Breaking Barriers: A Multimodal Approach to Sign Language Accessibility

Ismail Hammami (357804), Youssef Torgeman (360550), Antoine Pelletier (356029)
COM-304 Final Project Report

***Abstract*—This project presents a sign language translation using the 4M architecture. By combining rgb video, pose video, and text data, the model enables flexible any-to-any modality translation. The system shows promising results in generating meaningful visual outputs, offering a step toward more accessible and inclusive communication for the deaf and hard-of-hearing community.**

I. Introduction

The ability to communicate effectively is foundational to human interaction, yet significant barriers persist for the deaf and hard-of-hearing community due to widespread unfamiliarity with sign languages. Despite their critical role in enabling communication for millions globally, sign languages remain inaccessible to most, isolating those who rely on them in everyday interactions. This lack of accessibility creates substantial obstacles in educational settings and daily social exchanges, underscoring a pressing need for technological solutions.

Our project directly addresses this critical challenge by developing a multimodal model capable of translating sign language into accessible textual and human pose formats in real-time. By integrating video streams, precise pose data, and textual annotations, our solution employs the 4M models and improved tokenizers to make an essential step towards a more inclusive and equitable society.

II. Related Work

Several prominent approaches have been proposed for Sign Language Translation (SLT), each addressing different aspects of the problem:

- **End-to-End Models** such as *Sign2Text* and *RWTH-PHOENIX-Weather* [1], [2] translate sign language videos directly into natural language text. These models eliminate intermediate steps but require extensive annotated datasets and often generalize poorly across diverse signers.
- **Pose-Based Models** like *TokenPose* [3] represent signs using human keypoints (e.g., hands, body joints), reducing visual variability and computational complexity. However, they may struggle to encode the full expressive richness of sign language when used alone.
- **SignGemma** [4], recently introduced by Google DeepMind in May 2025, employs vision transformers to perform real-time translation from sign to spoken text. Although impressive, its fixed unidirectional modality restricts its usefulness for more interactive or educational settings.

Our Approach

Our model builds upon the **4M architecture** [5], which supports massively multimodal masked modeling and enables true *any-to-any modality translation*. This architectural flexibility allows our system to translate across arbitrary combinations of input and output modalities including video, human pose, and text.

This is particularly powerful in an educational context. A learner can:

- Begin by writing a sentence in text, and have the model generate a video of a virtual signer performing the corresponding sign language.
- Later, record themselves signing and receive a generated caption to validate whether their signs were interpreted correctly.
- Use the integrated **pose modality**, powered by TokenPose-like keypoint tokenization, to isolate and visualize specific hand and body configurations providing fine-grained insight into which signs are correctly produced and which require adjustment.

Compared to prior work,transforms the SLT pipeline into a flexible, bidirectional learning assistant. This novel use of multimodality offers both accessibility and active pedagogy in one unified framework.

III. Method

To address the problem, our approach focused on leveraging a multimodal dataset to develop a robust model for sign language recognition, emphasizing data preprocessing, tokenization, and model training. Below, we outline the methodology, justify design choices, discuss alternatives, and highlight challenges encountered.

A. Dataset Selection

The first step was identifying a dataset that supports multiple modalities with strong alignment and coherence to ensure robust model performance. We selected the How2Sign dataset, which contains 30,000 samples, each including RGB videos, keypoint videos (human pose data), and JSON files per frame. These JSON

files provide coordinates for critical features such as `face_keypoints_2d`, `hand_left_keypoints_2d`, and `hand_right_keypoints_2d`. Additionally, the dataset includes aligned captions for each video, enabling multimodal learning. This dataset was chosen for its comprehensive modalities and alignment, which are critical for capturing the spatial and temporal dynamics of sign language.

B. Preprocessing and Tokenization Strategy

Preprocessing the dataset required an efficient tokenization strategy to transform video data into a format suitable for model training. Initially, we explored video tokenizers such as Vidtok and Cosmos Video Tokenizer. While these produced high-quality reconstructions, they were computationally expensive, taking approximately 15 seconds per sample, equating to roughly five days for the entire dataset. Moreover, memory constraints caused frequent crashes due to limited infrastructure. Given the time constraints, we pivoted to an alternative strategy: extracting keyframes from videos and using an image-based tokenizer (Cosmos).

We began by selecting five random frames from both RGB and keypoint videos. However, these frames often lacked sufficient information to capture distinct signs, as sign language relies on sequential gestures. To address this, we increased the number of frames to 17 and developed a motion-based keyframe extraction algorithm. This algorithm identifies frames with the highest motion changes, ensuring the selection of informative keyframes that capture critical sign language gestures.

