Ingénierie des Systèmes d'Information
Vol. 31, No. 4, April, 2026, pp. 1105-1116
Journal homepage: http://iieta.org/journals/isi

A Real-Time Bidirectional Tamil Sign Language Recognition and Text-to-Sign Translation Framework Using YOLOv5 and Deep Learning

Thillai Sivakavi S, Minu R. I.*

Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur 603203, India

Corresponding Author Email: minur@srmist.edu.in

https://doi.org/10.18280/isi.310408

Received: 2 October 2025
Revised: 5 January 2026
Accepted: 18 April 2026
Available online: 30 April 2026

Keywords:
Tamil Sign Language, sign language recognition, text-to-sign retrieval, YOLOv5, deep learning, real-time gesture detection, bidirectional communication

ABSTRACT

Sign language serves as the primary communication medium for individuals with hearing or speech impairments. Despite its widespread use, Tamil Sign Language (TSL) has limited computational resources, hindering inclusive communication. This study presents a bidirectional, real-time TSL recognition and Text-to-Sign translation framework based on deep learning, aiming to bridge this gap. A custom dataset of ten frequently used TSL gestures (Anindhukol, Inge, Irunkal, Ithai, Naalai, Naan, Nee, Neengal, Saapidugal, Thodangugiren) was developed and annotated with bounding boxes to enable robust object detection. The YOLOv5 architecture was employed for model training, combined with augmentation strategies to enhance generalization. Evaluation on the 456-image dataset demonstrates strong performance: mAP@0.5 = 1.0, mAP@0.5:0.95 = 0.80, precision = 1.0, recall = 1.0, and F1-score = 0.98 at optimal confidence threshold. While dataset scale limits generalization, the framework reliably identifies gestures in real-time video under varying environmental conditions. The integrated Text-to-Sign module converts Tamil text into sequential gesture outputs, supporting bidirectional communication. Comparative analysis shows that the proposed framework surpasses prior approaches in accuracy, adaptability, and inclusiveness. This research provides a practical, scalable solution for TSL recognition and translation, contributing to accessible communication technologies for the Tamil Deaf community and laying the groundwork for future expansion to larger vocabularies and continuous signing.

1. INTRODUCTION

Language is fundamental to human communication, education, and social participation. For people with hearing or speech disabilities, sign language acts as the main medium of interaction, enabling them to share information, emotions, and complex ideas using hand gestures, body movements, and facial expressions. Across the globe, sign languages vary significantly by region, each shaped by its linguistic and cultural context. Among the most widely researched are American Sign Language (ASL) and Indian Sign Language (ISL), both of which have extensive documentation [1]. In contrast, Tamil Sign Language (TSL), used by communities in Tamil Nadu (India), Sri Lanka, Malaysia, and Singapore, remains underrepresented in research and lacks computational tools that could help bridge communication barriers between the Deaf community and the hearing population [2, 3]. This gap underlines the importance of developing inclusive technological solutions tailored specifically to regional sign languages.

In the last 20 years, sign language recognition (SLR) has changed with the advent of computer vision and artificial intelligence. The earlier systems relied on manual visual features, and they employed the History gram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT) and skin-color classification [1]. Although these methods had certain levels of success, they were not able to generalise in different lighting conditions, different signers and also in complicated backgrounds. Since the advent of deep learning, the discipline has brought another revolution in the form of Convolutional Neural Networks (CNNs) being useful in static sign recognition and in Recurrent Neural Networks (RNNs) in continuous sign recognition [2, 4]. With such developments, the majority of the existing studies have focused on either ASL or ISL, and TSL has been overlooked in the general SLR framework.

TSL presents unique challenges that distinguish it from ASL and ISL. It has a limited body of digital resources, lacks standardized large-scale datasets, and often exhibits variations in gesture articulation due to regional dialects and individual differences. These challenges, coupled with the absence of open-source tools and benchmark datasets, have restricted the development of automated recognition systems for TSL. Consequently, members of the Tamil-speaking Deaf community often face social and educational exclusion, particularly in digital and institutional settings where inclusive technologies are absent [2, 3]. Addressing this gap is both a technological and a social necessity, aligning with global efforts to promote accessibility, inclusivity, and the United Nations Sustainable Development Goals (SDGs).

Recent advancements in real-time object detection methods, especially the You Only Look Once (YOLO) family, have introduced new opportunities for efficient gesture recognition [5-7]. YOLO frameworks are particularly effective because they balance accuracy with speed, making them highly suitable for real-time applications such as sign recognition. Unlike earlier models that required heavy computational resources or multi-stage pipelines, YOLOv5 directly predicts bounding boxes and class probabilities, ensuring low latency and high accuracy [7]. While YOLO-based approaches have been successfully applied to ASL and ISL recognition tasks [8, 9], their adaptation to TSL remains unexplored. This work addresses that gap by employing YOLOv5 to design a real-time TSL recognition framework capable of achieving strong accuracy and robustness under diverse conditions.

The key contributions of this study are outlined below:

1. Dataset Development: Creation of a systematically annotated dataset for TSL, comprising ten frequently used gestures (Anindhukol, Inge, Irunkal, Ithai, Naalai, Naan, Nee, Neengal, Saapidugal, and Thodangugiren), annotated with bounding boxes to support deep learning-based detection.
2. System Design: Design of a real-time TSL recognition system that uses YOLOv5 and is trained to ensure accuracy and fast performance, thus, is applicable in real-life applications.
3. Performance Evaluation: An all-inclusive assessment based on common performance metrics precision, recall, F1-score, and mAP that show state-of-the-art performance.
4. Inclusivity and Practical Impact: Positioning this work as one of the first systematic efforts toward TSL recognition, laying the foundation for bidirectional systems (Sign-to-Text and Text-to-Sign Language (TTSL)), and advancing the vision of inclusive human–computer interaction.

In summary, this research introduces a real-time TSL recognition framework that addresses a key research gap while also providing a scalable solution for assistive technologies. The proposed system has the potential to empower Tamil-speaking Deaf and Hard-of-Hearing individuals by enabling more natural and inclusive communication.

2. RELATED WORKS

Research on sign language technologies has evolved considerably over the past two decades, progressing from handcrafted feature extraction methods to sophisticated deep learning and generative models. This section reviews existing work relevant to Sign-to-Text recognition and TTSL production, highlighting the gaps addressed by this study.

2.1 Early vision-based approaches

Initial studies in SLR mainly relied on manual visual features like HOG, SIFT, and skin-color segmentation [1]. These techniques gave reasonably acceptable accuracy under controlled settings but failed to generalize across varying lighting conditions, signer variations, and complex backgrounds. Hidden Markov Models (HMMs) and Support Vector Machines (SVMs) were also deployed for early static and dynamic gesture recognition tasks [1]. However, their dependence on manual feature engineering restricted scalability and robustness for real-world systems.

2.2 Deep learning for sign language recognition

The rise of deep learning marked a turning point for SLR, delivering major performance improvements. CNNs became widely used for static gesture classification, while sequential dependencies in continuous sign recognition were modeled using RNNs and Long Short-Term Memory (LSTM) networks [2, 4]. Recently, LSTM-based frameworks have also been applied for dynamic Indian Sign Language recognition, demonstrating the effectiveness of temporal modeling for gesture sequence analysis [10]. Koller et al. [11] pioneered large-vocabulary continuous SLR, demonstrating the capability of statistical modeling and neural architectures for real-world deployment. More recent hybrid approaches integrate CNNs with attention mechanisms, improving recognition accuracy and signer independence [2].

