Translating Signals to Languages for sEMG-Based Activity Recognition

Supplementary Material

1. Additional Ablation Experiments.

This section presents additional ablation studies conducted on the GRABMyo dataset. All experiments follow the same setup as in the main paper unless otherwise stated. An illustration of the GRABMyo activity classes is provided in Fig. 1.

Effect of LoRA. To efficiently adapt large language models for sEMG-based activity understanding while keeping their pre-trained weights intact, we employ LoRA for fine-tuning. To validate its effectiveness, we compare it with full-parameter fine-tuning ("Full Tuning"), where all parameters are updated. As shown in Table 1, LoRA achieves higher accuracy, whereas replacing it with full tuning causes a notable performance drop. This demonstrates that LoRA effectively transfers linguistic priors and reasoning capabilities from natural language pre-training to sEMG activity inference while preserving pre-trained semantic knowledge.

Table 1. Effect of LoRA in terms of ACC (%) and STD.

Model	Single-finger		Multi-finger		Wrist		Rest		Overall	
	ACC	STD	ACC	STD	ACC	STD	ACC	STD	ACC	STD
Full Tuning	81.64	0.02	83.37	0.01	84.45	0.01	86.60	0.02	83.30	0.02
Full Model (Ours)	**92.25**	0.02	**94.50**	0.01	**95.19**	0.01	**96.82**	0.01	**95.14**	0.01

Effect of using Different LLM Backbone Sizes. To assess the dependence of LLM-sEMG on backbone scale, we compare model variants using LLaMA-7B and LLaMA-13B as the language reasoning module. All models are trained under identical VQ-VAE settings, LoRA configurations, and tokenization.

Table 2. Effect of different LLM backbone sizes in terms of ACC (%) and STD.

Backbone	Single-finger		Multi-finger		Wrist		Rest		Overall	
	ACC	STD	ACC	STD	ACC	STD	ACC	STD	ACC	STD
LLaMA 7B	89.52	0.02	91.15	0.02	93.45	0.01	95.20	0.01	92.57	0.02
LLaMA 13B	**92.25**	0.02	**94.50**	0.01	**95.19**	0.01	**96.82**	0.01	**95.14**	0.01

As shown in Table 2, using a larger LLM generally improves performance, since larger models contain richer semantic priors.

Effect of Token Budget. The proposed adaptive token allocation mechanism distributes a fixed total number of tokens across temporal slices according to residual energy. We evaluate multiple token budgets $T_{\max}$ to analyze the influence.

As shown in Table 3, using an excessively small token budget degrades performance, since the model is forced to compress rich transient dynamics into too few tokens, leading to information loss in highly non-stationary regions. Increasing $T_{\max}$ gradually improves accuracy, and the best performance is achieved around $T_{\max} = 128$, where the emergent "sEMG language" captures sufficient temporal granularity while avoiding redundant representations.

Table 3. Effect of token budget in terms of ACC (%) and STD.

Token Budget	Single-finger		Multi-finger		Wrist		Rest		Overall	
	ACC	STD	ACC	STD	ACC	STD	ACC	STD	ACC	STD
$T = 64$	86.32	0.02	88.54	0.02	90.17	0.01	95.24	0.02	89.56	0.02
$T = 96$	89.25	0.02	90.37	0.02	92.75	0.01	95.90	0.01	92.54	0.01
$T = 128$	**92.25**	0.02	**94.50**	0.01	**95.19**	0.01	**96.82**	0.01	**95.14**	0.01
$T = 160$	91.90	0.02	94.02	0.02	94.85	0.01	96.25	0.01	94.81	0.01

Effect of Codebook Size. We investigate the impact of different codebook sizes K on emergent language quality and downstream gesture recognition.

Table 4. Effect of codebook size in terms of ACC (%) and STD.

Codebook Size	Single-finger		Multi-finger		Wrist		Rest		Overall	
	ACC	STD	ACC	STD	ACC	STD	ACC	STD	ACC	STD
$K = 256$	89.35	0.02	91.26	0.02	92.85	0.01	95.74	0.01	91.78	0.02
$K = 512$	**92.25**	0.02	**94.50**	0.01	**95.19**	0.01	**96.82**	0.01	**95.14**	0.01
$K = 1024$	91.83	0.02	93.65	0.02	94.77	0.01	96.20	0.01	94.59	0.01

As shown in Table 4, the $K = 512$ configuration achieves the best overall performance. A smaller codebook lacks representational capacity, while a larger one introduces redundancy and reduces token utilization efficiency. This suggests that a moderately sized codebook strikes the right balance between expressiveness and stability for "sEMG language" emergence.

