and augmentation in 72 CPU-hours, reducing computational costs by 99.93% compared to Rust [2], while also achieving higher BLEU scores.

Taken together, these findings underscore WMSM's strengths of high accuracy, adaptability, and efficiency, even under demanding conditions. By balancing computational demands with superior performance, WMSM represents a significant step toward practical, scalable SLT solutions.

V. Conclusion

This paper presents the WMSM framework, a word-level recognition model coupled with a sentence-level mechanism, offering a streamlined alternative to traditional Seq2Seq frameworks for SLT, specifically designed for HSL. By leveraging the x-y plane of hand landmarks, WMSM demonstrates that 3D depth (z-coordinate) is unnecessary, as Conv1D layers effectively capture spatial patterns while reducing complexity. A three-phase augmentation strategy introduces variability at the video, frame, and landmark levels through transformations such as scaling, translations, rotations, and brightness adjustments, preserving temporal and spatial coherence. Trained on $252{,}712$ gestures and evaluated on $1{,}350$ sentence-level videos, WMSM achieves state-of-the-art performance, demonstrating both scalability and adaptability. These findings represent meaningful progress toward efficient, practical sign language recognition systems for real-world deployment.

VI. Future Work

Future work will prioritize significantly expanding and diversifying the dataset through recordings of authentic HSL signers, emphasizing varied signing styles, speeds, and contexts. Community-driven initiatives will facilitate the creation of comprehensive datasets essential for robust generalization and real-world applicability. Furthermore, future research aims to develop a bidirectional system supporting text-to-sign and speech-to-sign translations, fostering seamless, real-time communication between signers and non-signers. These efforts, combined with active community engagement, will help bridge accessibility gaps and promote inclusive communication.

A key focus of ongoing development is Handibur, a live video chat app currently being developed in collaboration with two colleagues. Handibur combines the functionality of modern social platforms with a groundbreaking feature: the integration of our model and mechanism directly into the video chat interface. This feature aims to enable real-time translation

TABLE I
COMPARISON OF BLEU SCORES ACROSS METHODS AND DATASETS

Method	Dataset	BLEU-1	BLEU-2	BLEU-3	BLEU-4	GPU-Hours
WMSM (Proposed)	HSL	**55.66**	**41.32**	**30.16**	**21.15**	**24**
Zhang (MT) [6]	H2S	N/A	N/A	N/A	21.06	N/A
Rust (SSVP-SLT, pre-trained) [2]	SSVP-SLT-LSP YT+H2S 600→200	43.2	28.8	20.8	15.5	32912*
	YT(50) 800 YT→H2S	41.9	27.7	19.8	14.7	18768*
SignMusketeers (MS) [7]	YT(1.2) 5 YT→H2S	41.5	27.2	19.3	14.3	880
Uthus (pose estimation) [18]	YT→H2S	37.8	24.1	16.9	12.4	N/A
	YT + H2S	36.3	23.0	16.1	11.9	N/A
Tarrés [19]	H2S	35.2	20.62	13.25	8.89	N/A
	H2S	34.0	19.3	12.2	8.03	N/A
Lin [20]	H2S	21.78	13.35	9.61	7.51	N/A

The table shows the top H2S results for each paper from the sources. BLEU-4 scores were unavailable for some methods.
* GPU hours for Rust et al. were calculated using Section C.3 of their paper, as reported by Gueuwou [7].
† While cross-language BLEU comparisons are imperfect due to differences in structure, we include them to contextualize performance relative to prior work. Our results reflect the best available evaluations under current constraints.

of HSL into text using a modified version of the sentence mechanism. Now in beta on Apple's Developer platform, Handibur is designed to combine social connectivity with advanced AI, providing sign language users with an inclusive tool for seamless interaction.

LIMITATIONS

Despite advancements, several limitations remain. The dataset, with one video per gesture and limited signing style diversity, restricts generalization to unseen signers and real-world scenarios. The sentence-level evaluation dataset is small compared to larger ASL datasets, and limited computational resources hinder broader testing and fine-tuning. The system currently supports unidirectional sign language-to-text translation, and expanding to bidirectional translation would enhance conversational capabilities. Additionally, linguistic challenges in Hebrew, such as gender distinctions and missing prepositions, require post-processing for accuracy. A lightweight secondary model could address these issues without compromising real-time performance. These limitations highlight areas for future research to improve the system's robustness and real-world applicability.

ACKNOWLEDGMENTS

We thank the Institute for the Advancement of Deaf Persons in Israel for providing the Hebrew Sign Language dataset. We wish to express our gratitude to Eva Karkar and Nir Tuttnauer for their collaboration on the development of "Handibur," a live video chat application integrating real-time sign language recognition. We are also thankful to the College of Management Academic Studies for providing access to computational resources that supported this research.

REFERENCES

[1] D. Li, C. R. Opazo, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," *arXiv*:1910.11006, 2020. [Online]. Available: https://arxiv.org/abs/1910.11006

[2] P. Rust *et al.*, "Towards privacy-aware sign language translation at scale," in *Proc. 62nd Annu. Meet. Assoc. Comput. Linguist. (ACL)*, Bangkok, Thailand, Aug. 2024, pp. 8624–8641. [Online]. Available: https://aclanthology.org/2024.acl-long.467

[3] K. He *et al.*, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 15979–15988.

[4] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meet. Assoc. Comput. Linguist. (ACL)*, Philadelphia, PA, USA, Jul. 2002, pp. 311–318.

[5] K. M. Duarte *et al.*, "How2Sign: A large-scale multimodal dataset for continuous American Sign Language," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nashville, TN, USA, Jun. 2021, pp. 2848–2858.