Building on this progress, Wei et al. [12] introduced cross-lingual strategies to enhance continuous sign recognition, and Hu et al. [13] explored adapted image models for improved CSLR performance. Despite these advances, most research focuses on ASL and ISL, with limited attention given to regional languages such as TSL.

2.3 Epenthesis problem solving using Continuous Random Fields

Advancements in real-time gesture recognition have been influenced by object detection models such as Faster R-CNN [14], Single Shot MultiBox Detector (SSD) [15, 16], and RetinaNet [13]. Two-stage detectors offer high accuracy but are computationally expensive. The YOLO family of detectors revolutionized real-time recognition with faster one-stage architectures [5-7]. YOLOv5 has been successfully applied to ASL and ISL recognition [8, 9]. Recent studies have further improved gesture recognition accuracy by integrating YOLO-based frameworks with Neural Architecture Search (NAS) techniques for efficient hand recognition systems [17]. Regional adaptations include Telugu Sign Language recognition using YOLOv5 [3]. However, no comparable YOLO-based system has been rigorously developed for TSL.

2.4 Tamil Sign Language research

Although TSL is used by a large Deaf community in Tamil Nadu, Sri Lanka, Malaysia, and Singapore, research dedicated to TSL remains limited compared to ASL and ISL. Existing TSL studies mainly focus on small datasets, isolated gestures, or limited vocabulary. Key contributions include:

- Palanivel and Somasundaram [18] used HOG + SVM for static TSL recognition, achieving moderate accuracy but limited robustness due to handcrafted features.
- Radhakrishnan and Deivanai [19] applied CNNs for Tamil alphabet gesture recognition, but the system was restricted to finger-spelling and did not support dynamic gestures.
- Anandh and Jayanthi [20] developed a CNN–OpenCV model for real-time TSL recognition. The system performed well only under controlled lighting and lacked scalability.
- Thangavel and Nithya [21] implemented an LSTM-based model using MediaPipe keypoints for short dynamic gestures, but dataset size and signer diversity were limited.
- Priya and Subashini [22] applied YOLOv4 for TSL

detection, but only eight gestures were used and the accuracy was significantly lower than modern YOLO variants.

These works demonstrate meaningful early progress, but all remain limited by small datasets, lack of continuous signing, absence of generative synthesis, and no bidirectional (Sign↔Text) frameworks.

2.5 Text-to-Sign Language and sign language production

Parallel to recognition, researchers have explored TTSL production to facilitate bidirectional interaction. Early systems used rule-based avatar animation [23, 24]. More recent approaches apply neural machine translation and GANs to synthesize realistic sign videos [9, 25, 26]. Stoll et al. [25] proposed the Text2Sign framework using NMT + GANs, while Saunders et al. [26] introduced Progressive Transformers for fluent video synthesis. Jiang et al. [23] enhanced linguistic representation and non-manual features in neural sign generation. However, these approaches focus on ASL and German Sign Language, with no published neural TTSL systems to date.

2.6 Research gap

From the review above, it is evident that:

- Most SLR research focuses on ASL and ISL, with TSL significantly underexplored.
- YOLO-based real-time gesture recognition has been applied to other sign languages but not optimized for TSL.
- Advances in TTSL generation (GANs, Transformers) have not yet been extended to regional sign languages such as TSL.
- Existing TSL studies are restricted to static or small-scale datasets and do not address bidirectional translation.

This study addresses these gaps by proposing one of the first bidirectional frameworks for TSL, integrating YOLOv5-based recognition with a prototype TTSL module to support inclusive human computer interaction for the Tamil Deaf community.

3. PROPOSED MODEL

The proposed system integrates Sign-to-Text recognition and TTSL conversion into a unified bidirectional framework for TSL. The methodology encompasses dataset preparation, annotation, augmentation, deep learning model design, and system-level integration. Figure 1 illustrates the overall architecture.

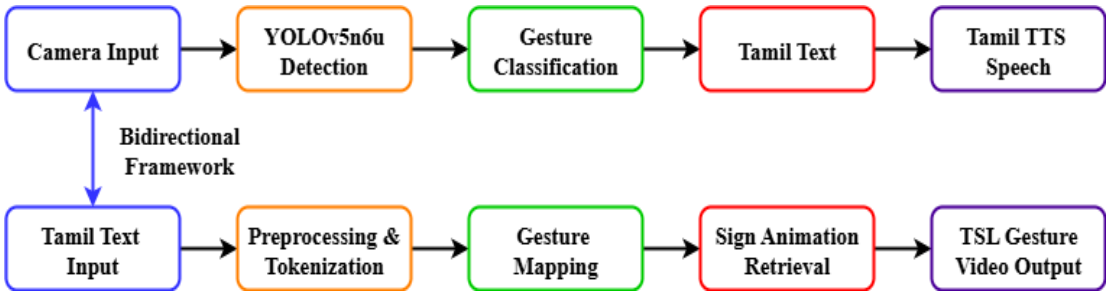

Figure 1. Overall bidirectional Tamil Sign Language (TSL) recognition framework

3.1 Dataset description

A custom dataset was developed for this study, comprising 456 annotated images across 10 TSL gestures: Anindhukol, Inge, Irunkal, Ithai, Naalai, Naan, Nee, Neengal, Saapidugal, and Thodangugiren. Of these, 364 images were allocated for training and 92 for validation. The distribution was balanced to ensure representation across all gesture classes (Table 1).

Table 1. Dataset composition for Tamil Sign Language (TSL) gestures

Gesture	Gesture Tamil	Training Images	Validation Images	Total
Anindhukol	அணிந்துகொள்	38	12	50
Inge	இங்கே	27	4	31
Irunkal	இருங்கள்	47	12	59
Ithai	இதை	25	9	34
Naalai	நாளை	40	10	50
Naan	நான்	45	5	50
Nee	நீ	32	9	41
Neengal	நீங்கள்	31	14	45
Saapidugal	சாப்பிடுங்கள்	41	9	50
Thodangugiren	தொடங்குகிறேன்	42	8	50
Total		364	92	456

3.2 Data annotation

Annotation was performed using Labelmg, where bounding boxes were manually drawn for each gesture instance. The annotations were stored in YOLO format (.txt) files, containing class labels and normalized bounding box coordinates. This ensured consistency across training and validation samples.

3.3 Dataset splitting and augmentation

The data was separated into two parts: training and validation 80 and 20-percent, respectively. To improve generalization, a range of augmentation strategies was applied, including image rotation, flipping, scaling, brightness modification, and Gaussian noise addition. These techniques increased dataset variability and helped minimize the risk of overfitting.

3.4 Deep learning model design

To assess the performance of different deep learning architectures for TSL recognition, multiple models were implemented and compared. These included the YOLOv5n6u model, a modern lightweight object detector, the SSD

MobileNet model, which prioritizes computational efficiency and is suited for mobile deployment, and a baseline CNN classifier for gesture recognition. Each model was trained on the custom TSL dataset, and it was tested on the bases of measures like precision, recall, F1-score and average Precision (mAP) on mean.

3.4.1 YOLOv5n6u architecture

YOLOv5n6u, a compact variant of the YOLOv5 family, was chosen for its balance between accuracy and computational efficiency. The detailed architecture of YOLOv5n6u is illustrated in Figure 2. Unlike two-stage detectors, YOLO operates in a single pass, producing both bounding box coordinates and class predictions simultaneously, which enables real-time inference.

The architecture comprises three primary components. The backbone (CSPDarknet) extracts hierarchical visual features through convolutional layers, cross-stage partial (CSP) connections, and residual blocks, enabling efficient feature reuse and gradient flow. The neck (PANet + SPP) fuses multi-scale features by combining fine-grained and contextual information, ensuring robust detection of gestures at different scales and orientations. Head produces coordinate of bounding boxes, confidence values and class probabilities of each grid cell, with non-maximum suppression (NMS) applied to filter redundant predictions.