Effect of Removing Language Emergence Losses Together. In the main paper, we disentangle the contributions of each linguistic bias individually. We further evaluate a more extreme variant where all language-emergence losses are removed simultaneously.

Table 5. Effect of removing all language-emergence losses in terms of ACC (%) and STD.

Model Variant	Single-finger		Multi-finger		Wrist		Rest		Overall	
	ACC	STD	ACC	STD	ACC	STD	ACC	STD	ACC	STD
w/o All Language Losses	84.30	0.03	86.72	0.03	88.30	0.02	89.12	0.02	86.85	0.02
Full Model	**92.25**	0.02	**94.50**	0.01	**95.19**	0.01	**96.82**	0.01	**95.14**	0.01

As shown in Table 5, removing all language-emergence losses dramatically reduces performance. Without linguistic priors, token usage collapses into a frequency-skewed distribution dominated by a few overused entries, leading to poorly structured and weakly discriminative "sEMG language." This confirms that linguistic biases collectively play a central role in guiding a stable and interpretable language emergence process.

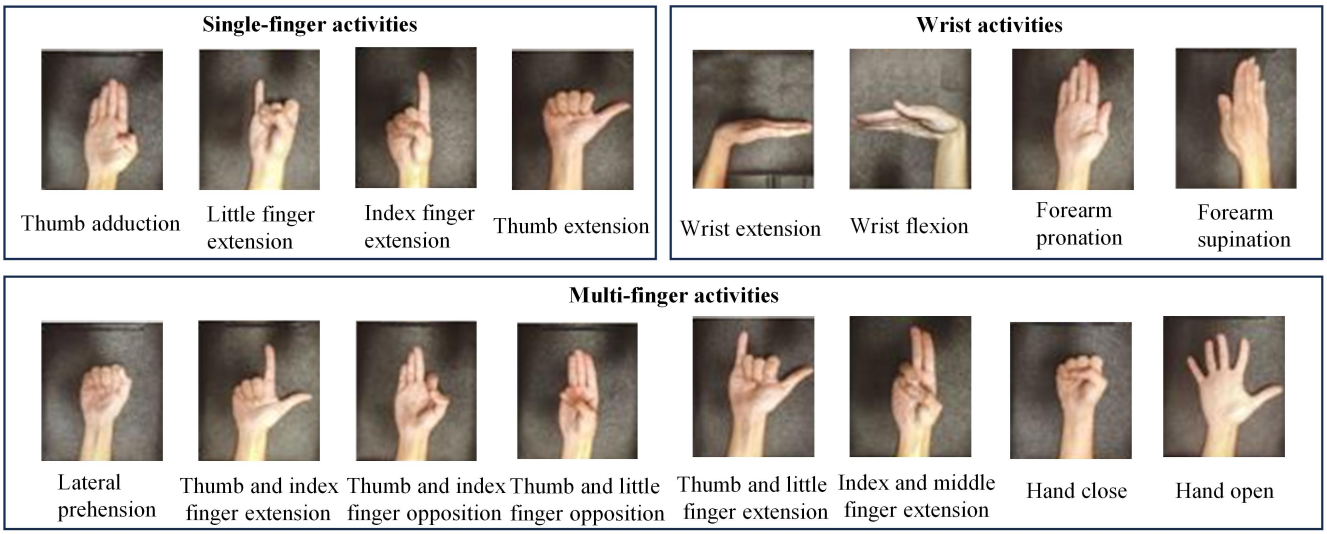

Figure 1. Illustration of activity classes in the GRABMyo dataset.

2. Analysis of Freezing Different Components during Warm-up

To further examine the role of the warm-up stage in our iterated learning process, we conducted an additional experiment that investigates the effect of freezing both the codebook and encoder simultaneously, instead of freezing only the codebook as in the main paper. The motivation behind this test is to verify whether encoder freezing affects the knowledge transfer efficiency between generations. During the standard warm-up phase, only the codebook is frozen, while the newly initialized decoder is trained to adapt to the existing token space defined by the frozen codebook. In contrast, in the variant where both the codebook and the encoder are frozen, only the decoder is updated.

Table 6. Effect of freezing both the codebook and encoder during the warm-up phase on the GRABMyo dataset in terms of ACC (%) and STD.

