

# Geo-Sign: Hyperbolic Contrastive Regularisation for Geometrically Aware Sign Language Translation - Appendix

---

<b>A</b>	<b>Introduction</b>	<b>2</b>
<b>B</b>	<b>Hyperbolic Geometry Preliminaries: A Brief Refresher</b>	<b>2</b>
<b>C</b>	<b>Methodology Details</b>	<b>3</b>
C.1	Pose Extraction and ST-GCN Architecture . . . . .	3
C.2	Hyperbolic Alignment Strategies . . . . .	4
<b>D</b>	<b>Mathematical Foundations</b>	<b>6</b>
D.1	Fréchet Mean in the Poincaré Ball . . . . .	6
D.2	Gradient of the Hyperbolic Distance . . . . .	7
<b>E</b>	<b>Learnable Model Parameters</b>	<b>7</b>
E.1	Discussion on Learnable Curvature . . . . .	7
E.2	Discussion on Loss Blending Factor $\alpha$ . . . . .	8
<b>F</b>	<b>Experimental Setup, Analysis, and Qualitative Results</b>	<b>8</b>
F.1	Computational Considerations and Profiler Analysis . . . . .	8
F.2	Further Technical Implementation Details . . . . .	12
F.3	Limitations and Future Work . . . . .	12
F.4	Qualitative Results . . . . .	13
<b>G</b>	<b>Code Listings</b>	<b>18</b>

---

## A Introduction

In this appendix, we provide comprehensive supplementary details to accompany our main paper. The goal is to offer an in-depth understanding of our methodology, experimental setup, and the underlying geometric principles, thereby ensuring clarity and facilitating the reproducibility of our work.

This document elaborates on:

- The specifics of pose feature extraction and the Spatio-Temporal Graph Convolutional Network (ST-GCN) architecture employed (Appendix C.1).
- Detailed explanations and implementations of our proposed hyperbolic alignment strategies, including the Pooled Method and the Token Method (Appendix C.2).
- Further mathematical derivations and discussions pertinent to hyperbolic operations, such as Fréchet mean computation and contrastive loss gradients (Appendix D).
- Elaboration on the learnable parameters within our model, particularly the manifold curvature  $c$  and the loss blending factor  $\alpha$  (Appendix E).
- A discussion of computational considerations, experimental setup, and qualitative results (Appendix F).
- Key code snippets for essential components of Geo-Sign are provided in Appendix G to aid in understanding and replication.

## B Hyperbolic Geometry Preliminaries: A Brief Refresher

To ensure this supplementary material is self-contained and accessible, this section briefly recaps key concepts from hyperbolic geometry, as introduced in Section 3.1 (“Hyperbolic Geometry Essentials”) of the main paper.

We operate within the  $d_{\text{hyp}}$ -dimensional Poincaré ball model, denoted  $\mathbb{B}_c^{d_{\text{hyp}}} = \{\mathbf{x} \in \mathbb{R}^{d_{\text{hyp}}} : \|\mathbf{x}\|_2 < 1/\sqrt{c}\}$ . This space is characterised by a constant negative curvature  $\kappa = -c$ , where  $c > 0$  is a learnable parameter representing the magnitude of the curvature.

The Poincaré ball model is chosen for its conformal nature, where angles are preserved locally, and its intuitive representation of hyperbolic space within a Euclidean unit ball (scaled by  $1/\sqrt{c}$ ). Key operations include:

- **Geodesic Distance**  $d_{\mathbb{B}_c}(\mathbf{u}, \mathbf{v})$ : This is the shortest path between two points  $\mathbf{u}, \mathbf{v}$  within the curved space of the Poincaré ball. It is formally defined in Eq. (1) of the main paper. Unlike Euclidean distance, it expands significantly as points approach the boundary of the ball.
- **Möbius Addition**  $\mathbf{u} \oplus_c \mathbf{v}$ : This operation is the hyperbolic analogue of vector addition in Euclidean space, defined in Eq. (2) of the main paper (consistent with formulations in, e.g., [5]). It is essential for defining translations and other transformations in hyperbolic space while respecting its geometry.
- **Exponential Map**  $\exp_{\mathbf{x}}^c(\mathbf{v})$ : This map takes a tangent vector  $\mathbf{v}$  residing in the tangent space  $\mathcal{T}_{\mathbf{x}}\mathbb{B}_c^{d_{\text{hyp}}}$  at a point  $\mathbf{x}$  on the manifold and maps it to another point on the manifold along a geodesic. The map from the origin,  $\exp_0^c(\cdot)$  (Eq. (3), main paper), is particularly important as it projects Euclidean feature vectors (which can be considered as residing in  $\mathcal{T}_0\mathbb{B}_c^{d_{\text{hyp}}}$ ) into the Poincaré ball.
- **Logarithmic Map**  $\log_{\mathbf{x}}^c(\mathbf{y})$ : This is the inverse of the exponential map. It takes two points  $\mathbf{x}, \mathbf{y}$  on the manifold and returns the tangent vector at  $\mathbf{x}$  that points along the geodesic towards  $\mathbf{y}$ .
- **Möbius Transformations**: These are isometries (distance-preserving transformations) of hyperbolic space. In our work, we use learnable Möbius transformations, such as Möbius matrix-vector products ( $\mathbf{M} \otimes_c \mathbf{v} = \exp_0^c(\mathbf{M} \log_0^c(\mathbf{v}))$ ) and Möbius bias additions, to implement affine-like transformations within our hyperbolic attention mechanism.

These tools allow us to define neural network operations directly within hyperbolic space. As with all hyperbolic operations in the paper, we utilise the geoopt library [9] in Pytorch.

## C Methodology Details

### C.1 Pose Extraction and ST-GCN Architecture Details

Our Geo-Sign framework utilizes skeletal pose data as input. This section details the extraction process and the architecture of the Spatio-Temporal Graph Convolutional Networks (ST-GCNs) used to encode this data.

#### C.1.1 Pose Data Source and Preprocessing

We use the 2D skeletal keypoints provided by the UniSign [10] framework, which were originally extracted using RTMPose-X [7] based on the COCO-WholeBody keypoint definition [8]. The keypoints are organised into four distinct anatomical groups for targeted processing:

- **Body:** Includes 9 joints (COCO indices 1, 4–11).
- **Left Hand:** Includes 21 joints (COCO indices 92–112).
- **Right Hand:** Includes 21 joints (COCO indices 113–133).
- **Face:** Includes 16 keypoints from the facial region (COCO indices 24, 26, 28, 30, 32, 34, 36, 38, 40, 54, 84–91).

For normalization, specific anchor joints are used for hand and face parts: joint 92 (left wrist) for the left hand, joint 113 (right wrist) for the right hand, and joint 54 (a central face point) for the face. The body part features are not anchor-normalised in this scheme to preserve global torso positioning.

#### C.1.2 ST-GCN Architecture

Each anatomical group is processed by a dedicated ST-GCN stream, following the methodology of Yan et al. [18]. The ST-GCN is adept at learning representations from skeletal data by explicitly modeling spatial joint relationships and temporal motion dynamics.

The core of the ST-GCN involves:

1. **Graph Definition:** The skeletal structure for each part is defined as a graph, where joints are nodes and natural bone connections are edges. The Graph class, detailed in Listing 1 (Appendix G), handles the construction of these graphs and their corresponding adjacency matrices.
2. **Initial Projection:** Input keypoint coordinates are first linearly projected to a higher-dimensional feature space using a linear layer (referred to as `proj_linear` in our codebase).
3. **ST-GCN Blocks:** A sequence of ST-GCN blocks processes these features. Each block (see `STGCN_block` in Listing 2, Appendix G) consists of:
  - A **Spatial Graph Convolution (SGC)** layer, which aggregates information from neighboring joints. The operation for a node (joint)  $v_i$  at layer  $(l)$  can be expressed generally as:

$$\mathbf{f}_{\text{out}}(v_i)^{(l)} = \sum_{k=1}^K \left( \sigma \left( \mathbf{A}_k \mathbf{X}^{(l)} \mathbf{W}_k^{(l)} \right) \right)_i, \quad (1)$$

where  $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times C_{in}}$  is the matrix of input features for  $N$  nodes with  $C_{in}$  channels,  $\mathbf{W}_k^{(l)} \in \mathbb{R}^{C_{in} \times C_{out}}$  are learnable weight matrices for the  $k$ -th kernel transforming node features to  $C_{out}$  channels.  $\mathbf{A}_k \in \mathbb{R}^{N \times N}$  is the adjacency matrix for the  $k$ -th spatial kernel, defining the neighborhood aggregation based on chosen strategies (we use the spatial configuration partitioning as in the original ST-GCN paper [18]).  $\sigma$  is an activation function (ReLU in our case), and  $(\cdot)_i$  denotes selection of the  $i$ -th row (features for node  $v_i$ ). The precise implementation involving tensor reshaping and einsum for efficient aggregation over multiple adjacency kernels is detailed in the `GCN_unit` code in Listing 2.

- A **Temporal Convolutional Network (TCN)** layer, which applies 1D convolutions across the time dimension to model motion patterns.
4. **Residual Connections:** To allow richer feature interaction, residual connections are introduced from the body stream’s ST-GCN output to the hand and face streams before their final temporal fusion layers. This allows global body posture context to inform the interpretation of fine-grained hand and face movements. Details are in Listing 3 (Appendix G). This design choice treats body features as fixed contextual input for the parts during each forward pass, isolating the body feature extractor from direct updates via part-specific losses.