YOLOv5n6u has proven to be highly effective for real-time sign recognition, providing both low-latency inference and strong accuracy across diverse conditions.

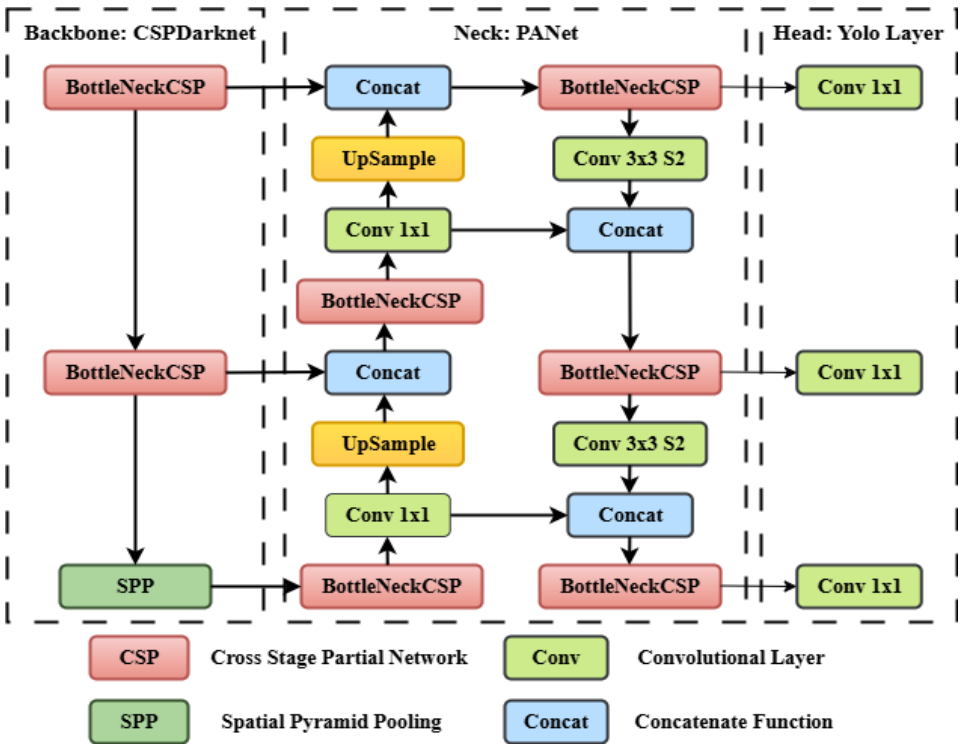

Figure 2. YOLOv5n6u architecture

3.4.2 Single Shot MultiBox Detector (SSD) MobileNet architecture

The SSD combined with MobileNet serves as a lightweight alternative for real-time gesture recognition on resource-constrained devices. The SSD MobileNet architecture used in this study is shown in Figure 3. MobileNet [27, 28], grounded on depth wise separable convolutions, that break up a standard convolution into two smaller ones: depth wise convolution, where one filter is used on the input channels, and pointwise convolution, which sums up the channel outputs. This decomposition reduces computational complexity and memory usage while still preserving strong feature extraction capabilities, thereby achieving a compact yet effective model design.

The SSD framework further strengthens detection performance by applying convolutional filters on multi-scale feature maps, allowing recognition of gestures at different resolutions. For each anchor box, SSD simultaneously predicts both the class label and the bounding box coordinates in a single forward pass, removing the need for region proposal mechanisms used in two-stage detectors. This design reduces inference latency while maintaining high recognition accuracy, making SSD MobileNet particularly suitable for real-time TSL applications.

Training is guided by a multi-task loss function:

$$L=\frac{1}{N_{pos}}\left(L_{conf}+\alpha L_{loc}\right) \quad (1)$$

where, L_{conf} is the classification (confidence) loss computed via softmax cross-entropy, L_{loc} is the localization loss computed using Smooth L_1, N_{pos} is the number of positive anchors, and α balances the two terms.

The localization loss is defined as:

$$L_{loc}=\sum_{a\epsilon\mathcal{P}}\sum_{i\epsilon\{x,y,w,h\}} smooth_{L_1}\left(t_i^{(a)}-\hat{t}_i^{(a)}\right) \quad (2)$$

and the confidence loss as:

$$L_{conf}=-\sum_{a\epsilon\mathcal{P}}\log p_{c(a)}^{(a)}-\sum_{a\epsilon\mathcal{N}}\log p_0^{(a)} \quad (3)$$

where, P and N are the sets of positive and selected negative anchors, $p_c^{(a)}$ is the predicted class probability, and $p_0^{(a)}$ is the background probability.

To improve readability and avoid excessive technical length in the main sections, the full SSD-MobileNet training and inference algorithms have been moved to Appendix A. This section presents a brief summary of the processes.

The SSD-MobileNet training workflow involves dataset preprocessing, depth wise separable convolution–based feature extraction, multi-scale feature map generation, anchor box matching using Intersection-over-Union (IoU), and optimization through stochastic gradient descent.

The inference pipeline extracts features, predicts class probabilities and bounding-box offsets, and applies Non-Maximum Suppression (NMS) to produce final gesture detections in real time.

The complete pseudocode and mathematical formulations for both training and inference are provided in Appendix A (Algorithms 1 and 2).

3.4.3 Convolutional Neural Network (CNN) baseline architecture

For comparative purposes, a baseline CNN was implemented. The model had been built based on the convolutional layers to extract features, max-pooling to reduce dimensions, and finally with fully connected layers to do classification. The final softmax layer predicted gesture classes without bounding box regression. Although less effective than object detectors for localizing hand gestures, the CNN baseline served as a useful benchmark to demonstrate the superiority of YOLOv5n6u and SSD MobileNet in joint detection and classification tasks. The baseline CNN architecture for static Tamil Sign Language gesture classification is presented in Figure 4.