This approach improved the informativeness of the selected frames compared to random selection, though it increased computational complexity slightly.

C. Tokenization Challenges and Solutions

For tokenization, we initially tokenized the original 1280x720 resolution frames using the Cosmos Image Tokenizer without resizing. While this yielded excellent reconstruction quality, it produced a high number of tokens (approximately 12 times more than with 256x256 resolution), making it computationally infeasible for training, especially with 17 frames per sample. Resizing frames to 256x256 reduced the token count but degraded reconstruction quality, particularly for critical features like hand signs and facial expressions.

To mitigate this, we experimented with adjusting color and contrast to enhance key features but found no significant improvement. Instead, we developed a cropping strategy to focus on the signer's face and hands while minimizing irrelevant background areas. Face detection was computationally expensive, so we applied it to a subset of samples, analyzed the signer's positioning, and generalized cropping coordinates across the dataset. This approach balanced computational efficiency with improved focus on relevant features. (See Figure 1)

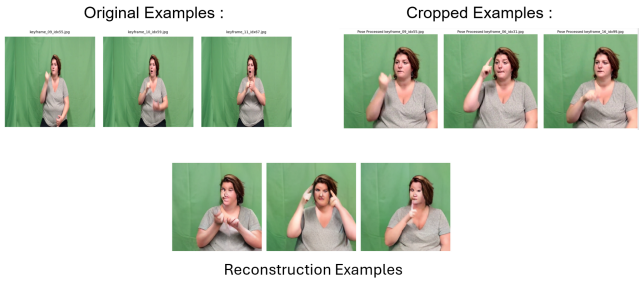

Figure 1. Some cropping results.

For keypoint videos, some keyframes lacked visible signs. To address this, we used the JSON coordinate data to enhance hand and face regions during reconstruction. By retaining frame indices during keyframe extraction, we remapped and emphasized these regions, improving keypoint reconstruction quality, as illustrated in Figure 2.

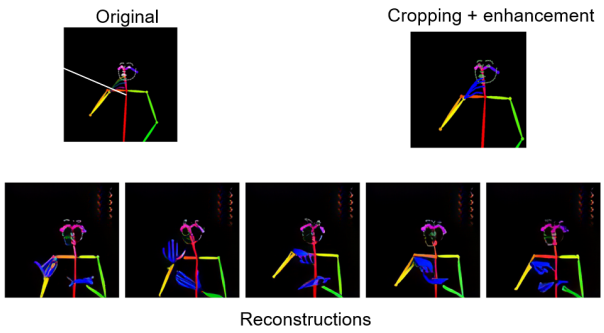

Figure 2. Keypoint enhancement and cropping results (placeholder).

D. Model Training and Enhancements

The tokenized data were then split into train, validation, and test set. The training set was used for model training. A key challenge was the model's inability to track token origins across frames. To address this, we introduced two positional embedding techniques:

- **Frame Positional Embedding**: To maintain the temporal order of frames.
- **Spatial Positional Embedding**: To preserve token locations within each frame.

These embeddings were integrated into the model architecture, and after tuning configurations, we achieved meaningful results. The embeddings ensured the model could effectively interpret the spatial and temporal relationships in the tokenized data.

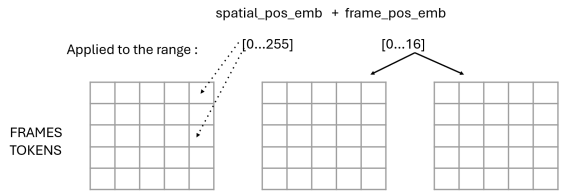

Figure 3. illustrative figure for the role of the frame positional embeddings and the spatial positional embeddings

E. Generation and Inference Strategy

Our method was to first identify the optimal set of hyperparameters number of steps, temperature, top-k, and top-p for generating poses (keypoints) directly from captions without intermediate modalities. This was done by conducting multiple generation experiments, varying one hyperparameter at a time to determine its optimal value. Each optimal value was then fixed for subsequent searches to optimize the remaining hyperparameters. Once the optimal settings for caption-to-keypoints generation were established, we used these results to tune other modality combinations, aiming to identify the best generation settings for any-to-any modalities generations.

F. Tokenizer Training

For each modality audio (initially planned), image/pose, and text we trained a standalone tokenizer to convert raw inputs into discrete token sequences. Specifically:

- **Audio Tokenizer:** We used Meta's EnCodec model, fine-tuned on the DailyTalk speech corpus to discretize waveform segments.
- **Image/Pose Tokenizer:** A VQGAN-style autoencoder was trained on our RGB frames.
- **Text Tokenizer:** A BPE-based GPT-2 tokenizer was adapted and trained on our transcript dataset.