The output of each part-specific ST-GCN stream is a feature map  $\mathbf{Z}_p \in \mathbb{R}^{T \times d'_{\text{gc\_out}}}$ , where  $T$  is the sequence length and  $d'_{\text{gc\_out}}$  is the GCN output feature dimension. For the hyperbolic regularization branch, these  $\mathbf{Z}_p$  are temporally mean-pooled to produce static summary vectors  $\bar{\mathbf{f}}_p \in \mathbb{R}^{d'_{\text{gc\_out}}}$ , which encapsulate the overall kinematics of part  $p$  for subsequent hyperbolic projection.

## C.2 Hyperbolic Alignment Strategies: Detailed Implementation

This section provides a more detailed explanation of the two hyperbolic alignment strategies introduced in Section 3.3 of the main paper. These strategies are designed to regularize the mT5 model by aligning pose and text representations within the Poincaré ball.

### C.2.1 Pooled Method (Global Semantic Alignment)

This strategy aims to align the holistic semantic content of the sign language video (represented by pose features) with the corresponding text translation.

**1. Part-Specific Hyperbolic Embeddings:** The temporally mean-pooled Euclidean feature vectors  $\bar{\mathbf{f}}_p$  for each anatomical part  $p$  (body, hands, face) are projected into the Poincaré ball  $\mathbb{B}_c^{d_{\text{hyp}}}$ . This projection, yielding hyperbolic embeddings  $\mathbf{h}_p$ , is achieved using the `HyperbolicProjection` layer (Listing 4 in Appendix G), as defined in Eq. (4) of the main paper:

$$\mathbf{h}_p = \exp_0^c(s_p \mathbf{W}^p \bar{\mathbf{f}}_p). \quad (2)$$

Here,  $\mathbf{W}^p$  represents a linear layer for part  $p$ , and  $s_p$  is a learnable scalar that adaptively scales the tangent space representation before the exponential map  $\exp_0^c(\cdot)$  projects it onto the manifold.

**2. Weighted Fréchet Mean for Global Pose Representation:** The set of part-specific hyperbolic embeddings  $\{\mathbf{h}_p\}$  is aggregated into a single global pose representation  $\boldsymbol{\mu}_{\text{pose}} \in \mathbb{B}_c^{d_{\text{hyp}}}$ . This is achieved by computing their weighted Fréchet mean, which is the hyperbolic analogue of a weighted average. The Fréchet mean is defined as the point that minimizes the sum of squared weighted geodesic distances to all input points:

$$\boldsymbol{\mu}_{\text{pose}} = \underset{\boldsymbol{\mu} \in \mathbb{B}_c^{d_{\text{hyp}}}}{\operatorname{argmin}} \sum_{p=1}^P w_p d_{\mathbb{B}_c}^2(\boldsymbol{\mu}, \mathbf{h}_p). \quad (3)$$

The weights  $w_p$  are designed to give more importance to parts whose embeddings are further from the origin of the Poincaré ball (i.e., parts with more "hyperbolic energy" or distinctness), normalised via softmax:

$$w_p = \frac{\exp(d_{\mathbb{B}_c}(\mathbf{0}, \mathbf{h}_p)/\lambda_w)}{\sum_{j=1}^P \exp(d_{\mathbb{B}_c}(\mathbf{0}, \mathbf{h}_j)/\lambda_w)}. \quad (4)$$

Here,  $\lambda_w$  is a temperature parameter for the softmax (e.g., fixed to 1.0 in our experiments) controlling the sharpness of the weight distribution. The computation is performed iteratively as detailed in Algorithm 1 of the main paper and Listing 5 (Appendix G).

**3. Global Text Representation:** Similarly, a global hyperbolic text embedding  $\mathbf{h}_{\text{text}} \in \mathbb{B}_c^{d_{\text{hyp}}}$  is derived from the mT5 model’s output. Euclidean token embeddings from the final layer of the mT5 decoder are first mean-pooled (respecting padding masks) to obtain a single sentence-level vector  $\bar{\mathbf{e}}_{\text{text}}$ . This vector is then projected into  $\mathbb{B}_c^{d_{\text{hyp}}}$  using a dedicated hyperbolic projection layer (structurally identical to Eq. (2)):

$$\mathbf{h}_{\text{text}} = \exp_0^c(s_{\text{text}} \mathbf{W}^{\text{text}} \bar{\mathbf{e}}_{\text{text}}). \quad (5)$$

The implementation details are shown in Listing 6 (Appendix G).

**4. Contrastive Alignment:** Finally, the geometric contrastive loss (Eq. (5) in the main paper) is applied between batches of these global pose embeddings  $\{\mu_{\text{pose},i}\}$  and global text embeddings  $\{h_{\text{text},i}\}$ . This encourages semantically similar pose-text pairs to be closer in hyperbolic space.

### C.2.2 Token Method (Fine-Grained Part-Text Alignment)

This strategy facilitates a more detailed alignment by relating individual pose part embeddings  $\{h_p\}$  with contextually relevant text segment embeddings  $\{c_p\}$ .

**1. Hyperbolic Pose Part Embeddings  $\{h_p\}$ :** These are obtained exactly as in the Pooled Method, using Eq. (2). Each  $h_p$  represents a specific anatomical part’s overall kinematic signature.

**2. Hyperbolic Text Token Embeddings:** Instead of a global text embedding, each Euclidean text token embedding  $e_{\text{token},j}$  (from the mT5 decoder’s final layer) is individually projected into the Poincaré ball  $\mathbb{B}_c^{d_{\text{hyp}}}$ :

$$h_{\text{token},j} = \exp_0^c(s_{\text{text}} \mathbf{W}^{\text{text}} e_{\text{token},j}). \quad (6)$$

This results in a sequence of hyperbolic token embeddings  $\{h_{\text{token},j}\}_{j=1}^{L_t}$ , where  $L_t$  is the text sequence length.

**3. Hyperbolic Attention Mechanism:** For each hyperbolic pose part embedding  $h_p$  (acting as a query), a contextual text embedding  $c_p$  is generated. This is achieved using a hyperbolic attention mechanism (see Listing 7 in Appendix G) that operates as follows:

- **Key Transformation:** The hyperbolic text token embeddings  $\{h_{\text{token},j}\}$  serve as keys. The embeddings are first transformed using learnable Möbius transformations to enhance their representational capacity:

$$\mathbf{k}_j = (\mathbf{M}_{\text{key}} \otimes_c h_{\text{token},j}) \oplus_c \mathbf{b}_{\text{key}},$$

where  $\mathbf{M}_{\text{key}}$  is a learnable Möbius matrix and  $\mathbf{b}_{\text{key}}$  is a learnable Möbius bias vector.

- **Attention Scores:** Attention scores are computed based on the negative geodesic distance between each pose query  $h_p$  and each transformed text key  $\mathbf{k}_j$ :

$$\text{score}_{pj} = -d_{\mathbb{B}_c}(h_p, \mathbf{k}_j).$$

- **Attention Weights:** These scores are normalised using a softmax function (after applying padding masks) to obtain attention weights  $\alpha_{pj}$ :

$$\alpha_{pj} = \text{softmax}\left(\frac{\text{score}_{pj}}{\tau_{\text{attn}}}\right),$$

where  $\tau_{\text{attn}}$  is a learnable temperature parameter for the attention mechanism, distinct from the temperature in the contrastive loss.

- **Contextual Text Embedding  $c_p$ :** The contextual text embedding  $c_p$  corresponding to pose part  $h_p$  is then computed as the hyperbolic weighted midpoint of the original hyperbolic text token embeddings  $\{h_{\text{token},j}\}$ , using the attention weights  $\{\alpha_{pj}\}$ .

**4. Contrastive Alignment:** The geometric contrastive loss (Eq. (5), main paper) is then applied for each pair  $(h_{p,i}, c_{p,i})$  across the batch. The total regularization loss for this strategy is the average of these individual contrastive losses over all parts  $P$ .

### C.2.3 Intuition Behind the Token Method

While the Pooled Method aligns the overall semantics of a sign sequence with its translation, it may not capture how specific signing elements (e.g., a handshake, movement, or facial expression) correspond to particular words or phrases. The Token Method aims to establish this more fine-grained understanding.

The core intuition is as follows:

1. **Compositional Language Understanding:** Sign languages, like spoken/written languages, are compositional. Different articulators (hands, body, face) convey distinct lexical or

grammatical information. The Token Method attempts to map these compositional units from pose to corresponding textual tokens (words/sub-words).

