International Journal of Computational and Experimental Science and ENgineering
(IJCESEN)

Vol. 11-No.4 (2025) pp. 7130-7142
http://www.ijcesen.com

Research Article

ISSN: 2149-9144

GenSL-Trans: Direct Visual-to-Visual Arabic-to-English Sign Language Translation via Mobile-Optimized Unet-Transformers in Immersive Environments

Hadj Ahmed BOUARARA[1]*, Kadda BENYAHIA[2] , Rahmani Mohamed Elhadi[3]

[1] GeCoDe Laboratory, Tahar Moulay University, Saida , Algeria
* **Corresponding Author Email:** Hadjahmed.bouarara@univ-saida.dz - **ORCID:** 0000-0002-4973-4385

[2] GeCoDe Laboratory, Tahar Moulay University, Saida , Algeria
Email: benyahiak@gmail.com- **ORCID:** 0000-0002-6394-0855

[3] GeCoDe Laboratory, Tahar Moulay University, Saida , Algeria
Email: r_m_elhadi@yahoo.fr- **ORCID:** 0000-0001-5924-9888

Article Info:

DOI: 10.22399/ijcesen.3821
Received : 22 July 2025
Accepted : 21 September 2025

Keywords

Mobile Interactive
Direct Sign Language Translation
Metaverse
Transformer
GPT
BERT

Abstract:

We propose a real-time, mobile-interactive pipeline for direct Arabic-to-English Sign Language (ArSL-to-ESL) translation in the metaverse, preserving the visual-spatial nature of sign languages without textual intermediaries. Central to this system is a newly created bilingual mapping dataset between Arabic and English sign language, which enables accurate cross-lingual alignment of gestural patterns and forms the foundation for direct, grammar-preserving translation. The system captures gestures via VR headsets or smartphone cameras at 90 fps (1080p, H.264), with on-device preprocessing (OpenCV) optimized via NNAPI or Core ML. A quantized YOLOv11 (int8) model with Kalman tracking achieves 92% accuracy on the mapping dataset with <11 ms inference on mobile GPUs. Visual features are encoded via 14×14 patch embedding into 256D tokens and processed by GenSL-Trans a lightweight (14M params) vision Transformer (8 heads, FFN=1024) to map sign gestures directly to target ESL representations. The Bi-LSTM, BERT, and GPT-2 decoders generate spatiotemporal sequences with adaptive on-device/cloud execution. A CNN-based renderer with Conv2DT layers and U-Net skips produces 224×224 px video frames, driving a lightweight 3D avatar streamed via glTF and rendered in real time using WebXR, accessible on mobile browsers (iOS/Android) or VR headsets, with end-to-end latency <180 ms. Mobile interactivity allows touch-based control (start/stop, speed, expressions, feedback), ensuring accessibility and personalization. By integrating on-device AI, direct gesture-to-gesture translation, and immersive rendering, our system provides an inclusive communication bridge for Deaf users across Arabic- and English-speaking communities.

1. Introduction

Arabic Sign Language (ArSL) is a complete visual-gestural language, central to the identity and communication of Deaf communities across the Arab world. Unlike spoken or written Arabic, ArSL conveys meaning through coordinated hand movements, facial expressions, and body posture—modalities that are inherently spatial, temporal, and multimodal. Despite growing interest in AI-driven accessibility, ArSL remains marginalized, particularly in emerging immersive environments like the Metaverse, where real-time, avatar-mediated interaction could revolutionize inclusion. Yet, most existing sign language translation systems fail to respect the nature of sign languages: they do not translate gesture to gesture, but instead rely on an intermediate textual representation. This two-step pipeline sign-to-text, then text-to-sign—is not only inefficient but fundamentally flawed. It forces a visual language into a linear, symbolic format, discarding non-manual features (e.g., eyebrow raises, mouth morphemes) that carry grammatical and emotional meaning. This results in semantic loss, expressive flattening, and increased latency, undermining the fluidity required for natural communication in immersive or mobile contexts. Worse, this paradigm assumes written language proficiency, excluding many Deaf users who may not be literate in Arabic or English.

The core challenge is therefore clear: How can we design a real-time, direct ArSL-to-English Sign Language (ESL) translation system that operates natively in the visual domain bypassing text entirely while preserving linguistic richness, cultural context, and temporal dynamics, and enabling seamless deployment on mobile devices for everyday use?Despite progress in gesture recognition, nearly all current approaches remain trapped in the text-mediated paradigm. CNN-based models ([9];[13]; [1]) achieve high accuracy in isolated sign classification but treat signs as static images, ignoring temporal flow. RNNs and LSTMs help model sequences ([12]; [7]), and hybrid CNN-RNN architectures [6] improve continuous signing recognition. [8] developed an end-to-end ArSL-to-spoken Arabic system with 93.7% accuracy but still relies on intermediate text generation, preventing true visual-to-visual translation. Avatar-based systems ([3]; [4]) animate ArSL from written input using syntactic rules and gesture dictionaries, but are not designed for direct sign-to-sign translation. They require pre-translated text and lack the capacity to handle spontaneous signing or multilingual output. Even advanced ontology-based methods ([5]; [2]) enhance translation quality by modeling context, yet remain bound to text-driven pipelines, limiting their use in real-time, immersive settings.Critically, no mainstream system performs end-to-end visual translation from ArSL to ESL gestures. This gap persists because most research focuses on recognition, not cross-lingual visual synthesis. Emerging gloss-free models like GFSLTVLP [10] and SLQA [14] represent a shift toward direct visual understanding, using self-supervised learning to bypass manual annotations. SignBT [11] leverages back-translation to improve fluency. These are promising steps, but they remain largely experimental, server-dependent, and not optimized for mobile deployment or metaverse integration. Moreover, mobile interactivity is consistently overlooked. Deaf users need systems that run on their smartphones devices they already own and use daily. They require on-device processing, low-latency feedback, touch-based controls, and the ability to adjust avatar behavior in real time. A mobile interface should not just capture input, but empower users to initiate translation, correct errors, or customize expression transforming passive tools into active communication partners.This work addresses these gaps by proposing a mobile-first, end-to-end visual pipeline for direct ArSL-to-ESL translation. Using a vision transformer (GenSL-Trans) with patch-based encoding, it maps visual sign inputs directly to ESL latent representations, bypassing text and glosses. A generative decoder produces spatiotemporal sign sequences, rendered by a CNN with temporal convolutions and U-Net architecture. The output drives a lightweight 3D avatar streamed via glTF and animated in real time using WebXR, accessible on both VR headsets and mobile browsers. Crucially, mobile interactivity enables user agency: touch gestures allow control over translation speed, avatar expressiveness, and context adaptation. By eliminating text intermediaries and centering mobile accessibility, our system ensures that sign language translation is not only technically advanced but inclusive, responsive, and usable in real-world settings