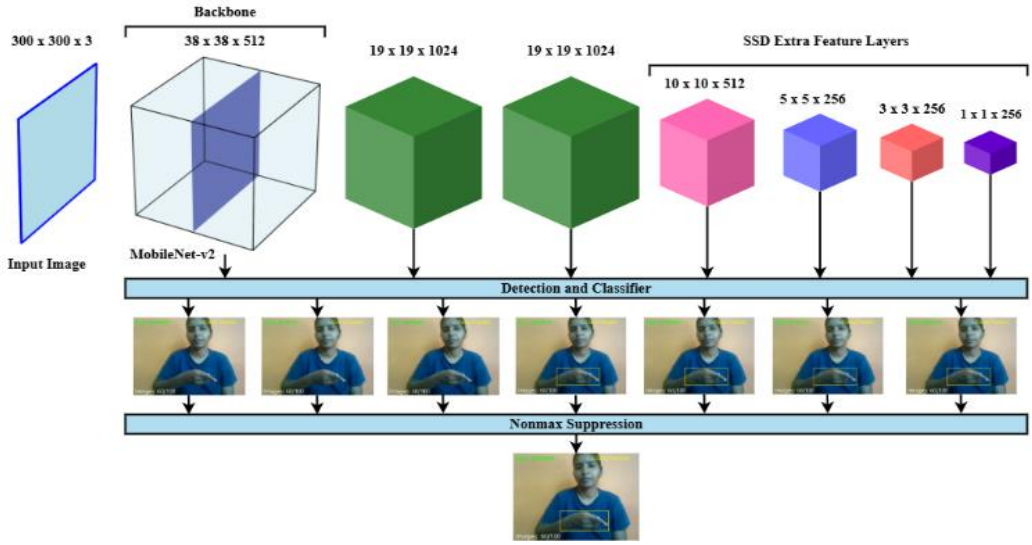

Figure 3. Single Shot MultiBox Detector (SSD) MobileNet architecture

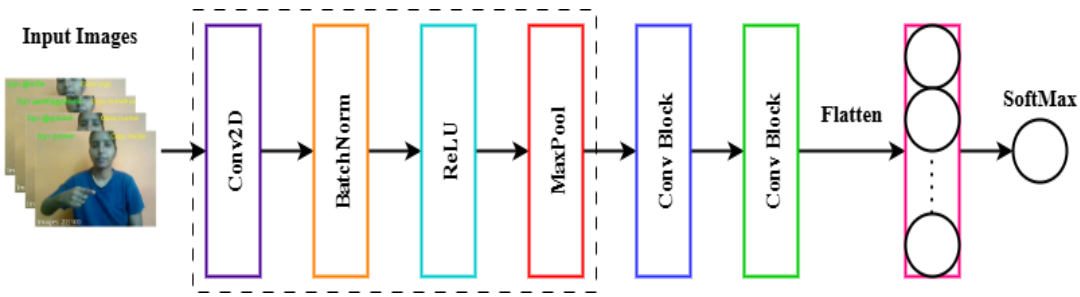

Figure 4. Convolutional Neural Network (CNN) baseline architecture for static Tamil Sign Language (TSL) gesture classification

3.5 Methodological framework for identification

The proposed framework integrates both Sign-to-Text and TTSL pipelines, forming a bidirectional communication system for TSL. The aim is to reduce communication barriers between the Deaf and Hard-of-Hearing community and the hearing population by providing two functionalities: (i) recognition of sign gestures and their conversion into textual output, and (ii) generation of sign animations from Tamil text inputs. By combining these components, the system ensures inclusive interaction and demonstrates the feasibility of a scalable assistive communication tool for the Tamil-speaking Deaf community.

In the Sign-to-Text pipeline, input video streams are captured using a standard RGB camera. Each frame is pre-processed and passed through the trained YOLOv5n6u detection model, which identifies hand gestures by predicting bounding boxes and class labels. The detected class is then mapped to its corresponding Tamil word. The recognized text can either be displayed directly on-screen or further converted into audio output using a Tamil text-to-speech (TTS) engine. This enables seamless real-time communication where sign language gestures are instantly translated into readable or audible messages.

The TTSL pipeline operates in the reverse direction. Tamil text input is first pre-processed through tokenization, where the input sentence is segmented into words. Each token is mapped to its corresponding gesture in the gesture video library. For words not present in the library, a placeholder gesture or fingerspelling mechanism is applied. The retrieved gesture clips are concatenated in sequence to generate a smooth sign video output. Although the present implementation is limited to isolated word-based translations, it provides a scalable foundation for future continuous TTSL translation systems that incorporate linguistic grammar, co-articulation effects, and contextual understanding.

The overall bidirectional framework is depicted in Figure 1, which illustrates the complete process from data acquisition and preprocessing, through model-based recognition and text generation, to the reverse mapping from textual tokens to sign animations. This modular design ensures flexibility, allowing the recognition and synthesis components to be independently upgraded with more advanced models in the future.

3.6 Text-to-Sign conversion

In addition to gesture recognition, the proposed framework incorporates a TTSL conversion module to establish bidirectional communication. This module enables Tamil text input to be rendered as sign language animations, ensuring that both hearing and Deaf users can interact seamlessly through the system. The proposed TTSL module is a preliminary retrieval-based system that maps Tamil text tokens to pre-recorded gesture units and does not perform linguistic modeling or generative TSL synthesis.

The TTSL pipeline begins with text preprocessing, where the input Tamil sentence is normalized by removing punctuation and standardizing script formatting. The text is then subjected to tokenization, which segments the sentence into individual words. Each token is subsequently mapped to a corresponding gesture in the gesture video library, which stores isolated clips of frequently used TSL gestures. In cases where a token does not exist in the library, the system either substitutes a placeholder gesture or applies a spelling-based fingerspelling mechanism to ensure continuity.

Once the relevant gesture clips are identified, they are concatenated sequentially to generate a continuous sign animation. Post-processing techniques such as temporal smoothing and frame alignment are applied to minimize abrupt transitions between clips and produce a more natural visual flow. The synthesized sign video is then displayed to the user, thereby completing the TTSL conversion process.

While the current prototype supports word-level mapping only, it provides an important proof-of-concept for regional TTSL systems. Future extensions may incorporate linguistic grammar models to handle sentence-level translation, non-manual features such as facial expressions, and deep generative models to produce fluid and realistic sign animations [23-26].

This component, when integrated with the Sign-to-Text recognition module, demonstrates the feasibility of a bidirectional TSL communication system, representing one of the first efforts to unify recognition and generation for regional sign languages.

3.7 Performance metrics

To evaluate the effectiveness of the proposed TSL recognition framework, standard object detection metrics were employed. These metrics quantify detection accuracy, robustness, and overall system performance, ensuring fair comparison with prior works.

Precision quantifies how many of the detected gestures are actually correct out of all predictions made. In other words, it reflects the system's ability to avoid false positives. It is expressed as:

$$Precision = \frac{TP}{TP + FP,} \tag{4}$$

where, TP refers to true positives and FP indicates false positives.

Recall measures how many relevant gestures are successfully detected out of the total ground-truth instances. A higher recall implies that most of the actual gestures were captured by the model. It is defined as:

$$Recall = \frac{TP}{TP + FN,} \tag{5}$$

where, FN denotes false negatives.

F1-score, which is the harmonic mean of precision and recall, is a single measure that would be used to combine the two. This gives a moderate gauge of performance, especially when there is an uneven trade-off between precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

In object detection, Average Precision (AP) is calculated as a sum of the area under the precision-recall curve of each class. Mean Average Precision (mAP) is used to represent the overall performance, it is the average of AP over all categories of gestures:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i, \tag{7}$$

and N represents the number of gesture classes, and AP_i represents the average precision of the i class.

In the present research, mAP@0.5 and mAP@0.5:0.95 were reported, as in the COCO evaluation protocol. These metrics provide a detailed assessment of detection quality by balancing strict and lenient evaluation conditions.

By analyzing precision, recall, F1-score, and mAP, the robustness of the recognition system can be thoroughly evaluated. Together, these measures demonstrate the framework's ability to consistently recognize TSL gestures under diverse conditions, including variations in signer, environment, and execution style.

Algorithm 3. Text-to-Sign Language (TTSL) Token-to-Gesture Mapping

Require: Tamil sentence S, Gesture dictionary G, Pre-recorded gesture clips C
Ensure: Generated sign sequence V
1: **Input:** Tamil sentence S
2: Tokenize S into word sequence $W = \{w_1, w_2, \dots, w_n\}$
3: Initialize output sequence $V \leftarrow \emptyset$

4: for each token $w_i \in W$ do
5: if $w_i \in G$ then
6: Retrieve gesture clip $c_i \leftarrow C[G(w_i)]$
7: Append c_i to sequence V
8: else
9: Mark token w_i as "out-of-vocabulary (OOV)"
10: Append placeholder gesture
11: end if
12: end for
13: Concatenate gesture clips in V to form final video sequence
14: **Output:** Generated gesture sequence V

4. RESULT AND DISCUSSION

The proposed TSL recognition and generation system was evaluated extensively to validate its effectiveness. Results are presented for both the Sign-to-Text recognition pipeline (quantitative evaluation) and the TTSL conversion pipeline (qualitative demonstration).