[6] B. Zhang *et al.*, "Scaling sign language translation," *arXiv*:2407.11855, 2024. [Online]. Available: https://arxiv.org/abs/2407.11855

[7] S. Gueuwou *et al.*, "SignMusketeers: An efficient multi-stream approach for sign language translation at scale," *arXiv*:2406.06907, 2024. [Online]. Available: https://arxiv.org/abs/2406.06907

[8] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," *arXiv*:2304.07193, 2023. [Online]. Available: https://arxiv.org/abs/2304.07193

[9] Institute for the Advancement of Deaf Persons in Israel, "Israeli Sign Language dictionary," 2024. [Online]. Available: https://sfatsimanim.co.il/en/home-page-2/

[10] C. Lugaresi *et al.*, "MediaPipe: A framework for building perception pipelines," *arXiv*:1906.08172, 2019. [Online]. Available: https://arxiv.org/abs/1906.08172

[11] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1746–1751.

[12] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[13] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, Haifa, Israel, Jun. 2010, pp. 807–814.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, Lille, France, Jul. 2015, pp. 448–456.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Stat. (AISTATS)*, Chia Laguna, Italy, May 2010, pp. 249–256.

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, Apr. 2014, pp. 1–10.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, ch. 6.

[18] D. Uthus, G. Tanzer, and M. Georg, "YouTube-ASL: A large-scale, open-domain American Sign Language–English parallel corpus,"

arXiv:2306.15162, 2023. [Online]. Available: https://arxiv.org/abs/2306.15162

[19] L. Tarrés *et al.*, "Sign language translation from instructional videos," *arXiv*:2304.06371, 2023. [Online]. Available: https://arxiv.org/abs/2304.06371

[20] K. Lin *et al.*, "Gloss-free end-to-end sign language translation," *arXiv*:2305.12876, 2023. [Online]. Available: https://arxiv.org/abs/2305.12876

[21] F. Chollet, "Keras: The Python deep learning library," in *Proc. IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, FL, USA, Dec. 2015, pp. 1–4.

APPENDIX

A. Terminology Clarification

To ensure clarity, we define the key terms used throughout this paper:

- **Gesture**: A specific hand movement or pose, typically represented by hand landmark data. Each gesture corresponds to a visual input segment processed by the model.
- **Sign**: A complete unit in sign language that conveys a concept or meaning. In our dataset, each sign corresponds to one isolated video gesture, aligned with a single HSL word.
- **Word**: The textual representation output by our system after classifying a sign gesture.

B. Hardware Specifications

The experiments were conducted using the following hardware and software configurations:

- **CPUs:** The experiments utilized multiple CPUs, including both Intel and AMD processors. Specifically:
  - Intel i7 12th Generation
  - AMD Ryzen 7 3700X

  The systems were equipped with up to 350 GB of RAM.
- **GPUs:** For model training and evaluation, we used NVIDIA A100 GPUs (PCIe 40 GB) with a total of 40 GB VRAM and 85 GB of system RAM.

C. Software Versions

The following software versions were used:

- TensorFlow: 2.17.0
- Keras: 3.4.1

These configurations ensured compatibility and efficiency during model training and testing.

D. Layer Type Breakdown

Table II provides a breakdown of layer types in the model architecture.

TABLE III
TOP 5 RESULTS BY BLEU-4 (SENSITIVITY ANALYSIS OF HYPERPARAMETERS; $x\_predictions = 3$, STEP INTERVAL= 5 FOR ALL ROWS)

Confidence	BLEU-1	BLEU-2	BLEU-3	BLEU-4	SacreBLEU
0.65	0.5566	0.4132	0.3016	**0.2115**	0.2540
0.60	0.5629	0.4154	0.3013	0.2096	0.2487
0.70	0.5505	0.4093	0.2991	0.2095	0.2549
0.75	0.5397	0.4017	0.2929	0.2042	0.2524
0.80	0.5304	0.3939	0.2860	0.1975	0.2475

TABLE II
LAYER TYPE BREAKDOWN

Layer Type	Count
Dense	2
CustomConfidenceActivation	1
AttentionLayer	1
TimeDistributed	3
InputLayer	1
Dropout	1
BatchNormalization	2
Bidirectional	1

E. Training Configuration

The model is trained using the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$) with a learning rate of 1×10^{-4} and gradient clipping (clipnorm $= 1.0$) for stability. Training is conducted with a batch size of 32 for up to 500 epochs. EarlyStopping halts training if validation loss does not improve for 20 epochs, preventing overfitting and unnecessary computation. ReduceLROnPlateau [21] lowers the learning rate by a factor of 0.5 after 10 stagnant epochs, down to 1×10^{-6}, ensuring sufficient convergence.

The dataset was split into training, validation, and test sets in proportions of 70%, 15%, and 15%, respectively. Labels were one-hot encoded to represent the gesture classes, facilitating effective training and loss computation. This encoding method ensures compatibility with the categorical cross-entropy loss used by the model and maintains class separation during training.

Details of the model's configuration and parameters, such as the total number of trainable and non-trainable parameters, as well as the depth of the model, are summarized in Table IV.

TABLE IV
ADDITIONAL MODEL INFORMATION

Parameter	Value
Trainable Parameters	2,259,750
Non-Trainable Parameters	2,048
Total Parameters	2,261,798
Model Depth (Number of Layers)	12

These values provide insights into the model's complexity, its capacity to learn from the dataset, and the computational resources required for training.

a) Hyperparameter Selection.: Initial values were chosen based on standard defaults from the literature and refined through empirical tuning. While it was not feasible to record every configuration tested, the final selections reflect stable and consistent results across multiple experiments.

F. BLEU Hyperparameter Sensitivity Analysis

The impact of hyperparameters on BLEU scores is shown in Table III. These results highlight the optimal configuration of a 0.65 confidence threshold, which balances precision and recall.