2. **Targeted Part-to-Segment Alignment:** Instead of a single global comparison, this method learns to connect individual pose part representations (e.g., features for the dominant hand, to the most relevant segments of the textual translation.
3. **Pose Parts as Queries, Text Tokens as Sources:** Each hyperbolic pose part embedding  $\mathbf{h}_p$  acts as a "query", effectively asking: "Which text tokens are most semantically relevant to this pose feature?" The sequence of hyperbolic text token embeddings  $\{\mathbf{h}_{\text{token},j}\}$  serves as the "information source" for these queries.
4. **Hyperbolic Attention for Geometric Relevance:**
  - Relevance between a pose part query  $\mathbf{h}_p$  and a (transformed) text token key  $\mathbf{k}_j$  is measured by their geodesic distance  $d_{\mathbb{B}_c}(\mathbf{h}_p, \mathbf{k}_j)$  in the learned hyperbolic space. A smaller distance implies higher relevance. Using hyperbolic geometry allows these comparisons to potentially leverage latent hierarchical relationships between concepts.
  - Learnable Möbius transformations on text tokens (to get keys  $\mathbf{k}_j$ ) enable the model to learn distinct tokens relevant to different pose parts (e.g., a verb token might be transformed to be closer to a body movement embedding).
  - The attention weights  $\alpha_{pj}$  then quantify the contribution of each text token  $j$  to the meaning conveyed by pose part  $p$ .
5. **Learning Textual Context for Each Pose Part:** The contextual text embedding  $\mathbf{c}_p$  is a hyperbolic weighted midpoint of all text token embeddings, using the attention weights  $\alpha_{pj}$ . Thus,  $\mathbf{c}_p$  is a summary of the sentence, but specifically customised by the interaction of pose part  $p$ .
6. **Refined Contrastive Learning:** The model is regularised to make each pose part embedding  $\mathbf{h}_p$  close to its corresponding contextual text view  $\mathbf{c}_p$  in hyperbolic space, while pushing it away from non-corresponding pairs.
7. **Overall Benefit:** This detailed, part-specific alignment encourages the mT5 model to learn more precise mappings between kinematic features of different articulators and semantic units within the text. For example, it can help distinguish visually similar signs based on subtle hand details (encoded in  $\mathbf{h}_{\text{hand}}$ ) that correlate with specific words, leading to more accurate and nuanced translations.

## D Mathematical Foundations

This section recalls two geometric components that Geo-Sign relies on:

- the *Weighted Fréchet Mean* inside the Poincaré ball (used in Algorithm 1 of the paper);
- the Euclidean gradient of the hyperbolic distance that appears in the contrastive loss.

### D.1 Fréchet Mean in the Poincaré Ball

Given points  $x_1, \dots, x_N$  in a metric space  $(\mathcal{M}, d)$  with normalised weights  $w_i > 0$ ,  $\sum_i w_i = 1$ , the **Fréchet mean** minimises

$$\mathcal{F}(\mu) = \sum_{i=1}^N w_i d^2(\mu, x_i), \quad \mu^* = \arg \min_{\mu \in \mathcal{M}} \mathcal{F}(\mu).$$

**Why not simply average the embeddings in Euclidean space?** Two issues appear inside the curved Poincaré ball:

- (a) **Manifold constraint.** A Euclidean average of interior points can fall *outside* the ball, i.e. outside valid hyperbolic space, forcing an ad-hoc projection that distorts geometry.
- (b) **Metric distortion.** Euclidean distance underestimates separation near the boundary because the hyperbolic metric stretches space there. A straight average therefore over-emphasises central points and washes out fine structure carried by peripheral ones.

The intrinsic Fréchet mean lives on the manifold and uses the true hyperbolic distance, so it respects curvature.

**Why distance-based weights?** Each pose part (body, face, left hand, right hand) yields a hyperbolic embedding  $h_p$ . We set  $w_p \propto \exp(d_{\mathbb{B}_c}(0, h_p)/\lambda_w)$  so parts farther from the origin, in regions of higher curvature and greater discriminative power, receive more influence. Without this weighting the mean would drift toward the centre, diluting information contributed by the hands and face.

**Iterative update.** On any Riemannian manifold the mean is found by Riemannian gradient descent; the update at iteration  $k$  is

$$\mu^{(k+1)} = \exp_{\mu^{(k)}} \left( \eta_k \sum_{i=1}^N w_i \log_{\mu^{(k)}}(x_i) \right), \quad (7)$$

with step size  $\eta_k > 0$ .

**Proposition D.1** (Convergence in  $\mathbb{B}_c^d$ ). *The Poincaré ball  $\mathbb{B}_c^d$  is a Hadamard manifold, hence  $\mathcal{F}$  is strictly convex and has a unique minimiser  $\mu^*$ . Let  $L$  be the Lipschitz constant of  $\nabla \mathcal{F}$  on the geodesic convex hull of  $\{x_i\}$ . If  $0 < \eta_k \leq 2/L$  for all  $k$ , the iterates (7) converge to  $\mu^*$ . In practice we observe  $L \leq 2$ , so the simple choice  $\eta_k = 1$  is usually sufficient and used in our approach.*

## D.2 Gradient of the Hyperbolic Distance

For  $u, v \in \mathbb{B}_c^d$  let  $w = (-u) \oplus_c v$  (the Möbius difference, i.e. the "vector" from  $u$  to  $v$  transported to the origin). The Poincaré distance is

$$d_{\mathbb{B}_c}(u, v) = \frac{2}{\sqrt{c}} \operatorname{artanh}(\sqrt{c} \|w\|_2).$$

Differentiating [5, 12] gives the Euclidean gradient required for autograd:

$$\nabla_u d_{\mathbb{B}_c}(u, v) = -\frac{2}{\lambda_u^c \lambda_v^c} \frac{w}{\|w\|_2} \frac{1}{1 - c\|w\|_2^2} \quad (8)$$

with conformal factor  $\lambda_x^c = \frac{2}{1 - c\|x\|_2^2}$ . The same formula (with sign reversed) holds for  $\nabla_v$ .

The update rule (7) and the gradient (8) provide all the geometric tools needed by Geo-Sign’s hyperbolic contrastive regulariser.

## E Learnable Model Parameters: $c$ and $\alpha$

Our Geo-Sign model incorporates several learnable parameters beyond standard network weights. This section details two key ones: the manifold curvature  $c$  and the loss blending factor  $\alpha$ .

### E.1 Discussion on Learnable Curvature

The curvature of the Poincaré ball,  $\kappa = -c$  (where  $c > 0$ ), is a crucial hyperparameter that dictates the “shape” of the hyperbolic space. Instead of fixing  $c$  heuristically, we make it a learnable parameter of our model (see Listing 9 in Appendix G).

**Optimization Strategy:** The curvature magnitude  $c$  is initialised (e.g., via `args.init_c` as mentioned in the main paper’s experiments) and then updated via standard gradient descent as part of the end-to-end training process. The `geoopt` library facilitates this by defining  $c$  as an `nn.Parameter` within its `PoincareBall` manifold object when `learnable=True`.

The main paper’s ablation studies (Table 2a) show that initializing  $c$  in the range of 1.0 – 2.0 (e.g., optimal BLEU-4 at  $c = 1.5$ ) yields strong performance. Figure 1 illustrates how  $c$  adapts during training from different initializations.

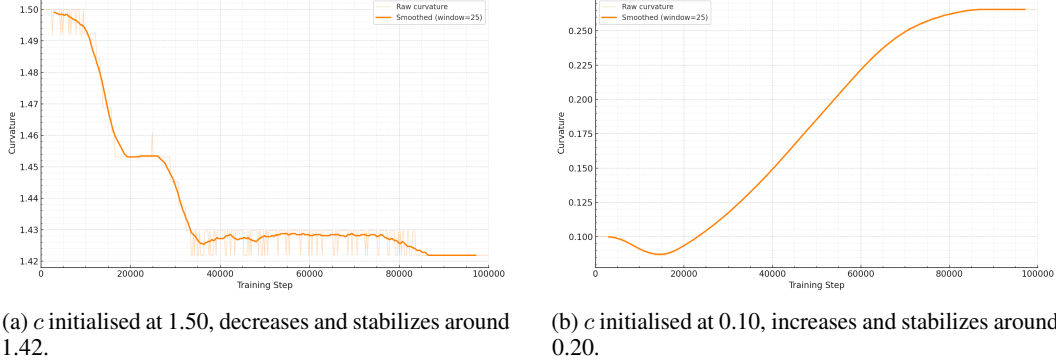


Figure 1: Evolution of the learnable manifold curvature  $c$  during training for different initializations. (a) When initialised at  $c = 1.50$ , the curvature magnitude slightly decreases, suggesting an optimal value around 1.42 for this setup. (b) When initialised at a low  $c = 0.10$ , the curvature increases, indicating the model benefits from more “hyperbolic space” initially. It stabilizes around  $c = 0.20$ , potentially influenced by the dynamic  $\alpha$  schedule that reduces regularization emphasis over time.