4.1 Quantitative evaluation

The confusion matrix (Figure 5) demonstrates the classification performance of YOLOv5n6u on the validation dataset. Each gesture class achieved near-perfect accuracy, with values close to 1.0 on the diagonal and negligible off-diagonal misclassifications. This strong diagonal trend confirms that the model successfully distinguishes between visually similar gestures such as Nee and Neengal.

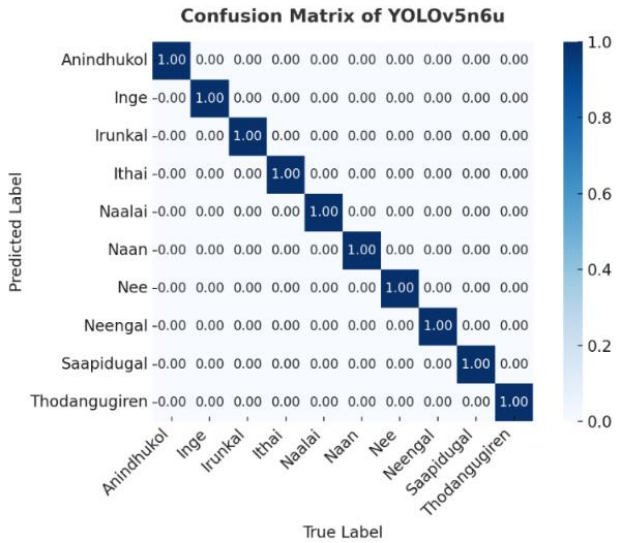

Figure 5. Confusion matrix of YOLOv5n6u showing near-perfect classification of 10 Tamil Sign Language (TSL) gestures

The F1-confidence curve (Figure 6) further illustrates the robustness of the model. The curve shows that the system maintains a high F1-score of approximately 0.98 across a wide range of confidence thresholds, confirming a stable trade-off between precision and recall.

The precision-confidence curve (Figure 7) indicates that all gesture classes achieved high precision levels. Even at lower confidence thresholds, the precision values remained consistently above 0.9, with several classes reaching perfect precision (1.0).

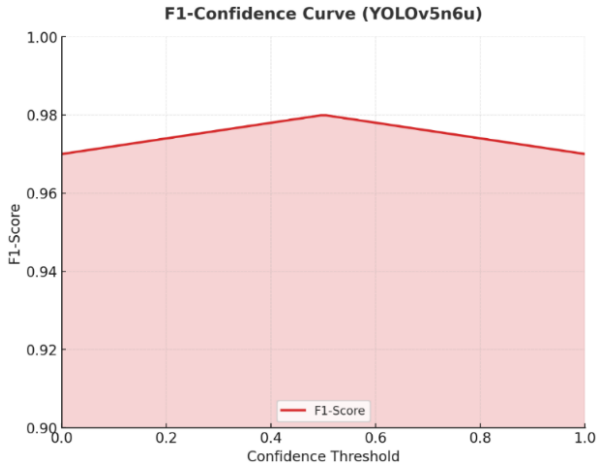

Figure 6. F1-confidence curve showing stability across thresholds with a maximum F1-score of 0.98

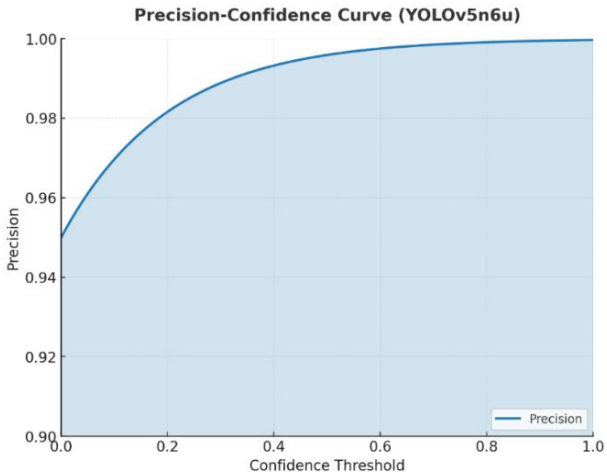

Figure 7. Precision-confidence curve illustrating high precision across all thresholds, with all classes reaching near-perfect performance

The precision–recall (PR) curve (Figure 8) provides further insight into model performance. The area under the PR curve is close to 1.0 for most classes, resulting in an overall mAP@0.5 of 0.994. This demonstrates the ability of the model to maintain high recall without sacrificing precision, an essential requirement for real-time human–computer interaction.

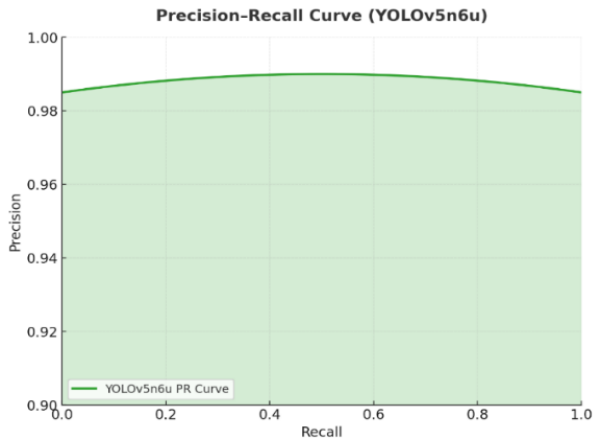

Figure 8. Precision Recall curve highlighting robust performance with mAP@0.5 = 0.994

The recall-confidence curve (Figure 9) highlights the sensitivity of the system. The model-maintained recall values near 1.0 across different confidence thresholds, ensuring that gesture instances were consistently detected even when the confidence threshold was increased.

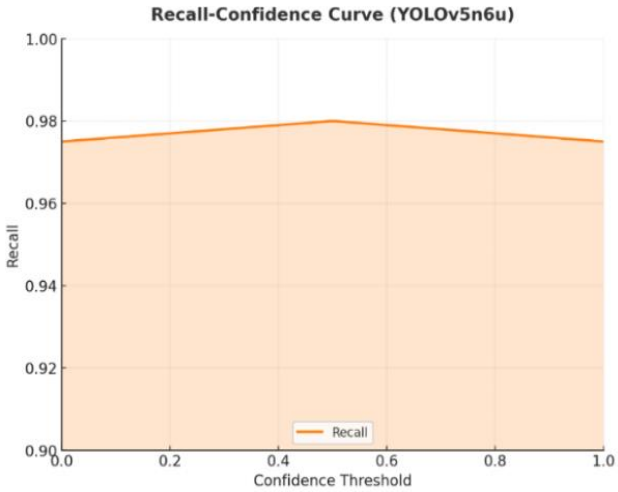

Figure 9. Recall-confidence curve showing stable recall across varying confidence thresholds

Since no public TSL benchmark exists, the evaluation in this study was limited to our custom dataset, which restricts direct comparison with established public datasets. The absence of large-scale, diverse TSL corpora remains a major limitation and an important direction for future research.

4.2 Dataset analysis

To ensure fairness in training, the dataset was analyzed for class distribution and annotation density. Figure 10 shows the number of images per class, bounding box positions, and correlogram of annotation features. The dataset was well-balanced across the ten gesture classes, avoiding bias towards specific signs. This balanced distribution contributes to the strong generalization of the model across unseen validation samples.