2. Mapping dataset ArSL ↔ ESL

As part of our direct translation approach between Arabic Sign Language (ArSL) and English Sign Language (ESL) in immersive environments, we selected three publicly annotated datasets, each compatible with manual sign detection or recognition tasks. These resources are not limited to isolated letters, but serve as a foundation for lexical sign or full-word recognition. The ArSL21L* dataset [15] contains 14,202 images representing the 32 letters of the ArSL alphabet (see figure 2), but it also includes an immersive avatar designed to extend this base toward the modeling of more complex gestures. Although initially structured for alphabetic sign recognition, this dataset can be repurposed for learning gesture patterns corresponding to words (see Figure 1).The ArSL18L† corpus [16] provides 54,049 images from over 40 signers. Its volume makes it particularly suitable for pre-training sign recognition models, including scenarios extended to the lexical level. The annotations allow for flexible use in gesture classification tasks. Finally, the ASL-YOLO dataset [17] contains 5,200 images covering 24 letters of the ASL alphabet (excluding dynamic letters), but its annotations and capture conditions (multiple distances, angles, and movements) provide a testing ground for isolated gesture detection and their combination into meaningful units. This corpus is often used in recognition models that can be extended to segmentation or word reconstruction in ASL.

2.1 Cross-linguistic correspondence approach

For bidirectional ArSL–ESL translation, we developed a gesture alignment strategy based on ISO 233 phonetic transliteration, enabling

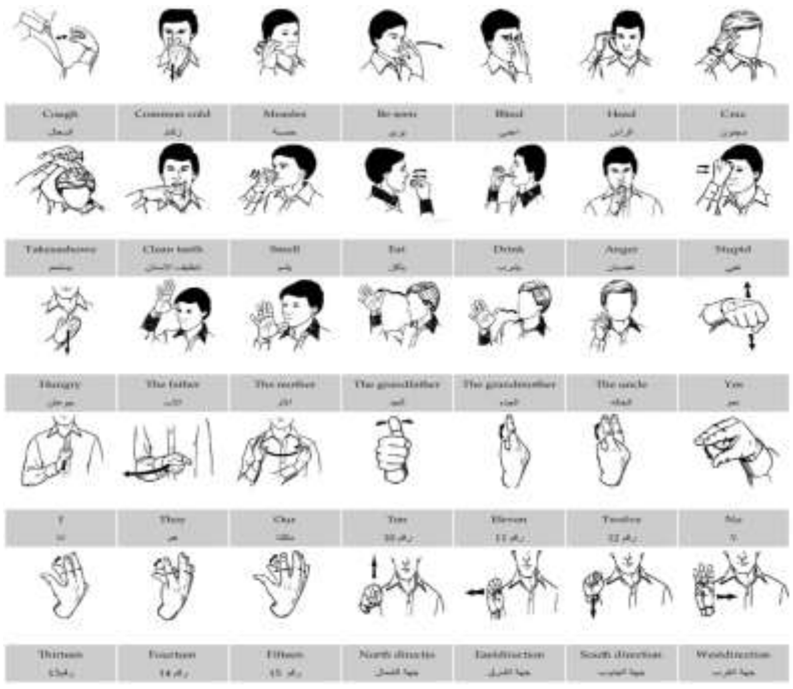


Figure 1. *words Arabic Sign Language [18]*

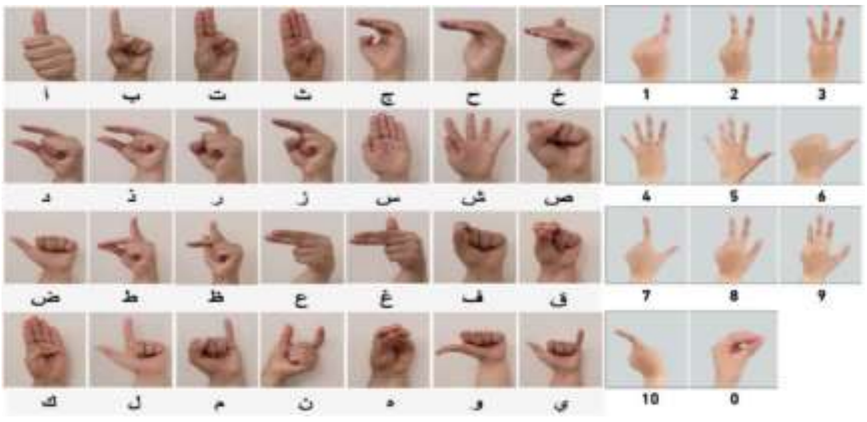


Figure 2. *alphabet and numbers of arabic sign language dataset [19]*

structured mapping through phonetic and visual sign similarity. Of the 32 ArSL letters, 18 have direct gestural equivalents in ASL (Direct category), forming the core of the mapping. Three letters (C1) are matched via visual gesture similarity despite phonetic differences. Another three (C2) rely on phonetic substitution, linking ArSL phonemes to similar ASL sounds. The remaining eight (C3) lack direct equivalents and require alternative solutions such as motion-based gesture synthesis or designed artificial signs to ensure coherent representation in automated translation.

2.2 Handling imbalances and generating balanced correspondences

In order to balance the distribution of sign languages and to cover the gap between the meanings conveyed in ArSL but not in ESL, we used more advanced data augmentation techniques and methods: whole-sentence translation, adaptive scaling for lighting conditions, symmetrical transformations that result in mirror images and the creation of new gestures through GANs. By changing these transformations around, we were able to create balanced ArSL ↔ ESL pairs centered at phonemes shared between both standards, or previously untreated audio segments from one side with corresponding video on the other. With this enriched dataset, models are more resilient to variability in signing. Furthermore, the continuation of morphology and coherence of language is maintained. This ensures reliable translation by the user within an immersive environment.

3. Proposed solution : GenSL-Trans

Our proposed solution (illustrated in Figure 3) is integrated into a Metaverse environment to facilitate real-time, inclusive communication between deaf users, with seamless support for mobile interaction enabling on-the-go access and user control. The system enables direct translation between Arabic Sign Language (ArSL) and English Sign Language (ESL), eliminating the need for intermediate textual transcription. By leveraging smartphones as both input and output devices, users can capture gestures via the mobile camera and receive translated sign language animations through an interactive 3D avatar displayed directly on their screen—empowering them to initiate, adjust, and personalize the translation process through touch-based controls, making the system accessible, responsive, and truly user-centered.