G. Design Choices and Alternatives

Our design choices were driven by the need for efficiency and quality within the constraints of available infrastructure. The shift from video to image-based tokenization was a pragmatic response to computational limitations, though it sacrificed some reconstruction fidelity. Alternatives like VidTok were effective but impractical due to time and memory constraints. The motion-based keyframe extraction algorithm was chosen over random frame selection to maximize information capture, despite increased preprocessing time. The cropping strategy was a compromise between computational cost and feature preservation, as full-frame face detection was infeasible. For generation, the chosen hyperparameters balanced quality and efficiency, with alternatives like higher temperature or more sampling steps increasing diversity but risking incoherence. Due to time constraints, we did not integrate the trained audio, image/pose, and text tokenizers into the 4M model training pipeline.

IV. Experiments

In this project, we conducted a series of experiments aimed at implementing our proposed ideas while filtering out those that were impractical under the constraints we faced primarily limited GPU resources and strict time limitations. The data preprocessing and tokenization stages underwent multiple iterations. Our final strategy involved extracting 17 keyframes per video and applying the Cosmos Image Tokenizer, combined with a custom pipeline of cropping, resizing, and an enhancement for keypoints. As described in the Method section, this approach yielded promising reconstruction results and served as a cornerstone for our training configuration.

Given the infrastructure constraints, we needed to minimize the number of tokens per sample. Each RGB and keypoint sample resulted in $17 \times 256 = 4352$ tokens due to concatenation. However, to comply with hardware limits, we fixed the token limit per sample to 512 in the training configuration and set the batch size to 16. Initial training runs showed severe instability, with loss values diverging to approximately 400. To mitigate this, we tuned the model's configuration specifically by reducing the maximum learning rate to 10^{-4} and incorporating both **frame positional embeddings** and **spatial positional embeddings** to better capture temporal and spatial coherence across frames. After these modifications, the model achieved a much-improved loss of 5.2 overall, as shown in the apendix Figure 7.

For the **inference and generation** stages, our approach involved sequential hyperparameter tuning. After making the different experiments for our strategy, we observed that generated results revealed several weaknesses. Notably, the signs themselves were not clearly learned by the model this was apparent even in the best-performing samples, where distinct signs could not be recognized across frames. This limitation stemmed from our constraint of using only 512 tokens per sample, whereas number of tokens generated for RGB and keypoint modalities is up to eight times higher (4352). Attempts to increase this to 1024 tokens per sample by reducing the batch size did not yield significant improvements. Due to memory constraints, we were unable to explore configurations with 2048 tokens per sample. Also, due to a lack of time, we couldn't train with a reduced number of frames which is an alternative that could have potentially improved performance.

Nevertheless, the **RGB and keypoint generations** retained the general human shape and structure. By carefully adjusting the temperature and setting a high top-p value, we were able to discourage repetitive outputs and promote variability. While noisy, the generated frames sometimes showed a semblance of meaningful sign representations, suggesting the model was making some effort to capture temporal gesture patterns.

On the other hand, **caption generation** exhibited poor

structural coherence and linguistic quality. This issue was attributed not only to learning limitations but also to the overly generous token length allowed for caption outputs (512 tokens). This excessive length introduced randomness and noise, significantly degrading the quality of generated text. (Figure 6)

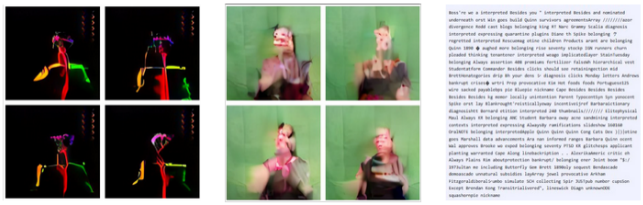

Figure 4. Some generations for Keypoints (Pose), Rgb and caption

For the audio tokenizer, we fine-tuned Meta's EnCodec model on the DailyTalk corpus using approximately **23,000 training samples**. We implemented a custom PyTorch **Dataset** that loads each **5-second** waveform (resampled from **16 kHz** to **24 kHz**), converts it to **mono**, and then pads or truncates it to a fixed length of **123,840 samples**. The corresponding **DataLoader** employs a simple **collate_fn** to stack and unsqueeze the audio tensors into the $(B, 1, L)$ format required by EnCodec. We configured Hugging Face's **TrainingArguments** (batch size = **4**, mixed precision **16-bit**, learning rate = $\mathbf{1 \times 10^{-4}}$)

For image tokenization, we adopted a VQGAN-based approach. We trained the model on a dataset containing **approximately 65,000 RGB training images**, along with **8,123 validation** and **8,124 test** images. Training was conducted using a batch size of **12**, with images resized to **256×256**. Key hyperparameters included a **codebook size of 1024** and a **learning rate of** 2×10^{-4}. The final tokenizer demonstrated strong performance, achieving a well-distributed codebook representation and low pixel-level reconstruction error.