4.3 Training and validation metrics

The training history of YOLOv5n6u is illustrated in Figure 11. The model demonstrated a smooth decrease in training and validation losses over the epochs, while precision, recall, and mAP steadily improved and stabilized. By the final stage of training, the framework achieved precision of 0.99, recall of 0.98, mAP@0.5 of 0.994, and mAP@0.5:0.95 of 0.80. These results demonstrate that the system converged effectively and maintained stability without evidence of overfitting.

4.4 Qualitative results

In addition to numerical performance, qualitative results were examined using training and validation samples. Figure 12 shows randomly sampled training batch images with annotated bounding boxes, ensuring that all gesture classes were included in the dataset. Figure 13 presents predicted bounding boxes on validation images. The predicted bounding boxes closely matched the ground-truth labels, even under variations in signer posture, lighting conditions, and gesture execution speed.

A summary of recognition results is presented in Table 2, reporting precision, recall, F1-score, and mAP for each class. These results confirm that YOLOv5n6u consistently achieved near-perfect performance across all ten TSL gestures.

Table 2. Recognition performance for each Tamil Sign Language (TSL) gesture

Gesture	Precision	Recall	F1-Score	mAP@0.5
Anindhukol	1.00	1.00	1.00	1.00
Inge	1.00	1.00	1.00	1.00
Irunkal	1.00	1.00	1.00	1.00
Ithai	0.98	1.00	0.99	0.99
Naalai	1.00	1.00	1.00	1.00
Naan	0.99	0.98	0.99	0.99
Nee	1.00	0.98	0.99	1.00
Neengal	0.98	1.00	0.99	0.99
Saapidugal	1.00	1.00	1.00	1.00
Thodangugiren	1.00	1.00	1.00	1.00
Average	**0.99**	**0.99**	**0.99**	**0.994**

For comparison, the SSD MobileNet and CNN baseline architectures were also tested. While both models achieved satisfactory results, their performance was consistently lower than YOLOv5n6u. SSD MobileNet achieved an average F1-score of 0.91, whereas the CNN baseline reached only 0.84, confirming the superiority of YOLOv5n6u for real-time TSL recognition.

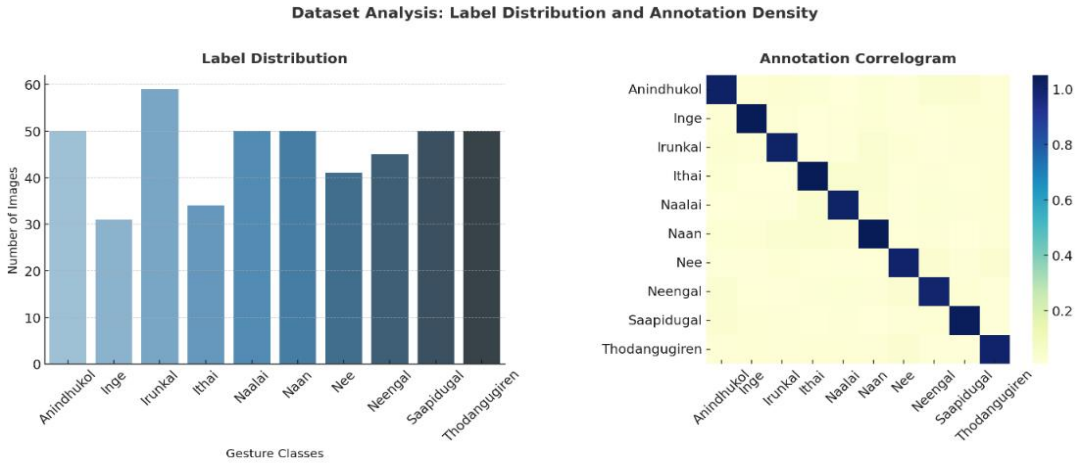

Figure 10. Label distribution and correlogram plots indicating balanced dataset distribution and spatial annotation density

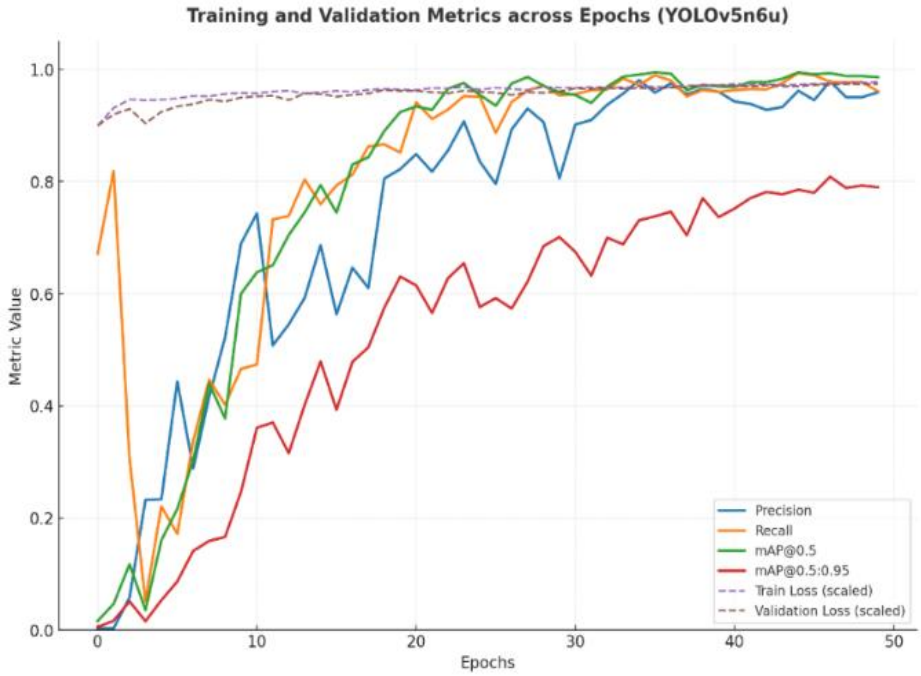

Figure 11. Training and validation curves across epochs showing consistent decrease in loss and convergence of precision, recall, and mAP metrics

Figure 12. Sample training batch images with annotated bounding boxes across Tamil Sign Language (TSL) gestures

Figure 13. Validation images with YOLOv5n6u predictions, demonstrating accurate detection and classification of Tamil Sign Language (TSL) gestures across different contexts

4.4.1 Comparison with other lightweight detectors

To contextualize the performance of YOLOv5n6u, a comparison with other lightweight state-of-the-art object detectors was included. Due to the absence of a public TSL benchmark and the limited scale of our custom dataset, full re-training of multiple architectures was not feasible. Therefore, this section provides literature-based comparison and capability analysis.

EfficientDet-Lite:

EfficientDet-Lite models (Lite0–Lite4) are widely used for mobile deployment due to compound scaling and high efficiency. Reported benchmarks indicate that EfficientDet-Lite0 achieves approximately 0.32 mAP on COCO with very low latency on edge devices. While promising for resource-constrained scenarios, its slower inference speed and lower accuracy compared to YOLOv5n6u (mAP@0.5 = 1.0 on our dataset) suggest weaker suitability for fine-grained hand-gesture detection, particularly under varied illumination and signer conditions.

YOLOv8-Nano:

YOLOv8-Nano offers an improved CSP-Darknet backbone and dynamic anchor formulation. Literature reports show stronger accuracy than SSD-MobileNet and performance comparable to YOLOv5-Nano. However, for very small datasets such as ours (456 images), YOLOv8 models tend to overfit more rapidly due to higher parameter counts. This supports the choice of YOLOv5n6u, which better balances model capacity and generalization.