## E.2 Discussion on Loss Blending Factor $\alpha$

The total training loss  $\mathcal{L}_{\text{total}}$  is a weighted combination of the primary cross-entropy translation loss  $\mathcal{L}_{\text{CE}}$  and our hyperbolic contrastive regularization term  $\mathcal{L}_{\text{hyp\_reg}}$ :

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{CE}} + (1 - \alpha) \cdot \mathcal{L}_{\text{hyp\_reg}}.$$

The blending factor  $\alpha$  is not fixed but is dynamically adjusted during training. This dynamic scheduling allows the model to potentially benefit from different loss emphases at different training stages. The calculation of  $\alpha$  at each training step (see Listing 10 in Appendix G) is:

$$\alpha_{\text{final}} = \text{clamp}((\alpha_{\text{init}} + 0.1 \cdot \text{progress}) + \sigma(\text{logit}_{\alpha}) \cdot 0.2, 0.1, 1.0), \quad (9)$$

where:

- $\alpha_{\text{init}}$  is the initial value for the blending factor, specified as a hyperparameter (e.g., `args.alpha = 0.7` from the main paper’s ablations, Table 2b, which was found to be optimal).
- `progress` is the current training progress, calculated as  $\frac{\text{current\_training\_step}}{\text{total\_training\_steps}}$ , ranging from 0 to 1. This component introduces a linear ramp, potentially increasing  $\alpha$ ’s baseline by up to 0.1 over the course of training.
- `logit $\alpha$`  is an `nn.Parameter` (a learnable scalar, referred to as `self.loss_alpha_logit` in the code).  $\sigma(\cdot)$  is the sigmoid function, so  $\sigma(\text{logit}_{\alpha})$  maps this learnable scalar to the range  $(0, 1)$ . This term provides a learnable adjustment to  $\alpha$  in the range of  $[0, 0.2]$ .
- `clamp( $\cdot$ , 0.1, 1.0)` ensures that the final  $\alpha_{\text{final}}$  remains within the bounds  $[0.1, 1.0]$ .

This dynamic  $\alpha$  allows for an initial phase where the hyperbolic regularization might have more relative influence (if  $\alpha_{\text{init}}$  is smaller), gradually shifting emphasis or allowing the model to fine-tune the balance via the learnable component. The ablation study in the main paper (Table 2b) indicates that an initial  $\alpha_{\text{init}} = 0.7$  (i.e., 30% weight to  $\mathcal{L}_{\text{hyp\_reg}}$  initially) provides the best results, highlighting the complementary role of the hyperbolic regularization.

## F Experimental Setup, Analysis, and Qualitative Results

### F.1 Computational Profile

This section discusses the computational profile of Geo-Sign, comparing it to a baseline Uni-Sign (Pose) model without hyperbolic regularization. The analysis is based on DeepSpeed profiler outputs for models run with a batch size of 8 on the CSL-Daily dataset for the Sign Language Translation (SLT) task.



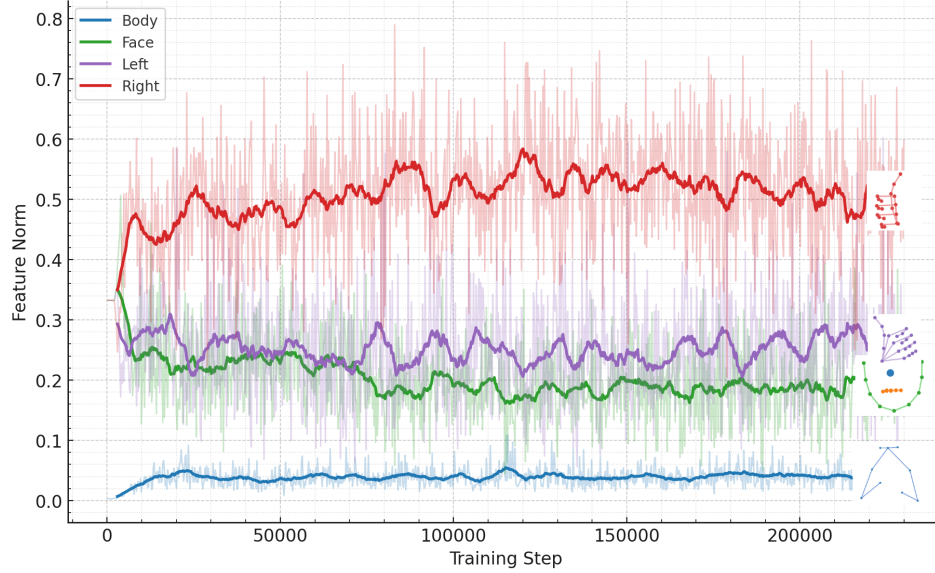


Figure 2: Plot of the geodesic distances from the origin (0) of the Poincaré disk to the hyperbolic pose embeddings ( $h_p$ ) during training, averaged per part type. This shows how features for different parts utilize the hyperbolic space. For instance, right hand features (often conveying detailed lexical information) tend to move further from the origin, leveraging more of the hyperbolic curvature for discriminability. Body and face features, which might represent broader semantics or prosody, may remain closer to the Euclidean-like central region.

**Experimental Context:** Key experimental conditions for fine-tuning include:

- **Hardware:** 4 NVIDIA RTX 3090 GPUs.
- **Training Time:** Approximately 10 hours for 40 epochs of fine-tuning on CSL-Daily.
- **Precision:** Mixed-precision training (bf16) is used for standard PyTorch layers, while float32 is maintained for Geoopt hyperbolic operations to ensure numerical stability.
- **Batching Strategy:** With an effective batch size of 8 per GPU, the model occupies  $\approx 20$ GB of memory. During training, we increase the total batch size to 32 and accumulate gradients over 8 steps, achieving a hypothetical batch size of 256. For the following profiler analysis, we report results for a single GPU with a batch size of 8 to provide a clear per-device profile.

### F.1.1 Profiler Summary and Comparative Analysis

Table 1 summarizes key metrics from the profiler. Parameter counts are consistent with the main paper’s Table 1, while MACs (Multiply-Accumulate operations) and Latency are derived from DeepSpeed profiler outputs for a batch size of 8. Table 2 provides a comparison of model parameters against other gloss-free methods.

Table 1: Computational profile comparison at Batch Size 8: Baseline Uni-Sign (Pose) vs. Geo-Sign variants. Parameter counts from main paper’s Table 1. MACs and Latency from DeepSpeed profiler outputs. “Hyperbolic Proj. Layer MACs” reflects profiled contributions from the learnable linear transformations within these layers.

Model Variant (Batch Size 8)	Total Params (M)	Added Params (M)	Total Fwd MACs (GMACs)	Hyperbolic Proj. Layer MACs (MMACs)	Fwd Latency (ms)	Latency Increase (%)
Baseline Uni-Sign (Pose)	587.75	-	116.59	-	415.73	-
Geo-Sign (Hyperbolic Pooled)	588.21	0.46	116.60	3.67	1630.00	292.10
Geo-Sign (Hyperbolic Token)	589.10	1.35	116.60	$\approx 9.96$	2550.00	513.40

**Parameter Overhead:** The increase in parameters due to the hyperbolic components is marginal compared to the overall model size, which is dominated by the mT5 language model ( $\approx 582.4$ M parameters).

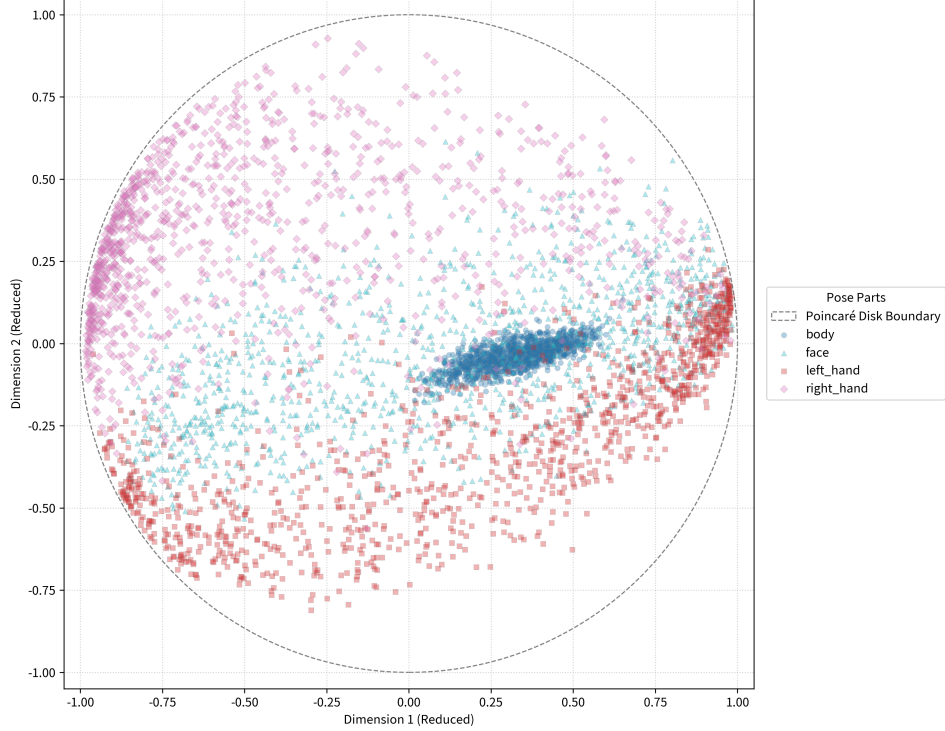


Figure 3: PCA projection of 1000 hyperbolic pose part embeddings (log-mapped to the tangent space at origin, then PCA-reduced to 2D) visualised within the Poincaré disk. Body features (blue) are tightly clustered near the origin, suggesting their discriminability is well-handled in a more Euclidean-like region. Hand features (left: red square, right: pink diamond) and face features (light blue triangle) are more dispersed, with hand features often pushed towards the periphery. This indicates these parts benefit from the increased representational capacity near the boundary of the Poincaré disk, where hyperbolic geometry provides more “space” to distinguish subtle variations crucial for sign language semantics.