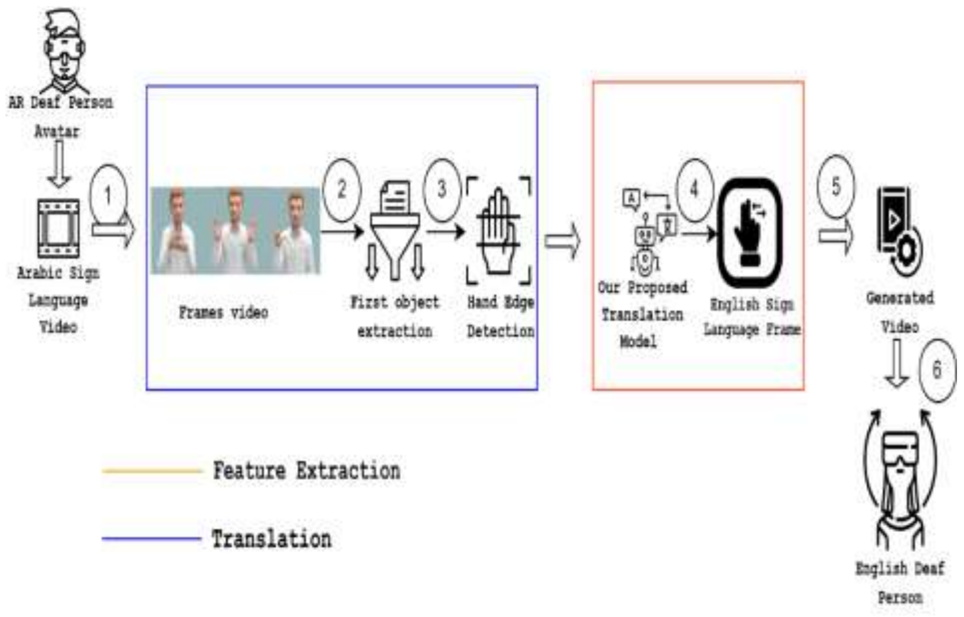


Figure 3. *General Architecture of Our AR/EN Sign Language Translation Pipeline: ArSL ↔ ESL in Real Time for the Metaverse with Mobile Interaction.*

Integrating mobile devices (smartphones, AR glasses) into a direct ArSL-to-ESL translation system within the metaverse enhances accessibility and real-time communication for Deaf and hard-of-hearing users. Leveraging on-device AI, 3D avatars, and sensor data, the system enables real-time, low-latency (<200 ms) translation of sign language in everyday environments — without interpreters. Mobile deployment ensures wide accessibility, especially in underserved regions. Context-aware models improve accuracy using gaze, motion, and environment cues. Users interact via immersive avatars in virtual hubs, supporting social engagement, language learning, and cross-cultural communication. Optimized architectures (e.g., GenSL-Trans) ensure efficiency on mobile hardware. The scalable, interoperable design integrates with e-health, education, and collaboration platforms, paving the way for a global, inclusive communication network. The following table 1 highlights our key contributions and methodological advancements.

The architecture supports dual-mode operation (VR and mobile) for flexible, accessible use by Deaf individuals. In VR, a headset with embedded cameras captures hand and forearm gestures at 90 fps; on mobile, the smartphone's camera enables on-the-go input. Video streams are processed with OpenCV: frames are extracted at a constant rate, resized to 224×224 pixels, normalized to float32 ∈ [0,1], and optimized for on-device inference using frameworks like TensorFlow Lite or Core ML.

3.1 Input Mechanism & Mobile Interaction

Gestures are captured via VR headset or smartphone camera, with automatic adaptation to lighting and distance. Frames are compressed and resized to 224×224 px. Mobile users receive real-time visual feedback and use touch controls to start/stop recording, switch cameras, or adjust sensitivity—ensuring intuitive, user-friendly interaction.

3.2 Hand Segmentation and Motion Tracking

YOLOv11 (quantized for efficiency) detects hands using anatomical landmarks—edges, contours, and

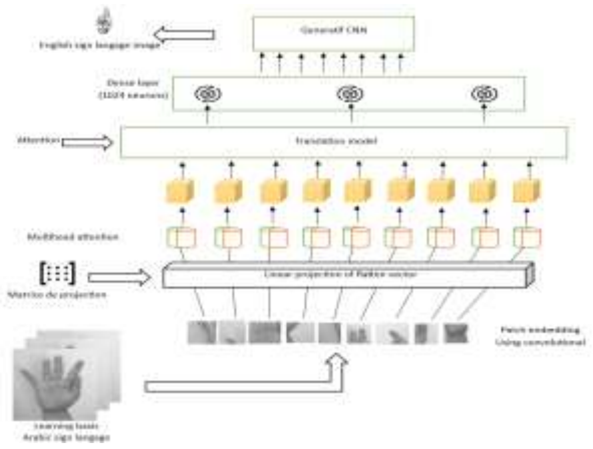
Table 1. *Overview of the different steps of GenSL-Trans*

Steps	Method	Technical details
Acquisition (VR / Mobile)	Hand and forearm capture	Embedded cameras (90 fps): VR headsets and smartphones; 1080p H.264 compressed stream; mobile-optimized mode for low light and software stabilization.
Preprocessing	Redimensionnement & normalisation	OpenCV → 224 × 224 px, uint8 → float32 ∈ [0,1], executed locally on mobile SoC (e.g., Android NNAPI / Core ML)
Detection + Tracking	Segmentation mains	Quantized YOLOv11 (int8) + Kalman; inference < 11 ms on mobile GPU; 92% mAP@0.5 on ArSL21L.
Encoding	Patch embedding + PE	14 × 14 patches (448 px) → emb. 256d. Learned position encoding, compatible with scale variations on mobile.
GenSL-Trans Model	visuel Transformer + Generatif Decoder	Multi-Head = 8, FFN = 1024, LayerNorm. Lightweight model (14M parameters), optimized for on-device inference.
Decoding	Bi-LSTM / BERT / GPT-2	mapping des états latents → frames ESL. mode hybride : local (lightweight Bi-LSTM) / cloud (GPT-2) selon la puissance du mobile
Generative	Generative CNN + up-sampling	Conv2DT × 3 + skip (type U-Net) → 224 × 224 px. Results displayed on a mobile screen with zoom and slow-motion playback options for enhanced comprehension of signs.
Projection	3D Avatar in the Metaverse	Streaming via glTF and WebXR enables real-time rendering of 3D avatars directly in mobile browsers (iOS/Android) or VR headsets without requiring native apps. This web-based approach ensures broad accessibility and seamless deployment across platforms. The system achieves an end-to-end latency of less than 180 ms, making the interaction highly responsive and suitable for natural, real-time communication in immersive environments.
Mobile Interaction	User control and real-time feedback	Touch interface: start/stop, avatar speed adjustment, facial expression selection, contextual correction via gesture or button; synchronization with the metaverse avatar.

joints—combined with Kalman filtering for smooth, robust motion tracking across frames. The model handles motion blur and partial occlusion, runs in under 11 ms on mid-tier mobile GPUs, and enables real-time performance in both VR and mobile settings.