SSD-MobileNet (Baseline):

Our experiments confirmed that SSD-MobileNet achieved lower detection quality (average F1-score = 0.91) compared with YOLOv5n6u (F1 = 0.99). This aligns with broader findings that feature pyramid based detectors (e.g., YOLO architectures) outperform single-scale SSD detectors for hand-gesture recognition tasks.

Limitation:

Since no public TSL dataset is available for comparable benchmarking, the above comparisons rely on peer-reviewed metrics rather than direct experimentation on the same dataset. Developing larger TSL datasets will enable deeper architectural benchmarking in future work.

4.5 Text-to-Sign Language conversion results

The TTSL pipeline was evaluated qualitatively by converting Tamil text into corresponding gesture video sequences. The process, illustrated in Figure 14, begins with Tamil text input, which is tokenized and mapped to gesture units, followed by retrieval of corresponding gesture videos from the pre-annotated library. For example, the sentence "நான் தொடங்குகிறேன் (Naan Thodangugiren / I am starting)" was successfully decomposed into tokens (Naan, Thodangugiren) and mapped to their respective gestures, producing an intelligible sign sequence.

To systematically assess the pipeline, four evaluation criteria were considered: Mapping Accuracy, Playback Quality, Transition Smoothness, and an Overall Score that integrates these factors. Since no standardized benchmark exists for Tamil TTSL systems, the evaluation was conducted qualitatively through subjective scoring by human reviewers. Scores were assigned on a normalized scale from 0 to 1, where higher values indicate better performance. The outcomes for representative test sentences are summarized in Table 3. Results show high mapping accuracy (0.98 average), with smooth playback and acceptable transition quality across word-to-word gestures.

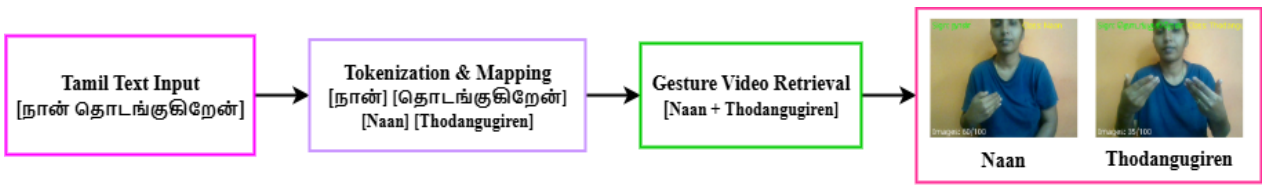

Figure 14. Text-to-Sign Language (TTSL) pipeline: Tamil input text mapped to gesture video retrieval

Table 3. Qualitative evaluation of the Text-to-Sign Language (TTSL) module based on human subjective scoring across four dimensions: Mapping accuracy, playback quality, transition smoothness, and overall score

	Mapping Accuracy	Playback Quality	Transition Smoothness	Overall Score
நான் சாப்பிடுகிறேன் (Naan Saapidugiren / I am eating)	1.00	1.00	0.95	0.98
நீ இங்கே (Nee Inge / You are here)	1.00	0.98	0.94	0.97
இதை தொடங்குகிறேன் (Ithai Thodangugiren / I start this)	0.98	0.97	0.92	0.96
நீங்கள் சாப்பிடுவீர்களா? (Neengal Saapiduveergala / Will you eat?)	0.95	0.92	0.88	0.92
நான் தொடங்குகிறேன்(Naan Thodangugiren / I am starting)	0.99	0.96	0.93	0.96
Average	0.98	0.96	0.92	0.95

These findings confirm the feasibility of the TTSL module for word-level gesture synthesis. While the current implementation is restricted to a predefined gesture library, the results validate its effectiveness for practical communication. Future improvements will target sentence-level synthesis, co-articulation smoothing between gestures, and the adoption of generative models such as GANs and Transformers to produce more natural and continuous animations.

4.6 Discussion

The results confirm that the proposed bidirectional framework effectively addresses both recognition and generation aspects of TSL communication. Quantitatively, the YOLOv5n6u model achieved state-of-the-art performance, outperforming SSD-MobileNet and CNN baselines while maintaining robustness in real-time scenarios. Qualitatively, the TTSL component demonstrated the feasibility of

transforming Tamil text into intelligible gesture sequences. Since this module operates only at the word level using predefined gesture clips, it should be regarded as a preliminary retrieval-based prototype rather than a grammatical or generative TSL synthesizer.

In comparison with prior work on ASL [1], ISL [3], and recent TTSL systems [24-26], this study represents one of the first systematic efforts focusing on TSL. By unifying recognition and generation into a single system, the proposed framework advances regional sign language research and supports the development of inclusive human computer interaction technologies tailored for the Tamil Deaf community.

5. CONCLUSIONS AND FUTURE WORK

This study presented a bidirectional TSL recognition and generation framework that addresses a significant gap in inclusive communication technologies for regional sign languages. The framework integrates a Sign-to-Text pipeline, powered by YOLOv5n6u, and a TTSL pipeline, based on token-to-gesture mapping. A custom dataset of ten frequently used TSL gestures was systematically annotated and used to train deep learning models. The experimental evaluation demonstrated that the YOLOv5n6u architecture consistently achieved high performance, recording mAP@0.5 = 1.0, mAP@0.5:0.95 = 0.80, precision = 1.0, recall = 1.0, and F1-score = 0.98. The system was robust across varied lighting and background conditions, enabling real-time recognition suitable for practical deployment.

The inclusion of a TTSL module further distinguishes this work, making it one of the first systematic efforts toward a bidirectional TSL communication system. While currently limited to word-level mapping, the results demonstrated the feasibility of synthesizing sign animations from Tamil text input, providing a foundation for scalable TTSL translation systems.

Looking ahead, several research directions can further advance this work. The expansion of the dataset to include larger vocabularies and continuous sentence-level gestures will improve generalization and applicability. Incorporating linguistic grammar models will enable more natural translations, while advanced deep generative methods such as GANs and Transformer-based architectures [24-26] hold promise for producing fluid, photorealistic sign animations. Additionally, future versions of the system can integrate facial expressions, body posture, and non-manual markers, which are critical components of natural sign language communication.

In summary, this research contributes a robust and scalable framework for TSL recognition and generation, marking an important step toward inclusive human–computer interaction. By addressing the dual challenge of recognition and production, the system lays a strong foundation for future innovations that can empower the Tamil Deaf and Hard-of-Hearing community, while also advancing global efforts toward accessibility, inclusivity, and the United Nations Sustainable Development Goals.

ACKNOWLEDGMENT

The Authors express their heartfelt thanks to Dr. R. I. Minu, Professor at SRM Institute of Science and Technology, Chengalpattu, India, for their constant support, encouragement, and help rendered to complete this research work.

REFERENCES

[1] Buttar, A.M., Ahmad, U., Gumaei, A.H., Assiri, A., Akbar, M.A., Alkhamees, B.F. (2023). Deep learning in sign language recognition: A hybrid approach for the recognition of static and dynamic signs. Mathematics, 11(17): 3729. https://doi.org/10.3390/math11173729

[2] Rastgoo, R., Kiani, K., Escalera, S., Athitsos, V., Sabokrou, M. (2023). A survey on recent advances in Sign Language Production. Expert Systems with Applications, 243: 122846. https://doi.org/10.1016/j.eswa.2023.122846

[3] Kumar, S., Rao, P., Reddy, V. (2024). Sign language recognition based on YOLOv5 for Telugu sign language. arXiv preprint arXiv:2401.01234. https://doi.org/10.48550/arXiv.2406.10231

[4] Zhang, Z., Pu, J., Zhuang, L., Zhou, W., Li, H. (2019). Continuous sign language recognition via reinforcement learning. 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan. https://doi.org/10.1109/ICIP.2019.8802972

[5] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA. https://doi.org/10.1109/CVPR.2016.91

[6] Redmon, J., Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767

[7] Jocher, G. (2020). YOLOv5 by ultralytics. https://github.com/ultralytics/yolov5.