- Baseline Uni-Sign (Pose):  $\approx 587.75\text{M}$  parameters.
- **Geo-Sign (Pooled)**: Adds  $\approx 0.46\text{M}$  parameters, primarily from the five hyperbolic projection layers (one for each of the four pose parts and one for the pooled text embedding).
- **Geo-Sign (Token)**: Adds  $\approx 1.35\text{M}$  parameters. This includes the  $\approx 0.46\text{M}$  for projection layers plus an additional  $\approx 0.89\text{M}$  for the learnable parameters within the hyperbolic attention mechanism (Möbius matrices and biases).

In both Geo-Sign variants, the parameter overhead from hyperbolic components is less than 0.25% of the total model size. As shown in Table 2, our Geo-Sign models achieve competitive or superior performance to recent RGB-based methods while maintaining a significantly smaller total parameter count. This highlights the efficiency of enhancing skeletal representations with geometric priors, challenging the trend that relies solely on scaling up visual encoders and language model decoders for performance gains in SLT.

**MACs Analysis:** The DeepSpeed profiler indicates that the total forward MACs are very similar across all configurations at this batch size:

- Baseline Uni-Sign (Pose):  $\approx 116.59$  GMACs.
- Geo-Sign (Hyperbolic Pooled):  $\approx 116.60$  GMACs. The profiler attributes  $\approx 3.67$  MMACs to the linear transformations within its HyperbolicProjection layers.
- Geo-Sign (Hyperbolic Token):  $\approx 116.60$  GMACs. Its HyperbolicProjection layers account for  $\approx 9.96$  MMACs from their linear components.

Table 2: Sign Language Translation performance (Test Set: BLEU-4, ROUGE-L) and model parameters on CSL-Daily. Scores are percentages (%). Higher is better. ‘Pose’ and ‘RGB’ indicate input modalities. VE/LM/Total Params are in Millions (M). Approx. values indicated by  $\approx$ . Data from CSL-Daily (Train: 18,401 sentences / 20.62 hours).

Method	VE Name	VE	LM Name	LM	Total	Modality		Test Set	
		Params (M)		Params (M)	Params (M)	Pose	RGB	B-4	R-L
Gloss-Free Methods (Prior Art)									
MSLU [21]	EffNet	5.3	mT5-Base	582.4	587.7	✓	–	11.42	33.80
SLRT [1] (G-Free)	EffNet	5.3	Transformer	≈30	≈35.3	–	✓	3.03	19.67
GASLT [19]	I3D	13	Transformer	≈30	≈43.0	–	✓	4.07	20.35
GFSLT-VLP [20]	ResNet18	11.7	mBart	680	691.7	–	✓	11.00	36.44
FLa-LLM [3]	ResNet18	11.7	mBart	680	691.7	–	✓	14.20	37.25
Sign2GPT [16]	DinoV2	21.0	XGLM	1732.9	1753.9	–	✓	15.40	42.36
SignLLM [6]	ResNet18	11.7	LLaMA-7B	6738.4	6750.1	–	✓	15.75	39.91
C <sup>2</sup> RL [2]	ResNet18	11.7	mBart	680	691.7	–	✓	21.61	48.21
Our Models and Baselines									
Uni-Sign [10] (Pose)	GCN	5.3	mT5-Base	582.4	587.7	✓	–	25.61	54.92
Uni-Sign [10] (Pose+RGB)	EffNet+GCN	9.7	mT5-Base	582.4	592.1	✓	✓	26.36	56.51
Geo-Sign (Hyperbolic Pooled)	GCN+Geo	5.8	mT5-Base	582.4	588.21	✓	–	27.17	57.75
Geo-Sign (Hyperbolic Token)	GCN+Geo+Attn	6.7	mT5-Base	582.4	589.1	✓	–	27.42	57.95

The MACs from the learnable linear transformations within the hyperbolic projection layers constitute a very small fraction ( $< 0.01\%$ ) of the total model MACs. The bulk of MACs originates from the mT5 model (profiled at  $\approx 66.29$  GMACs) and the ST-GCN modules (profiled at  $\approx 49.93$  GMACs). We should note, however, that standard profilers (like DeepSpeed’s MAC counter) primarily quantify MACs from common operations like convolutions and linear layers. The computational cost of specialised geometric functions within `geoopt` (e.g., `manifold.dist`, `expmap0`, `logmap0`, Möbius arithmetic) is not explicitly broken out as distinct hyperbolic operation MACs. These functions often involve sequences of elementary operations that are not all MAC-based (e.g., square roots, divisions, trigonometric functions like `artanh` or `tanh`). Thus, their computational load may be underestimated by MAC counters and is often better reflected in measured latency.

**Latency Analysis:** Latency figures clearly reveal the primary computational overhead introduced by the hyperbolic components during training:

- Baseline Uni-Sign (Pose):  $\approx 416$  ms forward latency per batch.
- Geo-Sign (Hyperbolic Pooled):  $\approx 1630$  ms (1.63 s), an increase of  $\approx 1214$  ms or  $\approx 292\%$  over the baseline (approx.  $3.9\times$  slowdown).
- Geo-Sign (Hyperbolic Token):  $\approx 2550$  ms (2.55 s), an increase of  $\approx 2134$  ms or  $\approx 513\%$  over the baseline (approx.  $6.1\times$  slowdown).

The substantial increase in training latency, despite modest increases in parameters and profiled MACs from learnable layers, underscores that the geometric operations themselves are the main performance consideration during the training phase. These operations (e.g., geodesic distance, exponential/logarithmic maps, Möbius transformations) are inherently more complex than their Euclidean counterparts. The Token method is notably slower than the Pooled method during training due to its per-token hyperbolic attention.

Importantly, a key advantage of our regularization approach is that these geometric operations and the hyperbolic branch are **not utilised at inference time**. Consequently, Geo-Sign models incur no additional latency increase over the baseline Uni-Sign (Pose) model during inference, preserving efficiency for deployment.

### F.1.2 Discussion on Data Efficiency

While not directly evaluated, it is hypothesised that skeletal data’s abstraction from visual noise (lighting, background, clothing) can enhance robustness and generalization [17], especially when training data is limited. Hyperbolic geometry further imposes a structural prior on the representation space. This inductive bias could potentially improve data efficiency by guiding the learning process, particularly in scenarios with sparse data, although specific experiments to quantify this effect were not part of the current study. One trade-off of this approach is that we cannot directly leverage

large pre-trained visual encoders as in the case of other RGB approaches, and so pre-training on a sign-specific dataset like CSL-News (1,985 hours, used by Uni-Sign) is essential. However, this pre-training data size is comparable to that used by other SLT methods which use datasets such as How2Sign [4] (2000 hours) or YouTube-ASL [14, 15] (6000 hours). We anticipate that our method would continue to scale well with larger pre-training datasets in other sign languages, though resource constraints prevented evaluation of this aspect.

## F.2 Further Technical Implementation Details

This section provides additional details that are pertinent for a full understanding and potential reimplementaion of Geo-Sign.

- **Core Libraries:** Our implementation relies on PyTorch [13] as the primary deep learning framework. For Transformer models, we utilize the HuggingFace Transformers library. All hyperbolic geometry operations and Riemannian optimization are handled by the Geoopt library [9]. For distributed training and profiling, DeepSpeed is employed.
- **Hyperparameter Tuning Strategy:** Key hyperparameters specific to the hyperbolic components, such as the initial curvature  $c$ , the initial loss blending factor  $\alpha_{\text{init}}$  (referred to as `args.alpha` in code/main paper), and the hyperbolic embedding dimension  $d_{\text{hyp}}$ , were tuned using a grid search strategy on the CSL-Daily development set. Full hyperparameters are outlined in Table 3.
- **Numerical Stability Measures:**
  - Operations within `geoopt` are performed using `float32` precision to maintain numerical stability, while the rest of the model uses mixed precision.
  - Small epsilon values (e.g.,  $10^{-5}$ ) are added in denominators and inside logarithms/arc-tanh functions where appropriate to prevent division by zeros.
  - **Tangent Vector Clipping:** Before applying an exponential map  $\exp_{\mathbf{x}}^c(\mathbf{v})$  from a point  $\mathbf{x}$  with a tangent vector  $\mathbf{v}$ , especially  $\exp_0^c(\mathbf{v})$ , it’s crucial to ensure the resulting point remains strictly within the Poincaré ball and that the norm of  $\mathbf{v}$  doesn’t cause numerical issues in  $\tanh(\cdot)$ . We apply a clipping strategy as mentioned in Section 3.4 of the main paper:

$$\mathbf{v}_{\text{clipped}} \leftarrow \frac{\mathbf{v}}{\max(1, \sqrt{c}\|\mathbf{v}\|_2 + \epsilon_{\text{clip}})},$$

for a small  $\epsilon_{\text{clip}} > 0$  (e.g.,  $10^{-5}$ ). This ensures that the argument to  $\tanh$  in  $\exp_0^c$  does not become excessively large and that mapped points do not reach or exceed the boundary of the Poincaré ball. The `project=True` flag in `geoopt`’s `expmap` functions also helps enforce this by projecting points back onto the ball if they numerically fall outside.

- **Gradient Clipping:** Standard norm-based gradient clipping is applied to all model parameters during training to stabilize the optimization process.

In Table 3 we provide the full hyper-parameters for the best performing model. The full code will be released following the review process.