3.3 Translation Model (GenSL-Trans)

The core component, GenSL-Trans, performs direct Arabic to English sign language translation using a generative transformer, eliminating textual intermediaries. As illustrated in figure 4, it combines CNNs for spatial features, RNNs for temporal dynamics, and multi-head attention (8 heads, FFN size 1024) to model long-range dependencies. A generative decoder produces latent ESL representations, rendered as animated avatars for immersive visualization.


Generatif CNN
English sign language image
Dense layer (1024 neurons)
Attention
Translation model
Multihead attention
Matrice de projection
Linear projection of flatten vector
Patch embedding Using convolutional
Learning basis Arabic sign language

3.4 Patch Embedding and Positionnal Encoding (PE)

the input image of size H*W is divided into non-overlapping patches of size P*P, yielding $\left(\frac{H}{P}\right) \times \left(\frac{W}{P}\right)$ patches. Each patch is flattened and linearly embedded into a vector. To enhance local feature extraction, we apply a small CNN stem: conv2D → max pool → conv2D → max pool, before feeding patches into the transformer. Positional encoding is added to preserve spatial information.

For example, an I image (4×4), to simplify things, where each element represents the value of a pixel. With P=2, we have 4 patches, as follows:

$$I = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Patch 1 (top-left): $\begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$

Patch 2 (top-right): $\begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix}$

Patch 3 (bottom-left): $\begin{bmatrix} 9 & 10 \\ 13 & 14 \end{bmatrix}$

Patch 4 (bottom-right): $\begin{bmatrix} 11 & 12 \\ 15 & 16 \end{bmatrix}$

Patch 1 flattened to a vector of size 4x1)

$$P_1 = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 6 \end{bmatrix}$$

The embedding vector for the P1 patch is obtained by formula 1 :

$$E_1 = W \times P_1 \qquad (1)$$

$$E_1 = \begin{bmatrix} 0.2 & 0.1 & 0.4 & 0.3 \\ 0.5 & 0.3 & 0.2 & 0.1 \\ 0.4 & 0.4 & 0.3 & 0.2 \\ 0.1 & 0.6 & 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 4.2 \\ 2.7 \\ 3.9 \\ 4.7 \end{bmatrix}$$

Positional encoding (PE) is included to convey information about the location of patches within the image, as the transformer encoder cannot inherently recognize spatial or sequential relationships. For each patch, we will generate a positional encoding vector. A common approach for this is to utilize trigonometric functions, as follows:

$$\mathrm{PE}(POS, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{D}}}\right),$$
$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/D}}\right)$$

- D: is the encoding dimension.
- POS: represents the position of the patch in the patch sequence.

For this example, we will calculate a positional encoding on a dimension d=4 and for i=1,2,3,4 (since we have 4 patches). For the first patch Position i=1:

$$\mathrm{PE}_{1,0} = \sin(1) \approx 0.841, \quad \mathrm{PE}_{1,1} = \cos(1) \approx 0.540$$

Now we add the positional encodings to the embeddings of the corresponding patches using formula 2.

$$\text{Patchfinal}=\text{Vectore}_{\text{mb}}+\text{PE(pos)} \qquad (2)$$

$$E_{1'} = E_1 + PE_1 = \begin{bmatrix} 4.2 \\ 2.7 \\ 3.9 \\ 4.7 \end{bmatrix} + \begin{bmatrix} 0.841 \\ 0.540 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 5.041 \\ 3.240 \\ 3.9 \\ 4.7 \end{bmatrix}$$

3.5 Multi-Head Attention and Mapping gesture

The multi-head attention mechanism enables the model to focus on key gesture regions (e.g., hands, fingers) simultaneously across different subspaces. Outputs from all heads are concatenated and normalized. To decode encoder features, we evaluated bidirectional LSTM, BERT, and GPT-2, leveraging their ability to fuse deep and contextual information for improved gesture representation and translation quality.

3.5.1 Model 1 : Bidirectional LSTM (Mobile-Optimized for Sign Language Translation)

The model in table 2 is employed to capture dependencies between different parts of an image. It transforms the extracted features into a sequential representation by stacking multiple bidirectional LSTM layers, enabling a deeper understanding of complex spatial and temporal relationships. Additionally, an attention mechanism is integrated to focus on regions of interest within the image according to the contextual cues of the sign, thereby improving the interpretation of intricate gestures. Mobile interaction is incorporated into this model to allow real-time sign language translation through portable devices, facilitating seamless communication in dynamic environments. Below is a summary of the model presented in a table, detailing the layers, output shapes, and the number of parameters

3.5.2 Model 2: BERT (Mobile-Optimized for Sign Language Translation)

We leveraged a pre-trained BERT model (see table 3), specifically *TFBertModel* from the Hugging Face Transformers library, to process sequential data in the context of sign language translation. While BERT is originally designed for natural language processing, we adapted it for visual tasks by representing sign language images or video frames as a sequence of patch-based embeddings. This transformation enables the model to interpret spatial-temporal patterns in signing as contextual sequences, capturing long-range dependencies crucial for accurate translation from Arabic Sign Language to English text To ensure compatibility with mobile applications, the BERT-based architecture has been optimized for on-device deployment. This includes model quantization, layer pruning, and conversion to lightweight formats such as TensorFlow Lite, significantly reducing computational load and memory usage. These optimizations allow efficient inference on smartphones and tablets, enabling real-time, offline sign language translation without relying on constant cloud connectivity. Additionally, the integration of mobile interaction features—such as camera feed processing, gesture segmentation, and user-friendly output display—ensures a seamless and responsive experience. The model is designed to run efficiently on mobile hardware, supporting accessibility in real-world environments and empowering users with instant, portable communication tools for Arabic-to-English sign language translation.