Setting	Single-finger		Multi-finger		Wrist		Rest		Overall	
	ACC	STD	ACC	STD	ACC	STD	ACC	STD	ACC	STD
Freeze Codebook	**92.25**	0.02	**94.50**	0.01	**95.19**	0.01	**96.82**	0.01	**95.14**	0.01
Freeze Codebook+Encoder	89.62	0.02	91.94	0.02	93.28	0.01	94.65	0.01	92.37	0.01

As shown in Table 6, simultaneously freezing the codebook and encoder results in a clear performance degradation, with the overall accuracy dropping from 95.14% to 92.37%. This decline occurs because the decoder receives both fixed encoder outputs and a fixed token space, leaving no room to adapt to the inherited semantic structures established in the previous generation. Without the ability to adjust to these evolving representations, the decoder is forced to reconstruct signals solely from frozen encoder latent features, preventing it from forming meaningful sEMG-to-language mappings. Consequently, the intergenerational transfer of linguistic structure is weakened, and the emergence of a stable "sEMG language" is adversely affected. Overall, these findings demonstrate that updating the decoder while freezing only the codebook is crucial for effective warm-up. This setting allows the new decoder to better fit the inherited token space and preserves smooth semantic transfer across generations, ensuring stable and coherent evolution of the emergent "sEMG language."

3. More Implementation Details

Following prior work [3, 5, 7], the loss weights are set to $\lambda_1 = 0.02$ and $\lambda_2 = 0.1$. During iterated learning, only the decoder is reinitialized while the codebook remains frozen, allowing each new decoder to interpret the existing token space without disrupting previously formed semantics. The entire training process spans 200 epochs, and residual energies for adaptive token allocation are recalculated once per epoch. Each iteration begins with a 200-step warm-up phase, followed by an 8000-step knowledge-transfer stage. For adaptive token discretization, we use a fixed token budget of 128. During LLM fine-tuning, LoRA is applied to LLaMA-13B with a rank of $r = 16$, scaling factor $\alpha = 32$, while all pretrained parameters remain frozen.

4. Additional Details about Human-Linguistic Bias

We supplement the main paper with additional mathematical formulations, implementation details, and design considerations for the Human-Linguistic Bias Guided Learning module. We further elaborate on the computation of Zipf-weighted sampling, the construction of the token co-occurrence matrix, the correlation-based context loss, and the interactions among these components across successive generations during iterative learning.

Zipf's Law Prior. As detailed in the main paper, the evolution of the emergent "sEMG language" is encouraged to follow the frequency imbalance characteristic of natural languages [1, 2]. To initialize such a bias, the first 25% of the iterative learning process adopts a Zipf-weighted sampling strategy. Given the encoder output logits $z \in \mathbb{R}^K$, instead of performing standard nearest-neighbor quantization, we apply a Zipf-biased sampling distribution:

$$P_{\text{Zipf}}(i) = \frac{(i+\gamma)^{-\beta}}{\sum_{j=1}^{K}(j+\gamma)^{-\beta}} \tag{1}$$

where token indices are sorted by their usage frequency from previous updates. The quantized token index is drawn as $k = \arg\max_i \left(z_i + \lambda \cdot \log P_{\text{Zipf}}(i)\right)$, where λ controls the strength of Zipf prior during sampling. This step biases the early generations toward forming a small set of highly reusable core-tokens, thereby inducing an initial Zipf-shaped structure for the emergent sEMG lexicon. During later training stages (after Zipf-weighted sampling), the target Zipf distribution is:

$$D_{\text{Zipf}}(i) = \frac{(i+\gamma)^{-\beta}}{\sum_{j=1}^{K}(j+\gamma)^{-\beta}} \tag{2}$$

To align the token usage of the emergent "sEMG language" with the Zipfian structure, we minimize the Jensen–Shannon divergence between the empirical and theoretical distributions as $\mathcal{L}_{\text{zipf}} = \text{JS}(D_{\text{freq}} \parallel D_{\text{Zipf}})$. Through this regularization, the "sEMG language" is encouraged to evolve toward a Zipf-like distribution.

Context-Sensitivity Prior. Unlike prior work that defines correlation via manually constructed token pairs, we compute contextual correlation using cosine similarity in the learned embedding space [4, 6]. This formulation better aligns with the semantic structure of the emergent "sEMG language" and is more consistent with the representation space of large language models. Formally, recall that in the main paper we introduce the context-sensitivity loss:

$$\mathcal{L}_{\text{context}} = 1 - \frac{1}{N}\sum_{n=1}^{N} \text{Corr}(t_n) \tag{3}$$

where N denotes the batch size, and $\text{Corr}(t_n)$ measures the degree of contextual correlation within the n-th token sequence $t_n = (t_{n,1}, \ldots, t_{n,L})$. In the following, we provide the detailed formulation of the correlation operator $\text{Corr}(\cdot)$ used in our framework.