## F.3 Limitations and Future Work

While offering representational benefits, hyperbolic operations can add computational overhead compared to purely Euclidean ones though this is generally offset by avoiding raw video processing. The optimal choice of hyperbolic model parameters (e.g., curvature strategy) warrants further study. Generalizability to a wider range of sign languages also needs investigation. Promising directions include exploring other hyperbolic models (e.g., Lorentz), developing more sophisticated dynamic curvature adaptation, integrating Geo-Sign’s hyperbolic skeletal features into multi-modal frameworks, and applying these geometric principles to other sign language processing tasks like recognition or generation. Further research into the interpretability of learned hyperbolic embeddings could also yield deeper insights into how sign language structure is captured.

Table 3: Hyperparameter summary for Geo-Sign experiments. Values are for the best reported model configuration.

Category	Hyperparameter	Value	Description
<b>General Training Configuration</b>			
	Random Seed	42	Seed for reproducibility
	Training Epochs	40	Number of fine-tuning epochs on CSL-Daily
	Batch Size (per GPU)	8	Micro-batch size per GPU
	Gradient Accumulation Steps	8	Effective batch size becomes $8 \times \text{accum\_steps} \times \text{num\_gpus}$
	Training Precision (dtype)	bf16	Mixed precision training data type
<b>Data Handling</b>			
	Max Pose Sequence Length	256	Maximum number of frames for pose sequences
	Max Target Text Length (max_tgt_len)	100	Max new tokens for generation during evaluation
<b>Optimizer (Euclidean: ST-GCN, mT5, Linear Layers)</b>			
	Optimizer Type (opt)	AdamW	[11]
	Learning Rate (lr)	$3 \times 10^{-5}$	For Euclidean parameters (AdamW)
	AdamW $\beta_1, \beta_2$ (opt-betas)	[0.9, 0.999]	Exponential decay rates for moment estimates
	AdamW $\epsilon$ (opt-eps)	$1 \times 10^{-8}$	Term for numerical stability
	Weight Decay (weight-decay)	0.01	L2 penalty for Euclidean parameters
	LR Scheduler (sched)	Cosine Annealing	
	Warmup Epochs (warmup-epochs)	5	Number of epochs for LR warm-up
	Minimum LR (min-lr)	$1 \times 10^{-6}$	Lower bound for LR in scheduler
	Gradient Clipping Norm	1.0	Max norm for gradients
<b>Optimizer (Hyperbolic: Manifold Parameters, Projections)</b>			
	Optimizer Type	RAdam	Riemannian Adam
	Learning Rate (hyp_lr)	$1 \times 10^{-3}$	For hyperbolic parameters (RAdam)
<b>Model Architecture</b>			
	ST-GCN Output Dimension (gcn_out_dim)	256	Output dimension of ST-GCN part streams
	mT5 Projection Dimension (hidden_dim)	768	Target dimension for projecting GCN features to match mT5
<b>Hyperbolic Regularization</b>			
	Hyperbolic Embedding Dimension ( $d_{\text{hyp}}$ , hyp_dim)	32	Dimension of embeddings in Poincaré ball
	Initial Curvature ( $c_{\text{init}}$ , init_c)	1.5	Initial value for learnable curvature $c$ (for best model)
	Loss Blend $\alpha_{\text{init}}$ (alpha)	0.70	Initial blending factor for $\mathcal{L}_{\text{CE}}$ vs $\mathcal{L}_{\text{hyp\_reg}}$ (for best model)
	Text Comparison Mode (hyp_text_cmp)	token	Strategy for aligning pose with text tokens (Token Method)
	Hyperbolic Contrastive Loss $\mathcal{L}_{\text{hyp\_reg}}$ :		
	Temperature ( $\tau$ )	Learnable	Temperature for scaling distances in contrastive loss
	Margin ( $m$ )	Learnable	Additive margin for negative pairs in contrastive loss
	Label Smoothing (label_smoothing_hyp)	0.2	Label smoothing for hyperbolic contrastive loss (InfoNCE)
<b>Loss Functions</b>			
	CE Loss Label Smoothing (label_smoothing)	0.2	Label smoothing for mT5 cross-entropy loss
<b>Distributed Training (DeepSpeed)</b>			
	ZeRO Optimization Stage (zero_stage)	2	DeepSpeed ZeRO Stage for memory efficiency
	Offload to CPU (offload)	False	Whether to offload optimizer/params to CPU

## F.4 Qualitative Results

**Additional Figures:** Figure 2 (similar to aspects shown in Figure 2 of the main paper, concerning learned embedding distributions) illustrates the dynamic utilization of the hyperbolic manifold by showing the average geodesic distance of different pose part embeddings from the origin during training. Notably, features corresponding to hand articulations, which often carry fine-grained lexical information, tend to migrate towards the periphery of the Poincaré disk. This suggests that the model leverages the increased representational capacity in high-curvature regions to distinguish subtle hand-based signs.

Furthermore, Figure 3 (again, related to Figure 2 of the main paper, specifically the UMAP projections) provides a PCA-reduced visualization of the learned hyperbolic pose part embeddings projected onto the 2D Poincaré disk for 1000 poses. This plot reveals a structured distribution where body features cluster near the origin (a more Euclidean-like region suitable for broader semantics), while hand and face features are more dispersed, with hand features populating regions further towards the boundary. This geometric organization, reflecting a learned kinematic hierarchy, likely contributes to the improved discriminability and, consequently, the enhanced translation quality demonstrated in the following examples. These visualizations support the hypothesis that the geometric biases induced by hyperbolic space aid in forming more effective representations for sign language translation.

**Translation Results:** In this section, we provide an overview of translation samples generated by Geo-Sign. All predictions are from our best-performing “Token” model. First, in Table 4, we show examples of prediction errors with analysis and a general measure of semantic similarity (introduced for readability, not a quantitative metric). English translations are automatically generated and then verified by a native Chinese speaker. We observe that translation quality with respect to semantics is generally high, though our method, like many SLT systems, can sometimes miss pronouns or struggle with complex tenses. In Table 5, we showcase examples where our approach generates perfect or



near-perfect translations. Finally, in Table 6, we select some examples to compare our model’s output with that of the Uni-Sign (Pose) baseline. These comparisons illustrate improvements in semantic meaning and accuracy, consistent with the quantitative gains in ROUGE and BLEU-4 scores reported in the main paper.

Table 4: Examples of Prediction Errors and Analysis from Geo-Sign (Token Method)

Prediction	Ground Truth	Analysis of Error	Semantic Similarity
她今年50岁。(She is 50 years old.)	他今年四岁。(He is 4 years old.)	Pronoun error: 她 (she) vs. 他 (he). Number error: “50” (50) vs. 四 (four). The prediction gets the topic (age) but is wrong on subject and specific age.	Partial (topic: age)
今天星期五。(Today is Friday.)	今天星期几?(What day of the week is it today?)	Statement vs. Question: Prediction states a specific day. GT asks for the day. Character error: 五 (five) vs. 几 (how many/which).	Partial (topic: day of week)
你什么时候认识小张?(When did you meet Xiao Zhang?)	你和小张什么时候认识的?(When did you AND Xiao Zhang meet?)	Missing words: Prediction lacks “和” (and) and the particle “的”. This subtly changes the meaning from a one-way recognition to a mutual acquaintance.	High
我要去超市买椅子。(I want to go to the supermarket to buy a chair.)	我要去超市买椅子, 你去吗?(I want to go to the supermarket to buy a chair, are you going?)	Missing clause/question: Prediction omits the follow-up question “你去吗?” (are you going?).	High (core statement identical)
下午你们要去做什么?(What are you [plural] going to do in the afternoon?)	他们下午要做什么?(What are they going to do in the afternoon?)	Pronoun error: 你们 (you plural) vs. 他们 (they).	High
下午你们需要做什么?(What do you [plural] need to do this afternoon?)	他们下午要做什么?(What are they going to do this afternoon?)	Pronoun error: 你们 (you plural) vs. 他们 (they). Word choice: 需要 (need) vs. 要 (going to/want to) - subtle semantic shift, GT is more natural for general plans.	High
大家觉得什么时候去买椅子?(When does everyone think we should go buy chairs?)	他们想什么时候去买椅子?(When do they want to go buy chairs?)	Subject error: 大家 (everyone) vs. 他们 (they). Verb choice: 觉得 (feel/think) vs. 想 (want/think).	High
我手表不见了。(My watch is missing.)	这块手表是你的吗?(Is this watch yours?)	Different intent: Prediction states a loss. GT asks about ownership of a present watch. Both are about watches but different scenarios.	Medium (topic: watch)
你手表多少钱?(How much is your watch?)	这块手表多少钱买的?(How much did you buy this watch for?)	Missing context/words: Prediction is a bit abrupt. GT is more complete with “这块” (this) and “买的” (bought for).	High
我发现了他的偶像。(I discovered his idol.)	你看见我的杯子吗?(Did you see my cup?)	Completely different semantic intent and topic. Prediction is about an idol, GT is about a missing cup.	Very Low