Figure 5. Reconstruction Error of the AudioTokenizer : Specialized vs Default

The specialized tokenizer (blue) consistently lowers MSE versus the default (orange), hence fine-tuning EnCodec on our data yields significantly better reconstruction across utterances.

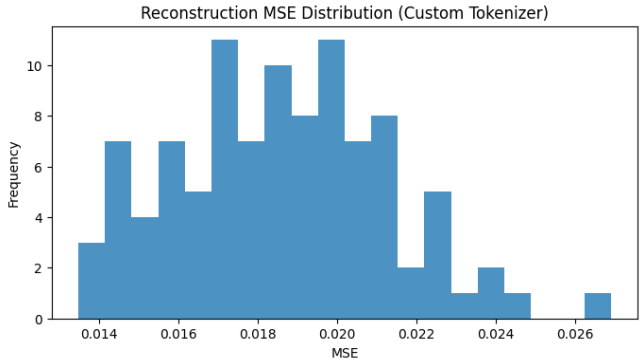

Figure 6. Reconstruction MSE Distribution of Custom Image Tokenizer

For the text tokenizer, we trained a custom BPE model on our corpus of sign-language transcripts. We split the data into **24,962 training**, **3,086 validation**, and **3,117 test** sentences. Using ByteLevel tokenization and a BPE trainer with a vocabulary size of **15k** and a minimum token frequency of **2**. Our custom tokenizer exhibits a clear tendency toward shorter, single-character tokens, which appear more frequently than in GPT-2's default tokenizer. This occurs primarily because our limited 15K-token vocabulary cannot cover every medium-length word or common substring. Practically, this means our tokenizer ensures zero out of vocabulary issues every unseen word is represented by known fragments but at the cost of increased fragmentation.

V. Conclusion and Limitations

We trained specialized tokenizers for images, audio, and text, and implemented a pipeline for converting raw sign language data into discrete tokens. Our lightweight nano4M model reduced total loss from 11 to 5 over 140,000 steps, achieving stable RGB and pose reconstructions. Preliminary multimodal generation and any-to-any modality translation capabilities were demonstrated.

Limited GPU availability via SCITAS significantly restricted development pace, and initial experiments with the Vidtok tokenizer were impractical and time-consuming. Nonetheless, our results show promising potential. With better resources and additional training time, this system could become an effective educational and assistive tool for improving sign language accessibility.

VI. Individual Contributions

All project stages were completed collaboratively, with constant mutual consultation and verification at each step to ensure shared understanding and consistent progress.

VII. Appendix

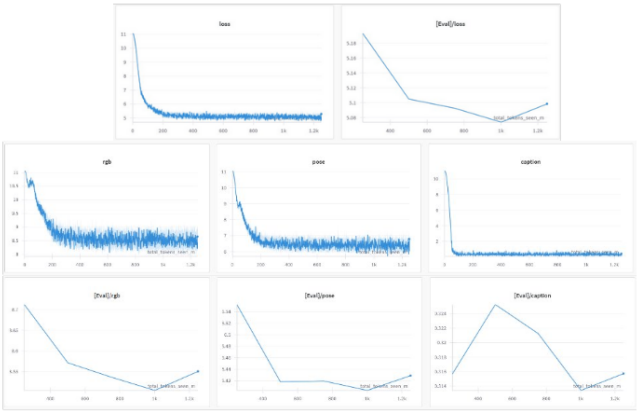

Figure 7. Training and evaluation losses : Up for global losses, middle for training modalities losses, down for evaluation modalities losses

References

[1] X. W. Kezhou Lin1 *et al.*, "Sign2text: End-to-end sign language translation," *Journal of Sign Processing*, 2020.

[2] N. C. Camgoz and et al., "Rwth-phoenix-weather multisigner slt dataset," *CVPR*, 2018.

[3] Y. Li, S. Zhang *et al.*, "Tokenpose: Learning keypoint tokens for human pose estimation," *arXiv preprint arXiv:2104.03516*, 2021.

[4] DeepMind, "Signgemma: Google deepmind's sign-to-speech model," 2025, https://www.gadgets360.com/ai/news/google-signgemma-ai-model-translate-sign-language-to-spoken-text-unveiled-8537400.

[5] R. Bachmann, D. Mizrahi *et al.*, "4m-21: An any-to-any vision model for tens of tasks and modalities," in *NeurIPS*, 2024.