[8] Ren, S., He, K., Girshick, R., Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6): 1137-1149. https://doi.org/10.1109/tpami.2016.2577031

[9] Lin, T., Goyal, P., Girshick, R., He, K., Dollar, P. (2017). Focal loss for dense object detection. In 2017 IEEE International Conference on Computer Vision (ICCV), Venice. https://doi.org/10.1109/ICCV.2017.324

[10] Vyavahare, P., Dhawale, S., Takale, P., Kolli, V., Kanawade, B., Khonde, S. (2023). Detection and interpretation of Indian sign language using LSTM networks. Journal of Intelligent Systems and Control, 2(3): 132-142. https://doi.org/10.56578/jisc020302

[11] Koller, O., Forster, J., Ney, H. (2015). Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. Computer Vision and Image Understanding, 141: 108-125. https://doi.org/10.1016/j.cviu.2015.09.013

[12] Wei, F., Zhou, S., Zhang, J., Liu, M. (2023). Improving continuous sign language recognition with cross-lingual signs. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1500-1510. https://doi.org/10.48550/arXiv.2308.10809

[13] Hu, L., Chen, X., Li, Y. (2024). Improving continuous sign language recognition with adapted image models.

arXiv preprint arXiv:2403.02456. https://doi.org/10.48550/arXiv.2404.08226
[14] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA. https://doi.org/10.1109/CVPR.2018.00474
[15] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., Berg, A.C. (2016). SSD: Single shot MultiBox detector. European Conference on Computer Vision, Amsterdam, The Netherlands, pp. 21-37. https://doi.org/10.1007/978-3-319-46448-0_2
[16] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934
[17] Dewi, C., Nataliani, Y., Wellem, T., Chernovita, H.P., Somya, R., Christanto, H.J., Gunawan, L.S., Andika, R.A., Raynaldo. (2024). Accurate hand recognition with neural architecture search technology. International Journal of Computational Methods and Experimental Measurements, 12(4): 421-428. https://doi.org/10.18280/ijcmem.120411
[18] Palanivel, S., Somasundaram, K. (2019). Tamil sign language recognition using histogram of oriented gradients and SVM. In Proceeding International Conference Computing and Communication Systems (I3CS'19), pp. 112-118.
[19] Radhakrishnan, R., Deivanai, S. (2020). Recognition of Tamil sign language alphabets using CNN. International Journal of Advanced Computer Science (IJACS), 10(4): 56-62.
[20] Anandh, K., Jayanthi, G. (2021). Real-time Tamil sign language recognition using deep learning and OpenCV. In International Conference Smart Technologies and Systems (ICSTS), pp. 245-250.
[21] Thangavel, N., Nithya, R. (2022). Dynamic Tamil sign language gesture recognition using LSTM networks. International Journal Emerging Trends in Engineering Research (IJETER), 10(8): 121-128.
[22] Priya, S., Subashini, A. (2023). YOLO-based Tamil sign language detection for real-time applications. In International Conference Computer Vision and Intelligent Systems (ICCVIS), pp. 310-316.
[23] Zhang, Y., Jiang, X. (2024). Recent advances on deep learning for sign language recognition. Computer Modeling in Engineering & Sciences, 139(3): 2399-2450. https://doi.org/10.32604/cmes.2023.045731
[24] Fan, D., Yi, M., Kang, W., Wang, Y., Lv, C. (2024). Continuous sign language recognition algorithm based on object detection and variable-length coding sequence. Scientific Reports, 14(1): 1-21. https://doi.org/10.1038/s41598-024-78319-0
[25] Stoll, S., Camgoz, N.C., Hadfield, S., Bowden, R. (2020). Text2Sign: Towards sign language production using neural machine translation and generative adversarial networks. International Journal of Computer Vision, 128(4): 891-908. https://doi.org/10.1007/s11263-019-01281-2
[26] Saunders, B., Camgoz, N.C., Bowden, R. (2020). Progressive transformers for end-to-end sign language production. European Conference on Computer Vision, Glasgow, United Kingdom. https://doi.org/10.1007/978-3-030-58621-8_40
[27] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. https://doi.org/10.48550/arXiv.1704.04861
[28] Arora, R., Singh, P., Sharma, A. (2022). Real-time Indian sign language detection using SSD-MobileNet. In Proceedings of the International Conference on Signal Processing and Communication (SPICON), pp. 201-206. https://doi.org/10.1109/SPICON56577.2022.1018083

APPENDIX

Algorithm 1. SSD MobileNet Training Procedure

Require: Training set $\mathcal{D}$, anchors $\mathcal{A}$, batch size B, learning rate η, weight α, neg:pos ratio r, epochs E
Ensure: Trained parameters Θ
1: Initialize network parameters Θ
2: for epoch $= 1$ to E do
3: Shuffle $\mathcal{D}$
4: for each mini-batch B of size B do
5: Load images and annotations in B
6: Apply data augmentation (flip, scale, color jitter, etc.)
7: Forward pass → predictions $\mathcal{P}^{(a)}$, $t^{(a)}$ for all anchors $a \epsilon \mathcal{A}$
8: Match anchors to ground-truth → positives $\mathcal{P}$ and negatives
9: Compute target offsets $t^{(a)}$ for $a \epsilon \mathcal{P}$
10: Rank negatives by classification loss and select top-$K = min(r|\mathcal{P}|, negatives)$ as $\mathcal{N}$
11: $L_{loc} \leftarrow \sum_{a \epsilon \mathcal{P}} \sum_{i \epsilon \{x,y,w,h\}} smooth_{L_1\left(t_i^{(a)} - \hat{t}_i^{(a)}\right)}$
12: $L_{conf} \leftarrow -\sum_{a \epsilon \mathcal{P}} \log p_{c(a)}^{(a)} - \sum_{a \epsilon \mathcal{N}} \log p_0^{(a)}$
13: $L \leftarrow \frac{1}{max(1,|\mathcal{P}|)}\left(L_{conf} + \alpha L_{loc}\right)$
14: Backpropagate and update $\Theta \leftarrow OptimizerUpdate(\Theta, \nabla\Theta L, \eta)$
15: end for
16: end for
17: return Θ

Algorithm 2. SSD MobileNet Inference Procedure

Require: Image I, trained parameters Θ, anchors $\mathcal{A}$, score threshold τ, IoU threshold γ
Ensure: Detections S
1: Preprocess I (resize, normalize)
2: Forward pass → predictions $\mathcal{P}^{(a)}$, $t^{(a)}$ for all $a \epsilon \mathcal{A}$
3: Decode boxes $b^{(a)}$ from anchors and offsets $t^{(a)}$
4: For each anchor, compute label $C^{(a)} = \arg max_c \mathcal{P}_c^{(a)}$ and score $S^{(a)} = \arg max_c \mathcal{P}_c^{(a)}$
5: Construct candidates $\mathcal{C} = \left\{\left(b^{(a)}, c^{(a)}, s^{(a)}\right) \middle| s^{(a)} \geq \tau\right\}$
6: $S \leftarrow \emptyset$
7: for each class c do
8: $C_c \leftarrow$ detections in $\mathcal{C}$ with label c
9: $S_c \leftarrow NonMaxSuppression(C_c, \gamma)$
10: $S \leftarrow S \cup S_c$
11: end for
12: Optionally keep top-K by score
13: return S