Continued on next page

Table 4 – continued from previous page

Prediction	Ground Truth	Analysis of Error	Semantic Similarity
爸爸的房间里大了。 (It has become big in dad's room / Dad's room has become bigger.)	左边的房间是我爸爸妈妈的,他们的房间很大。(The room on the left is my parents', their room is very big.)	Garbled/incomplete prediction: The prediction is grammatically awkward and misses the entire context of the GT.	Low
公司离家远,他为什么打车去公司? (The company is far from home, why does he take a taxi to the company?)	公司离家很远,她为什么不打车? (The company is very far from home, why doesn't she take a taxi?)	Pronoun error: 他 (he) vs. 她 (she). Logic error: Prediction asks why he does take a taxi, GT asks why she doesn't.	Medium
阴天说什么话?天气什么的,明天有事。 (What to say on a cloudy day? Weather something, have things to do tomorrow.)	阴天,电视上说多云,怎么了?明天有事? (Cloudy day, TV says it's overcast, what's up? Got plans tomorrow?)	Nonsensical/Garbled prediction: Prediction is very disjointed and doesn't make sense, while GT is a coherent conversation about weather and plans.	Low
桌子上有饮料,你想喝什么? (There are drinks on the table, what do you want to drink?)	桌上放着很多饮料,你喝什么? (There are many drinks on the table, what do you want to drink?)	Slight phrasing difference: “桌子上有” (On the table there are) vs. “桌上放着很多” (On the table are placed many). GT is slightly more natural. Prediction is still good.	High
我刚才在家里找了一个桌子,不是找了。 (I just looked for a table at home, not looked for.)	你去房间找找,是不是刚才放在桌子上了? (Go look in the room, was it just placed on the table?)	Different speaker and intent: Prediction is a confused statement about searching. GT is a directive and question to someone else.	Low
一个人的癌症会变得很可能。 (A person's cancer will become very possible.)	人体的许多器官都可能发生癌变。 (Many organs of the human body can become cancerous.)	Vague and unnatural prediction: “会变得很可能” is awkward. GT is precise about “organs” and “癌变” (cancerous change).	Medium
老年人通过斑马线时可以走斑马线,而不走汽车。 (When elderly people cross the crosswalk, they can use the crosswalk, and not walk cars.)	一位老人正在慢慢地穿过斑马线,等待的司机却不耐烦地按起了喇叭。 (An old man was slowly crossing the crosswalk, but the waiting driver impatiently honked the horn.)	Nonsensical and irrelevant prediction: “而不走汽车” (and not walk cars) makes no sense. The GT describes a specific scenario.	Very Low

Table 5: Examples of Correct Predictions by Geo-Sign (Token Method)

Reference (Ground Truth)	Our Model Prediction (Perfect Match)
‘今天我想吃面条。’ (Today I want to eat noodles.)	‘今天我想吃面条。’ (Today I want to eat noodles.)
‘苹果是你买的吗?’ (Did you buy the apples?)	‘苹果是你买的吗?’ (Did you buy the apples?)

Continued on next page

Table 5 – continued from previous page

Reference (Ground Truth)	Our Model Prediction (Perfect Match)
‘我昨天有点累。’ (I was a bit tired yesterday.)	‘我昨天有点累。’ (I was a bit tired yesterday.)
‘吃完午饭要多吃点水果。’ (Eat more fruit after lunch.)	‘吃完午饭要多吃点水果。’ (Eat more fruit after lunch.)
‘我的妻子感冒了,我开车带她去医院。’ (My wife has a cold, I will drive her to the hospital.)	‘我的妻子感冒了,我开车去医院。’ (My wife has a cold, I will drive her to the hospital.)
‘我们会通过短信的方式来联系你。’ (We will contact you via text message.)	‘我们会通过短信的方式来联系你。’ (We will contact you via text message.)
‘我们将采用抽查的方式来进行检查。’ (We will use random checks for inspection.)	‘我们将采用抽查的方式来进行检查。’ (We will use random checks for inspection.)
‘你要把握好自己人生的方向。’ (You need to grasp the direction of your own life.)	‘你要把握好自己人生的方向。’ (You need to grasp the direction of your own life.)
‘病历是禁止涂抹、修改的。’ (Medical records are not allowed to be smeared or altered.)	‘病历是禁止涂抹、修改的。’ (Medical records are not allowed to be smeared or altered.)
‘他抛下家人,带着家中财物逃走了。’ (He abandoned his family and fled with the family’s belongings.)	‘他抛下家人,带着家中财物逃走了。’ (He abandoned his family and fled with the family’s belongings.)
‘这间玻璃作坊有一百年历史了。’ (This glass workshop has a hundred years of history.)	‘这间玻璃作坊有一百年历史了。’ (This glass workshop has a hundred years of history.)

Table 6: Comparative Analysis: Geo-Sign (Token) vs. Uni-Sign (Pose) - Selected Examples

Reference (Ground Truth)	Geo-Sign (Token) Prediction	Uni-Sign (Pose) Prediction
‘他每天回来都很累。’ (He is very tired every day when he comes back.)	‘他每天来很累。’ (He comes very tired every day.)	‘他每天来得及很累。’ (He has enough time [to be/and is] very tired every day.)
‘小张,那个女生是你们公司的吗?你对她了解吗?’ (Xiao Zhang, is that girl from your company? Do you know her?)	‘小张那个女生是你公司的吗?’ (Xiao Zhang, is that girl from your company?)	‘那个小张是这家公司负责人,你了解吗?’ (That Xiao Zhang is the person in charge of this company, do you understand/know?)
‘阴天,电视上说多云,怎么了?明天有事?’ (Cloudy day, TV says it’s overcast, what’s up? Got plans tomorrow?)	‘阴天说什么话?天气什么的,明天有事。’ (What to say on a cloudy day? Weather something, have things to do tomorrow.)	‘阴阳怪气地讲着天赋,不知不觉就发生了什么。’ (Sarcastically talking about talent, something happened unknowingly.)
‘但是你一点也没瘦,你做什么运动了?’ (But you haven’t lost any weight, what exercise have you been doing?)	‘但是你没有太吃饱,你去做什么运动?’ (But you didn’t eat too full, what exercise are you going to do?)	‘但是你已经吃不消了,你能做什么呢?’ (But you already can’t stand it, what can you do?)

Continued on next page



**Table 6 – continued from previous page**

<b>Reference (Ground Truth)</b>	<b>Geo-Sign (Token) Prediction</b>	<b>Uni-Sign (Pose) Prediction</b>
‘手机选不到靠近窗户的位置。’ (The mobile phone cannot select a seat near the window.)	‘靠近窗户的位置是不能选择的。’ (Seats near the window cannot be selected.)	‘坐在靠近窗户的位置是不能取手机的。’ (Sitting in a seat near the window, you cannot take out your mobile phone.)
‘他对自己一直高标准严格要求。’ (He has always had high standards and strict requirements for himself.)	‘他对自己有着严格的标准要求。’ (He has strict standard requirements for himself.)	‘他对自己最严格的标准提出了更高的要求。’ (He put forward higher requirements for his strictest standards.)
‘这位厨师制作的甜品,全部受欢迎。’ (The desserts made by this chef are all popular.)	‘厨师的作品很受欢迎。’ (The chef’s work is very popular.)	‘厨师在设计作品时非常受欢迎。’ (The chef is very popular when designing works.)

## G Code Listings

These code examples provide an overview of key components in the architecture to help improve readability of the paper.

1. Defines the skeleton topology and a row-normalised adjacency tensor A.

```
1 def hop_distance(num_nodes, edges, max_hop=1):
2     """Shortest path length (<= max_hop) for every pair of nodes."""
3     adj = np.zeros((num_nodes, num_nodes))
4     for i, j in edges:
5         adj[i, j] = adj[j, i] = 1
6     hop = np.full_like(adj, np.inf, dtype=float)
7     for d in range(max_hop + 1):
8         hop[np.linalg.matrix_power(adj, d) > 0] = d
9     return hop
10
11 class Graph:
12     def __init__(self, layout='hand', strategy='uniform', max_hop=1):
13         self._init_edges(layout)
14         self.hop = hop_distance(self.num_nodes, self.edges, max_hop)
15         self.A = self._adjacency(strategy)
16
17     # --- edge lists -----
18     def _init_edges(self, layout):
19         if layout in ('left', 'right'): # hand (21 joints)
20             self.num_nodes = 21
21             fingers = [[0,1,2,3,4],[0,5,6,7,8],[0,9,10,11,12],
22                       [0,13,14,15,16],[0,17,18,19,20]]
23             links = [(i, i) for i in range(21)]
24             links += [(f[i], f[i+1]) for f in fingers for i in range(
25                 len(f)-1)]
26             self.edges, self.center = links, 0
27         elif layout == 'body': # torso + arms
28             self.num_nodes = 9
29             torso = [(0,i) for i in range(1,5)]
30             arms = [(3,5),(5,7),(4,6),(6,8)]
31             self.edges, self.center = [(i,i) for i in range(9)] +
32                 torso + arms, 0
33         elif layout == 'face_all':
34             self.num_nodes = 16
35             ring = [(i,(i+1)%16) for i in range(16)]
36             self.edges, self.center = [(i,i) for i in range(16)] +
37                 ring, 8
38         else:
39             raise ValueError(f'Unknown layout: {layout}')
40
41     # --- adjacency -----
42     def _adjacency(self, strategy):
43         A = (self.hop <= 1).astype(float) # neighbours 1-hop
44         if strategy == 'uniform':
45             A = A / (A.sum(1, keepdims=True) + 1e-6)
46         elif strategy == 'distance': # 1 / hop distance
47             A = 1 / (self.hop + 1e-6); A[A == np.inf] = 0
48             A = A / (A.sum(1, keepdims=True) + 1e-6)
49         return torch.tensor(A, dtype=torch.float32).unsqueeze(0)
```