Let $e_{t_{n,j}} \in \mathbb{R}^D$ denote the embedding vector of the token at position j in the n-th sequence, obtained from the learned codebook. To capture local contextual continuity, we define $\text{Corr}(t_n)$ as the average cosine similarity between the embeddings of neighboring tokens:

$$\text{Corr}(t_n) = \frac{1}{L-1}\sum_{j=1}^{L-1} \text{sim}\left(e_{t_{n,j}}, e_{t_{n,j+1}}\right) \tag{4}$$

where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity. This definition encourages adjacent tokens in a sequence to lie in semantically compatible regions of the embedding space, thereby inducing local semantic continuity in the emergent "sEMG language."

In practice, the cosine similarities in Eq. (4) are computed in a vectorized manner over the entire batch for efficiency. Since cosine similarity is naturally bounded in $[-1, 1]$, $\text{Corr}(t_n)$ also remains bounded, which stabilizes the optimization of $\mathcal{L}_{\text{context}}$ in Eq. (3). Furthermore, we found that restricting the correlation computation to immediate neighbors strikes a good balance between capturing local dependencies and avoiding excessive smoothing of token usage patterns. Extending the window size to include longer-range interactions is possible but did not bring significant additional benefits in our experiments.

As noted in the co-occurrence analysis in the main paper, during iterative learning, for the model at generation t, we compute the token co-occurrence frequencies within a fixed-size contextual window to construct the co-occurrence matrix, formulated as follows:

$$M_{ij}^{(t)} = \frac{\text{count}(k_i, k_j)}{\sum_{p=1}^{K}\sum_{q=1}^{K}\text{count}(k_p, k_q)} \tag{5}$$

where k_i and k_j denote the indices of the i-th and j-th tokens in the codebook, $\text{count}(k_i, k_j)$ represents the number of times tokens k_i and k_j co-occur within the contextual window, and K is the size of the codebook.

5. Further Details on Residual-Based Adaptive Token Allocation

To complement the description in the main paper, we provide the full formulation and implementation details of the proposed feature-preserving tokenization strategy. Given an sEMG segment $\mathbf{x}$, we uniformly divide it into S temporal slices $\{\mathbf{x}_1, \ldots, \mathbf{x}_S\}$, each capturing local temporal dynamics. For the s-th slice, the VQ-VAE decoder produces a reconstruction $\hat{\mathbf{x}}_s$, and the residual energy is computed as $R_s = \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2$, which serves as an estimate of the slice's local information density: large residuals typically correspond to abrupt muscle activations or transitions, whereas small residuals indicate stable contractions or resting phases. To allocate a fixed global token budget $T_{\max}$ across slices in proportion to their importance, we normalize residual energies into a probability distribution $P_s = R_s / \sum_{i=1}^{S} R_i$, and assign an initial token count $T_s = T_{\max} \cdot P_s$, rounded to the nearest integer. Since directly using P_s may lead to some low-residual slices receiving zero tokens, which undermines representational completeness and temporal consistency, we enforce a minimum-coverage constraint by setting $n_s = \max(1, T_s)$ and adjusting the resulting allocation to satisfy $\sum_{s=1}^{S} n_s = T_{\max}$.

When the total allocation exceeds the budget, surplus tokens are removed from slices with the smallest residuals to preserve high-information regions. This residual-guided allocation ensures that information-rich transient phases receive fine-grained tokenization while stable phases remain sparsely encoded, enabling the emergent "sEMG language" to maintain the non-stationary and burst-like characteristics of sEMG signals, reduce redundancy in steady regions, and preserve discriminative temporal dynamics essential for downstream activity recognition.

References

[1] Bernat Corominas-Murtra, Jordi Fortuny, and Ricard V. Solé. Emergence of zipf's law in the evolution of communication. *Physical Review E*, 83:036115, 2011. 3

[2] Ramon Ferrer i Cancho and Ricard V. Solé. Least effort and the origins of scaling in human language. *Proceedings of the National Academy of Sciences*, 100(3):788–791, 2003. 3

[3] Lee Kezar, Zed Sehyr, and Jesse Thomason. Phonological representation learning for isolated signs improves out-of-vocabulary generalization. *arXiv preprint arXiv:2509.04745*, 2025. 2

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3111–3119, 2013. 3

[5] Evonne Ng, Sanjay Subramanian, Dan Klein, Angjoo Kanazawa, Trevor Darrell, and Shiry Ginosar. Can language models learn to listen? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10083–10093, 2023. 2

[6] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6306–6315, 2017. 3

[7] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. *arXiv preprint arXiv:2301.06052*, 2023. 2