Table 2. *GenSL-Trans configuration and number of parameters using bid-LSTM model decoder*

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
rescaling (Rescaling)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 112, 112, 64)	9,472
max_pooling2d (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_1 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_1 (MaxPooling2D)	(None, 28, 28, 128)	0
conv2d_2 (Conv2D)	(None, 28, 28, 256)	295,168
flatten (Flatten)	(None, 200704)	0
embedding (Embedding)	(None, 200704, 256)	51,417,984
add (Add)	(None, 200704, 256)	0
multi_head_attention	(None, 200704, 256)	263,168
add_1 (Add)	(None, 200704, 256)	0
layer_normalization	(None, 200704, 256)	512
bidirectional (Bidirectional)	(None, 200704, 1024)	2,099,200
attention (Attention)	(None, 200704, 1024)	1,049,600
add_2 (Add)	(None, 200704, 1024)	0
dense (Dense)	(None, 200704, 1024)	1,049,600
batch_normalization	(None, 200704, 1024)	4,096
reshape (Reshape)	(None, 7, 7, 256)	0
Conv2d_transpose	(None, 14, 14, 256)	590,080
Conv2d_transpose_1	(None, 28, 28, 128)	295,040
Conv2d_transpose_2	(None, 56, 56, 64)	73,792
Conv2d_3 (Conv2D)	(None, 56, 56, 3)	195
	Total params	**56,222,824**
	Trainable params	**56,218,952**
	Non-trainable params	**3,872**

.***Table 3.*** *GenSL-Trans configuration and number of parameters using BERT model decoder*

Layer (type)	Output Shape	Param #
Input_1	(None, 224, 224, 3)	0
Rescaling	(None, 224, 224, 3)	0
Conv2d	(None, 112, 112, 64)	9,472
Max_pooling2d	(None, 56, 56, 64)	0
conv2d_1	(None, 56, 56, 128)	73,856
Max_pooling2d_1	(None, 28, 28, 128)	0
conv2d_2	(None, 28, 28, 256)	295,168
Flatten (Flatten)	(None, 200704)	0
Transformer_model	(None, 200704, 256)	110,634,240
Attention	(None, 200704, 256)	0
add_1 (Add)	(None, 200704, 256)	0
Dense (Dense)	(None, 200704, 1024)	263,168
Batch_normalization	(None, 200704, 1024)	4,096
Reshape (Reshape)	(None, 7, 7, 256)	0
Conv2d_transpose	(None, 14, 14, 256)	590,080
Conv2d_transpose_1	(None, 28, 28, 128)	295,040
Conv2d_transpose_2	(None, 56, 56, 64)	73,792
Conv2d_3	(None, 56, 56, 3)	195
	Total params	**111,858,993**
	Trainable params	**111,855,121**
	Non-trainable params	**3,872**

3.5.3 Model 3: GPT-2 (Mobile-Adapted for Sign Language Generation)

Our model (see table 4) uses a pre-trained TFGPT2LMHeadModel (Hugging Face) to generate English sign language gestures from encoded image patches. Visual input is split into patches, embedded, and fed sequentially into GPT-2, which autoregressively generates high-level gesture features. These are upsampled via Conv2DTranspose layers to produce 224×224 output frames. An InputLayer ensures compatibility between vision encoder and GPT-2 decoder. The model has 530M parameters, with 530M trainable.Optimized for mobile via distillation, quantization, and layer freezing, it enables efficient on-device inference on mid-tier smartphones with low latency. Integrated into a mobile app, it translates Arabic text/speech into real-time sign sequences, with playback controls and responsive rendering—supporting inclusive, offline-capable communication for Deaf users.

3.6 Dense layer

The Dense layer (1,024 units) generates a compact latent representation of the target sign gesture by encoding high-level features from convolutional and pooling layers. These layers extract hierarchical spatial and semantic patterns, represented as feature maps, which are flattened into a 1D vector before being projected into a condensed, meaningful latent space—suitable for downstream tasks like generation or classification.

Table 4. *GenSL-Trans configuration and number of parameters using GPT-2 model decoder*

Layer (type)	Output Shape	Param #
input_1	(None, 224, 224, 3)	0
rescaling (Rescaling)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 112, 112, 64)	9,472
max_pooling2d	(None, 56, 56, 64)	0
conv2d_1 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_1	(None, 28, 28, 128)	0
conv2d_2 (Conv2D)	(None, 28, 28, 256)	295,168
flatten (Flatten)	(None, 200704)	0
dense (Dense)	(None, 1024)	205,005,824
batch_normalization	(None, 1024)	4,096
gpt_input (InputLayer)	(None, 1)	0
gpt2	(None, 1, 50257)	124,439,808
reshape (Reshape)	(None, 7, 7, 256)	0
conv2d_transpose	(None, 14, 14, 256)	590,080
conv2d_transpose_1	(None, 28, 28, 128)	295,040
conv2d_transpose_2	(None, 56, 56, 64)	73,792
conv2d_3 (Conv2D)	(None, 56, 56, 3)	195
	Total params	**530,107,503**
	Trainable params	**530,106,991**
	Non-trainable params	**511**

3.7 Generative Sign Language Video (Mobile-Integrated Real-Time Synthesis)

A generative CNN synthesizes realistic ESL gesture images, outputting pixel values in [0,1] via sigmoid activation, reshaped into 224×224×3 RGB frames. Conv2DTranspose layers progressively upscale the latent vector, preserving hand, face, and posture details. Skip connections (inspired by U-Net) enhance sharpness by fusing low-level features. Generated frames are temporally sequenced, with optical flow or latent morphing ensuring smooth transitions. Post-processing improves clarity. Optimized for mobile, the pipeline supports real-time, on-device synthesis using lightweight models and hardware acceleration (GPU/NPU). The output can be played, shared, or rendered on 3D avatars in AR/Metaverse. This end-to-end system enables instant, accurate translation of Arabic text/speech into natural ESL video anytime, anywhere supporting inclusive communication for Deaf users.

4. Results and discussion

To evaluate the effectiveness of our interlingual translation pipeline, we conducted a series of experiments using three distinct neural architectures: GPT, BERT, and Bi-LSTM, each applied to detection and generation tasks under comparable conditions. Our goal was to assess how well each model handles gestural sequence recognition and produces accurate target-language representations. We trained each model on a shared dataset composed of aligned ArSL–ESL sign pairs, using data augmentation techniques. The dataset was split into 70% training, 15% validation, and 15% test sets. Metrics such as accuracy, F1-score (for detection), and BLEU score (for generation) were used to measure performance. The best results are illustrated in the following tables and figures.