2. ST-GCN definition. GCNUnit applies K spatial kernels; STGCNBlock adds a temporal conv and an optional residual path.

```

1 class GCNUnit(nn.Module):
2     def __init__(self, Cin, Cout, A, stride=1, K=None, adaptive=True):
3         super().__init__()
4         self.K = K or A.shape[0] # #adjacency kernels
5         self.A = nn.Parameter(A.clone()) if adaptive else A
6         self.conv = nn.Conv2d(Cin, Cout*self.K, (1,1))
7         self.bn = nn.BatchNorm2d(Cout)
8         self.act = nn.ReLU(inplace=True)
9
10    def forward(self, x): # x: (N,Cin,T,V)
11        N, _, T, V = x.shape
12        x = self.conv(x).view(N, self.K, -1, T, V)
13        x = torch.einsum('nkctv,kvw->nctw', x, self.A) # spatial agg
14        return self.act(self.bn(x))
15
16 class STGCNBlock(nn.Module):
17     def __init__(self, Cin, Cout, A, t_kernel=3, stride=1, residual=
18         True):
19         super().__init__()
20         self.gcn = GCNUnit(Cin, Cout, A)
21         pad = (t_kernel-1)//2
22         self.tcn = nn.Sequential(
23             nn.Conv2d(Cout, Cout, (t_kernel,1), (stride,1), (pad,0)),
24             nn.BatchNorm2d(Cout))
25         self.res = (nn.Identity() if Cin==Cout and stride==1
26                     else nn.Conv2d(Cin, Cout, 1, (stride,1)))
27         self.act = nn.ReLU(inplace=True)
28
29     def forward(self, x):
30         return self.act(self.tcn(self.gcn(x)) + self.res(x))

```

3. Body-to-part residual: body features broadcast to hands / face.models.py – residual context

```

1 body_ctx = None
2 for part in ('body', 'left', 'right', 'face_all'):
3     x = self.proj_linear[part](src_input[part]).permute(0,3,1,2)
4     x = self.gcn_spatial[part](x)
5     if part == 'body':
6         body_ctx = x.detach() # freeze context
7     else:
8         joint = body_ctx[..., idx_map[part]] # select joint
9         x = x + joint.unsqueeze(-1) # broadcast to V
10    out[part] = self.gcn_temporal[part](x)

```

4. Project Euclidean vector to the Poincaré ball.

```

1 class HyperbolicProjection(nn.Module):
2     def __init__(self, d_in, d_out, manifold):
3         super().__init__()
4         self.manifold = manifold
5         self.proj = nn.Linear(d_in, d_out)
6         self.log_scale = nn.Parameter(torch.zeros(())) # ln(scale)
7
8     def forward(self, x):
9         t = self.proj(x) * self.log_scale.exp() # tangent vec
10        return self.manifold.expmmap0(t, project=True)

```

5. Weighted Frechet mean (Algorithm 1).

```

1 def frechet_mean(pts, w, M, max_iter=50, tol=1e-5, eta=1.0):
2     """pts: (N,B,D), w: (N,B) or (N,) -> mu: (B,D)."""

```

```

3 w = w.unsqueeze(-1) / (w.sum(0, keepdim=True) + 1e-8)
4 mu = pts[0].clone()
5 for _ in range(max_iter):
6     v = (w * M.logmap(mu.unsqueeze(0), pts)).sum(0)
7     mu_next = M.expmap(mu, eta*v, project=True)
8     if (M.dist(mu_next, mu) < tol).all(): break
9     mu = mu_next
10 return mu

```

#### 6. Sentence-level text embedding (pooled method).

```

1 mask = txt_mask.unsqueeze(-1).float() # (B,T,1)
2 sent = (emb * mask).sum(1) / mask.sum(1).clamp_min(1) # mean-pool
3 h_text = self.hyp_proj_text(sent)

```

#### 7. Hyperbolic attention (token method).

```

1 h_tok = self.hyp_proj_text(tok_emb) # (B,T,D)
2 q = h_pose.unsqueeze(2) # (B,P,1,D)
3 k = self.manifold.mobius_add(
4     self.manifold.mobius_matvec(W_key, h_tok.unsqueeze(1)),
5     b_key)
6 logits = -self.manifold.dist(q, k) # (B,P,T)
7 logits.masked_fill_(~tok_mask.unsqueeze(1), -1e9)
8 alpha = F.softmax(logits / tau_attn, -1) # weights
9 ctx = self.manifold.weighted_midpoint(h_tok.unsqueeze(1), alpha,
10 [2])

```

#### 8. InfoNCE loss calculated in hyperbolic space.

```

1 class HyperbolicContrastiveLoss(nn.Module):
2     def __init__(self, M, tau0=0.5, m0=0.1):
3         super().__init__()
4         self.M = M
5         self.log_tau = nn.Parameter(torch.logit(torch.tensor(tau0/2)))
6         self.m = nn.Parameter(torch.tensor(m0))
7
8     def forward(self, a, b): # (B,D) pairs
9         d = self.M.dist(a.unsqueeze(1), b.unsqueeze(0))
10        s = -d / (torch.sigmoid(self.log_tau)*2 + 0.01)
11        s -= (~torch.eye(len(a), dtype=torch.bool, device=a.device)) *
12            self.m.clamp_min(0)
13        target = torch.arange(len(a), device=a.device)
14        return F.cross_entropy(s, target)

```

#### 9. Manifold with learnable curvature initialisation.

```

1 self.manifold = geoopt.PoincareBall(c=cfg.init_c, learnable=True)

```

#### 10. Dynamic alpha for loss blending.

```

1 progress = self.global_step / max(1, self.total_steps)
2 alpha_base = cfg.alpha_init + 0.05 * progress # <= 0.9
3 alpha_learn = 0.2 * torch.sigmoid(self.alpha_logit)
4 alpha = (alpha_base + alpha_learn).clamp(0.1, 0.99)
5 loss = alpha * ce_loss + (1 - alpha) * hyp_loss

```

## References

- [1] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

- [2] Zhigang Chen, Benjia Zhou, Yiqing Huang, Jun Wan, Yibo Hu, Hailin Shi, Yanyan Liang, Zhen Lei, and Du Zhang. C<sup>2</sup>rl: Content and context representation learning for gloss-free sign language translation and retrieval. In *arxiv*, 2024.
- [3] Zhigang Chen, Benjia Zhou, Jun Li, Jun Wan, Zhen Lei, Ning Jiang, Quan Lu, and Guoqing Zhao. Factorized learning assisted with large language model for gloss-free sign language translation. In *LREC-COLING*, 2024.
- [4] Amanda Duarte, Shruti Palaskar, Lucas Ventura, Deepti Ghadiyaram, Kenneth DeHaan, Florian Metze, Jordi Torres, and Xavier Giro-i Nieto. How2sign: a large-scale multimodal dataset for continuous american sign language. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.
- [5] Octavian Ganeva, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- [6] Jia Gong, Lin Geng Foo, Yixuan He, Hossein Rahmani, and Jun Liu. Llms are good sign language translators. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2024.
- [7] Tao Jiang, Peng Lu, Li Zhang, Ning Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. RTMPose: Real-time multi-person pose estimation based on mmpose. In *arxiv*, 2023.
- [8] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. Whole-body human pose estimation in the wild. In *ECCV*, 2020.
- [9] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch, 2020.
- [10] Zecheng Li, Wengang Zhou, Weichao Zhao, Kepeng Wu, Hezhen Hu, and Houqiang Li. Uni-sign: Toward unified sign language understanding at scale. In *arXiv preprint arXiv:2501.15187*, 2025.
- [11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *arXiv preprint arXiv:1711.05101*, 2017.
- [12] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.
- [14] Garrett Tanzer and Biao Zhang. Youtube-sl-25: A large-scale, open-domain multilingual sign language parallel corpus. In *arxiv*, 2024.
- [15] Dave Uthus, Garrett Tanzer, and Manfred Georg. Youtube-asl: A large-scale, open-domain american sign language-english parallel corpus. In *Advances in Neural Information Processing Systems*, 2024.
- [16] Ryan Wong, Necati Cihan Camgoz, and Richard Bowden. Sign2GPT: Leveraging large language models for gloss-free sign language translation. In *ICLR*, 2024.
- [17] Ryan Wong, Necati Cihan Camgoz, and Richard Bowden. Signrep: Enhancing self-supervised sign representations. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2025.
- [18] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.
- [19] Aoxiong Yin, Tianyun Zhong, Li Tang, Weike Jin, Tao Jin, and Zhou Zhao. Gloss attention for gloss-free sign language translation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

- [20] Benjia Zhou, Zhigang Chen, Albert Clapés, Jun Wan, Yanyan Liang, Sergio Escalera, Zhen Lei, and Du Zhang. Gloss-free sign language translation: Improving from visual-language pretraining. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.
- [21] Wengang Zhou, Weichao Zhao, Hezhen Hu, Zecheng Li, and Houqiang Li. Scaling up multi-modal pre-training for sign language understanding. In *arxiv*, 2024.