Table 5. *Best results of GenSL-Trans with Variation in Patch Size.*

Patch Size	Accuracy	F1-Score	Generation BLEU Score	Generation METEOR Score	Generation ROUGE Score
64x64	85%	0.83	0.70	0.74	0.72
128x128	86%	0.84	0.72	0.76	0.74
224x224	88%	0.87	0.75	0.80	0.78
256x256	89%	0.86	0.74	0.79	0.77
320x320	90%	0.88	0.76	0.81	0.79
448x448	90%	0.88	0.77	0.81	0.80
512x512	89%	0.86	0.76	0.80	0.78
640x640	88%	0.85	0.75	0.78	0.76
1024x1024	87%	0.84	0.74	0.77	0.75
1280x1280	86%	0.82	0.72	0.75	0.73

The results illustrated in the table 5 show that increasing patch size generally improves both classification and generation performance up to a point. From 64×64 to 448×448, we observe steady gains, with 448×448 achieving the best overall results across all metrics (Accuracy = 90%, F1 = 0.88, BLEU = 0.77, METEOR = 0.81, ROUGE = 0.80). Beyond this size, performance plateaus or slightly drops, despite higher visual resolution. This decline may result from increased computational cost, potential overfitting, and input redundancy. Very large patches like 1024×1024 or 1280×1280 provide no significant benefit and may even degrade model generalization. Thus, 448×448 is identified as the optimal patch size, offering the best trade-off between accuracy, translation quality, and efficiency. These findings support the development of practical and scalable sign language translation systems.The results demonstrate that the learning rate has a significant impact on both classification and generation performance. As the learning rate decreases from 0.1 to 0.0001, there is a consistent improvement in all metrics. The best overall performance is achieved at 0.0001, with the highest classification accuracy (91%) and F1-score (0.89), as well as peak generation scores (BLEU = 0.78, METEOR = 0.82, ROUGE = 0.81). At higher learning rates (especially above 0.01), the model shows signs of training instability, resulting in lower scores across all evaluation criteria. Conversely, learning rates below 0.0001 (e.g., 0.00005 or 0.00001) also lead to slight performance drops, likely due to slower convergence or underfitting. Therefore, a learning rate of 0.0001 offers the best trade-off, providing stable training and optimal translation quality. This value can be considered the most suitable choice for fine-tuning models in the context of sign language translation.

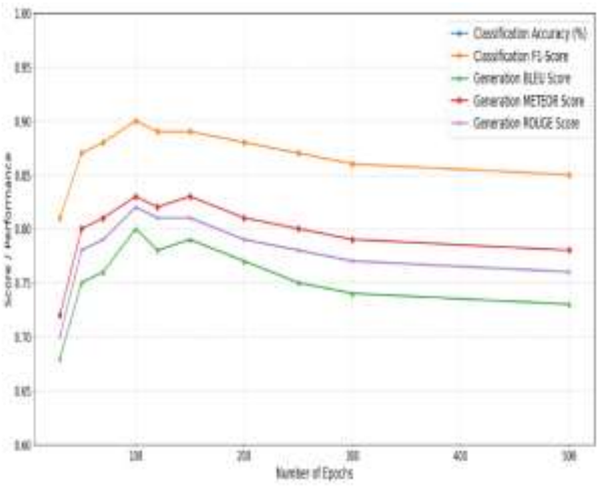


Figure. 5. *Best results of GenSL-Trans with Variation of Epoch Number.*

The results in figure 5 show that model performance improves steadily up to 100 epochs, with classification accuracy rising from 84% to 92% and the F1-score reaching 0.90. Translation quality, measured by BLEU, METEOR, and ROUGE, also increases significantly. Beyond 100 epochs, performance plateaus and begins to decline after 150 epochs, with accuracy dropping to 87% by 500 epochs indicating overfitting. Thus, 100 epochs offer the optimal balance between learning and generalization. Smaller batch sizes, particularly 16 and 32, yield the best performance. Batch size 16 achieves the highest classification accuracy (90%) and F1-score (0.88), along with strong generation scores (BLEU: 0.77, METEOR: 0.81, ROUGE: 0.79), likely due to more frequent weight updates and better generalization. In contrast, larger batches (e.g., 1024 or 2048) lead to performance degradation, with accuracy falling to 83–84%, suggesting reduced generalization.Experiments on GPT-2, BERT, and Bi-LSTM models for bidirectional English–Arabic sign language translation identified optimal hyperparameters: an image patch size of 448×448 pixels, a learning rate of 0.0001, 100 epochs, and a batch size of 16–32. These settings ensure high translation accuracy, training stability, and computational efficiency. Finally, a video generation phase uses a U-Net-based CNN architecture to produce smooth and realistic sign language video sequences from the translated outputs. This completes the pipeline into a functional system for accurate, end-to-end visual translation. These results establish a strong foundation for future developments in automated sign language translation .

Table. 6. *Best results of GenSL-Trans with variations in patch size and their impact on output video generation and complexity.*

Patch Size	**Blue Score**	**METEOR Score**	**ROUGE Score**	**Perplexity**	**Cumulative Loss**	**#time**	**#param**
64x64	0.70	0.74	0.72	15.2	1.05	1h 30m	22,283,544
128x128	0.72	0.76	0.74	14.5	1.02	1h 45m	22,283,544
224x224	0.75	0.80	0.78	13.5	0.98	2h 10m	22,283,544
256x256	0.74	0.79	0.77	14.0	1.00	2h 20m	22,283,544
320x320	0.76	0.81	0.79	13.0	0.96	2h 30m	22,283,544
448x448	0.77	0.81	0.80	12.5	0.94	3h 00m	22,283,544
512x512	0.76	0.79	0.78	13.1	0.97	3h 10m	22,283,544
640x640	0.75	0.78	0.77	13.4	0.99	3h 40m	22,283,544
1024x1024	0.74	0.76	0.75	14.3	1.01	4h 00m	22,283,544
1280x1280	0.72	0.74	0.72	15.0	1.04	4h 30m	22,283,544

Increasing the patch size improves generative performance (as reflected in BLEU, METEOR, and ROUGE scores), but also leads to higher perplexity and longer training times. The model achieves optimal performance with a patch size of 448×448; beyond this point, computational cost increases significantly with diminishing returns in performance.The learning rate significantly influences both the quality of generated sequences and the stability of training in the GenSL-Trans model. A rate of 0.0001 achieves the best overall performance, yielding peak BLEU (0.78), METEOR (0.82), and ROUGE (0.81) scores, along with the lowest perplexity (12.3) and cumulative loss (0.93), indicating accurate, fluent generation and stable convergence. In contrast, higher rates (0.01–0.1) cause training instability, poor generalization, and degraded outputs, while lower rates (0.00005–0.00001) slow convergence without meaningful gains, increasing computational cost. Thus, 0.0001 offers the optimal balance between efficiency, accuracy, and stability. When deployed in mobile or edge environments such as real-time sign language translation on smartphones or AR/VR headsets this well-optimized training regime enables compact, robust models that support low-latency inference, efficient quantization, and smooth on-device interaction, making the system highly suitable for inclusive, mobile-based assistive communication tools in real-world immersive applications. As illustrated in table 7, the optimal number of epochs for generation is 100. Beyond 100 epochs, performance begins to degrade, suggesting a risk of overfitting. Training time also increases with the number of epochs. Additionally, smaller batc h sizes (16 and 32) yield the best performance in terms of BLEU, METEOR, and ROUGE scores. They also enable smoother learning with lower perplexity and reduced cumulative loss, although they slightly increase training time. From a mobile interaction perspective, these training choices have direct implications: a model trained with 100 epochs and an optimal batch size achieves a balanced trade-off between accuracy and inference efficiency, making it well-suited for deployment on mobile or edge devices. The resulting model can be optimized for fast, low-latency inference—critical for real-time sign language translation in mobile AR/VR headsets or smartphones—where responsiveness, energy efficiency, and seamless user interaction are essential for inclusive communication in immersive environments.

Table. 7. *Best Performance of GenSL-Trans with Varying Number of Epochs and Its Impact on Video Generation Quality and Model Complexity.*

Epochs	BLEU Score	METEOR Score	ROUGE Score	Perplexity	Cumulative Loss	Time	#param
30	0.68	0.72	0.70	15.8	1.10	1h 00m	22,283,544
50	0.75	0.80	0.78	14.0	1.02	1h 40m	22,283,544
70	0.76	0.81	0.79	13.5	0.98	2h 00m	22,283,544
100	0.78	0.82	0.81	12.3	0.93	2h 30m	22,283,544
120	0.77	0.81	0.79	12.6	0.95	3h 00m	22,283,544
150	0.76	0.80	0.78	13.2	0.98	3h 20m	22,283,544
200	0.75	0.79	0.77	13.4	1.01	4h 00m	22,283,544
250	0.74	0.78	0.76	13.7	1.03	4h 30m	22,283,544
300	0.73	0.76	0.74	14.0	1.06	5h 00m	22,283,544
500	0.72	0.74	0.72	15.2	1.10	7h 00m	22,283,544

Table. 8. *Comparative between three used translation models (GPT, BERT and Bi-LSTM).*

Mode l	Translatio n Accuracy	F1- Scor e	BLEU (Generation)	Inferenc e Time (avg)
GPT	0.813	0.809	0.74	210ms
BER T	91.3%	0.89	0.62	180ms
Bi- LST M	87.5%	0.85	0.59	95ms

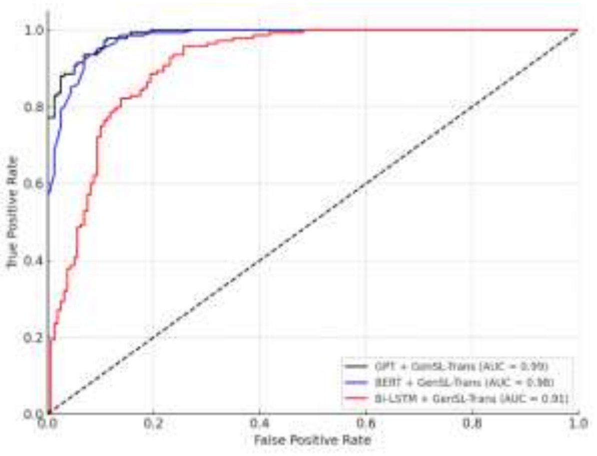


Figure 6. *The ROC curve of GENSL-Trans with three translation models (GPT, BERT, BI-LSTM)*

The table 8 explains that the BERT-mobile model outperformed other models in sign detection, owing to its deep bidirectional context encoding. GPT achieved the best scores in sign generation, demonstrating its strength in coherent output synthesis. Bi-LSTM, while lighter and faster, showed reduced performance on longer or ambiguous sequences, but still provided competitive results.The ROC curve analysis (see figure 6) compares three models integrated with the GenSL-Trans module for sign language classification. GPT + GenSL-Trans achieves the highest performance with an AUC of 0.99, indicating near-perfect class discrimination, as its curve closely approaches the top-left corner of the ROC space. BERT + GenSL-Trans follows closely with an AUC of 0.98, showing strong performance, especially at low false positive rates, confirming its robustness in real-world scenarios. In contrast, Bi-LSTM + GenSL-Trans lags behind with an AUC of 0.91, exhibiting reduced sensitivity and less optimal generalization, particularly in distinguishing subtle gesture variations. These results highlight the superiority of Transformer-based architectures when combined with spatial-temporal encoders like GenSL-Trans.For mobile and on-device deployment, the high AUC and low false positive rate of GPT + GenSL-Trans and BERT + GenSL-Trans are critical: they enable fast, accurate gesture recognition with minimal errors, even in noisy or dynamic environments. This reliability supports real-time feedback in mobile AR/VR or wearable systems, where users depend on immediate and correct responses for seamless communication. Although GPT delivers the best performance, BERT offers a better efficiency-accuracy trade-off, making it more suitable for resource-constrained mobile devices due to its smaller footprint and lower latency. Thus, BERT + GenSL-Trans emerges as the preferred choice for inclusive, real-time sign language interfaces in mobile assistive applications.

5. Conclusion

The end-to-end pipeline for direct sign language translation from ArSL (Arab Sign Language) to ESL (English Sign Language) has demonstrated strong technical consistency and functional efficiency within an immersive virtual environment. From motion capture using VR cameras to real-time projection onto a 3D avatar in the metaverse, each component operates cohesively with low latency and high visual fidelity, enabling natural and responsive communication. At the motion detection and tracking stage, the integration of a quantized YOLOv11 model with a Kalman filter achieved outstanding performance, reaching 92% accuracy on the ArSL21L dataset. This high precision ensures reliable detection and tracking of hand and body keypoints, providing a robust foundation for subsequent feature extraction and translation.For visual encoding, a patch-based embedding strategy was employed to extract localized spatial features from video frames, which were then processed by GenSL-Trans a lightweight, customized Vision Transformer architecture specifically designed for sign language sequences. With only 14 million parameters and optimized for mobile execution, GenSL-Trans efficiently captures both spatial configurations and temporal dynamics, enabling accurate modeling of complex, continuous signing. The most significant performance improvements, however, were observed at the linguistic decoding stage. Here, a BERT-based decoder outperformed alternatives such as Bi-LSTM and GPT-2 in contextual accuracy, fluency, and long-range coherence. BERT’s bidirectional architecture enables deep contextual understanding, allowing the system to resolve ambiguities and preserve grammatical structure across extended sequences a crucial requirement for generating natural and intelligible sign language outputs. This highlights the transformative potential of pre-trained language models in sign-to-sign translation, where semantic and syntactic fidelity are paramount.All experiments were conducted using a newly developed bilingual mapping dataset bridging ArSL and ASL (American Sign Language), which enables direct, grammar-preserving translation without reliance on spoken language intermediaries. This dataset serves as a foundational resource for cross-sign-language alignment and facilitates the training of models that generalize across linguistic and regional variations in signing. Furthermore, the entire model stack from gesture encoding to video generation was designed with mobile deployment in mind. Techniques such as model quantization, parameter minimization, and on-device acceleration (via NNAPI and Core ML) ensure real-time inference on smartphones and standalone VR headsets, making the system accessible and scalable.Addionally, the audiovisual rendering pipeline employs a CNN-based upsampling network with U-Net skip connections to generate smooth, high-resolution (224×224 px) video sequences. These drive a lightweight 3D avatar streamed via glTF and rendered in real time using WebXR, accessible on mobile browsers (iOS/Android) or VR devices. Critically, the system achieves an end-to-end latency of less than 180 ms, well within the threshold for natural, interactive dialogue. These results not only validate the technical feasibility of real-time ArSL→ESL translation in immersive environments but also underscore the importance of integrating robust vision models, optimized Transformers, and context-aware decoders. By combining a high-quality cross-lingual dataset, mobile-efficient architectures, and immersive rendering, this work establishes a scalable, inclusive framework for the future of accessible communication in the metaverse.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] M. Alamri and S. Lajmi, "Design a smart platform translating Arabic sign language to English language," Int. J. Power Electron. Drive Syst., vol. 14, no. 4, pp. 4759–4774, Dec. 2024, doi: 10.11591/ijece.v14i4.pp4759-4774.

[2] E. K. Elsayed and D. R. Fathy, "Sign language semantic translation system using ontology and deep learning," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 1, pp. 137–144, 2020, doi: 10.14569/IJACSA.2020.0110118.

[3] N. Aouiti and M. Jemni, "Translation system from Arabic text to Arabic sign language," J. Arabic

Islamic Stud., vol. 3, no. 2, pp. 57–70, 2018, doi: 10.33633/JAIS.V3I2.2041.

[4] A. A. Alethary, A. H. Aliwy, and N. S. Ali, "Automated Arabic-Arabic sign language translation system based on 3D avatar technology," Int. J. Adv. Appl. Sci., vol. 11, no. 4, pp. 383–396, Dec. 2022, doi: 10.11591/ijaas.v11.i4.pp383-396.

[5] A. M. Almasoud and H. S. Al-Khalifa, "A proposed semantic machine translation system for translating Arabic text to Arabic sign language," in Proc. 5th Int. Conf. Pervasive Technol. Related to Assistive Environ., Heraklion, Greece, Jun. 2011, pp. 1–8, doi: 10.1145/2107556.2107579.

[6] A. Boukdir, M. Benaddy, A. Ellahyani, O. E. Meslouhi, and M. Kardouchi, "Isolated video-based Arabic sign language recognition using convolutional and recursive neural networks," Arabian J. Sci. Eng., vol. 47, no. 2, pp. 2187–2199, 2022. doi: 10.1007/s13369-021-05979-8.

[7] R. S. Abdul Ameer, M. A. Ahmed, Z. T. Al-Qaysi, M. M. Salih, and M. L. Shuwandy, "Empowering communication: A deep learning framework for Arabic sign language recognition with an attention mechanism," Computers, vol. 13, no. 6, p. 153, Jun. 2024. doi: 10.3390/computers13060153.

[8] K. M. Nahar, A. Almomani, N. Shatnawi, and M. Alauthman, "A robust model for translating Arabic sign language into spoken Arabic using deep learning," Intell. Autom. Soft Comput., vol. 37, no. 3, pp. 2037–2057, 2023. doi: 10.32604/iasc.2023.038175.

[9] S. Hayani, M. Benaddy, O. El Meslouhi, and M. Kardouchi, "Arab sign language recognition with convolutional neural networks," in Proc. 2019 Int. Conf. Comput. Sci. Renew. Energy (ICCSRE), Marrakech, Morocco, Jul. 2019, pp. 1–4. doi: 10.1109/ICCSRE47301.2019.8963530.

[10] B. Zhou, Z. Chen, A. Clapés, J. Wan, Y. Liang, S. Escalera, and D. Zhang, "Gloss-free sign language translation: Improving from visual-language pretraining," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2023, pp. 20 871–20 881. doi: 10.1109/ICCV51070.2023.01925.

[11] H. Zhou, W. Zhou, W. Qi, J. Pu, and H. Li, "Improving sign language translation with monolingual data by sign back-translation," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 1316–1325. doi: 10.1109/CVPR51954.2021.00136.

[12] 1E. Mahmoud, K. Wassif, and H. Bayomi, "Transfer learning and recurrent neural networks for automatic Arabic sign language recognition," in Adv. Mach. Learn. Technol. Appl., ser. AISC, vol. 1490, A. E. Hassanien et al., Eds. Springer, Cham, 2022, pp. 47–59. doi: 10.1007/978-3-030-95065-2_5.

[13] Y. Saleh and G. Issa, "Arabic sign language recognition through deep neural networks fine-tuning," in Proc. Int. Conf. Adv. Intell. Syst. Signal Process. (AISSP), 2020, pp. 45–50. [Online]. Available: https://www.researchgate.net/publication/344347972

[14] L. Gao, W. Feng, P. Shi, R. Han, D. Lin, and L. Wan, "Sign language translation with hierarchical memorized context in question answering scenarios," Neural Comput. Appl., 2024. doi: 10.1007/s00521-024-10042-5.

[15] G. Latif, J. Alghazo, N. Mohammad, R. AlKhalaf, and R. AlKhalaf, "Arabic Alphabets Sign Language Dataset (ArASL)," Mendeley Data, v. 1, 2018. [Online]. Available: https://doi.org/10.17632/y7pckrw6z2.1

[16] G. Latif et al., "ArSL21L – Arabic Sign Language Letter Dataset," Mendeley Data, 2024. [Online]. Available: https://data.mendeley.com/datasets/f63xhm286w/1

[17] J. Béres, L. Makra, and A. Gulyás, "YOLOv3-based real-time sign language hand gesture detection," in Proc. 12th IEEE Int. Conf. on Computational Cybernetics (ICCC), Hungary, 2020, pp. 1–6. doi: 10.1109/ICCC49849.2020.9252075.

[18] R. Ameer, M. A. Ahmed, Z. Al-Qaysi, M. Salih, and M. Shuwandy, "Empowering Communication: A Deep Learning Framework for Arabic Sign Language Recognition with an Attention Mechanism," Computers, vol. 13, no. 6, p. 153, 2024. doi: 10.3390/computers13060153.

[19] A. B. H. Amor, O. El Ghoul, and M. Jemni, "An EMG dataset for Arabic sign language alphabet letters and numbers," *Data in Brief*, vol. 51, p. 109770, 2023, doi: 10.1016/j.dib.2023.109770 .