Evaluating Sign Language Representation Learning with Large Language Models

Fredrik Malmberg
fmalmb@kth.se

Anna Klezovich
annkle@kth.se

Jonas Beskow
beskow@kth.se

Division of Speech, Music and Hearing, KTH, Sweden

Abstract

This study explores integrating sign language representation learning with large language models (LLMs) in natural language processing (NLP). Using a Vector Quantized Variational Autoencoder (VQ-VAE) trained on Swedish Sign Language (SSL) 2D keypoints, it examines the effectiveness of the learnt motion codebook for SSL recognition tasks and whether the performance of an LLM, trained on this codebook to generate hand shape descriptions, can be a tool to guide the development of more powerful models for learning motion representations from sign language data.

1 Introduction

1.1 Background

Recent advancements in representation learning, such as VQ-VAEs (van den Oord et al., 2018), have shown promise in capturing the complexities of sign language gestures (Xie et al., 2022). LLMs, on the other hand, have revolutionized text-based natural language understanding. This study aims to explore the combination of these two fields by integrating sign language representation learning with LLMs.

1.2 Problem Statement

A key challenge in sign language processing is the effective representation of intricate movements that capture linguistic differences. VQ-VAEs are typically trained to reconstruct the input data but evaluating the result based on reconstruction loss poses several issues: it may not directly correlate with the usefulness of the learned motion codes, more powerful models may achieve lower reconstruction errors without necessarily producing better representations, and it restricts the exploration of different loss functions. Furthermore, qualitative analysis of VQ-VAE outputs is both time-consuming and tends to focus on the decoder's performance, which is not directly relevant to the goal of creating effective motion representations. This project aims to evaluate the usefulness of motion codes generated by a VQ-VAE trained on Swedish Sign Language (SSL) by employing LLMs for quantitative analysis.

2 Methodology

2.1 VQ-VAE for SSL Representation

A VQ-VAE is well-suited for compressing sequential data, such as sign language utterances, into a discrete codebook, which can later be used in downstream tasks like the language model integration.

Using an architecture based on (Malmberg et al., 2024) and inspired by (Jiang et al., 2023), we train a 2D sign motion tokenizer. The model consists of an encoder $\mathcal{E}$ and decoder $\mathcal{D}$, with a discrete latent space for structured motion representation. The encoder $\mathcal{E}$, based on 1D convolutions, processes sequences of 2D sign keypoints (length M) into latent vectors $\hat{z}^{1:L} = \mathcal{E}(m^{1:M})$, downsampling the sequence by a factor of l to produce a sequence of motion codes of length $L = M/l$. These latent vectors are quantized into a codebook $Z = \{z^i\}_{i=1}^{K}$ with K learnable embeddings of dimension d. The decoder $\mathcal{D}$ reconstructs the original sequence using these embeddings.

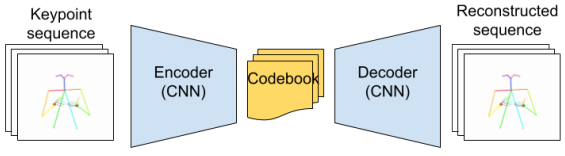

Figure 1: The VQ-VAE consists of an encoder that takes sequences of poses as inputs, a codebook that captures the motion codes and a decoder that outputs reconstructed sequences.

Optimization uses reconstruction loss $\mathcal{L}_r$ and commitment loss $\mathcal{L}_c$ to stabilize embeddings. We also apply exponential moving average (EMA) (Razavi et al., 2019) for improved embedding qual-

ity instead of the embedding loss ($\mathcal{L}_e$). To prevent codebook collapse we use codebook resets to reinitialize unused codes as in Jukebox (Dhariwal et al., 2020) and HVQ-VAE (Williams et al., 2020).

2.1.1 Dataset and Preprocessing

This study utilizes individual signs from the Swedish Sign Language (STS) Dictionary (Svenskt teckenspråkslexikon, 2024), which comprises 21,000 entries. Each entry features a video of the sign, the gloss, a shape description, variants, and example sentences.

DWPose (Yang et al., 2023) was used to extract 2D pose keypoints from video frames and focused on 56 keypoints related to the upper body, arms, and hands (see Figure 1). The keypoints were converted to relative locations given parent joint and normalized using mean and standard deviation of the training set. For preprocessing the video data, we center the sequence of poses on the keypoint connecting the body to the neck in the first frame, then scale the pose based on the shoulder width.

Instead of using glosses we use shape descriptions as targets for the LLM. The rationale behind this is that the shape description contains information related to the motions of the sign but also that using glosses would lead to having unique instances in the dataset and very limited overlap in labels, between training and test data. As the shape description for the selected dataset is only available in Swedish the text strings were translated to English using the Google AJAX Language API.

Example of translated shape description:

> *Clasped hands, forward and facing each other, exchange places with each other maintaining contact on top of each other.*

The shape descriptions contains description of movement, position and orientation of hands, references to different parts of the body, handshapes and finger positions and different types of interactions between the hands, or hands and the body. The dataset was divided into training, validation, and test sets with an 80/10/10 split for the VQ-VAE and the language model.

2.1.2 VQ-VAE training

To evaluate the usability of learnt codebooks with a language model, we trained a number of VQ-VAE models using the architecture described in Section 2.1.1, each with a different setup in terms of loss function, codebook size K, embedding dimension d and number of convolutional layers in $\mathcal{E}$ and $\mathcal{D}$.

Following Jiang et al., 2023 we defined a base VQ-VAE model with the following parameters:

Parameter	Value
Codebook size, K	512
Embedding dim, d	512
$\mathcal{E}$ and $\mathcal{D}$ depth	3
Loss	Euclidean

Table 1: Base VQ-VAE Model Parameters

2.2 Large Language Model Integration

As in Jiang et al. 2023, we used T5 (Raffel et al., 2023) (Text-To-Text Transfer Transformer) as the base model. T5 is a versatile sequence-to-sequence model pre-trained on a diverse range of text tasks, where both input and output are treated as text sequences. It leverages the transformer architecture (Vaswani et al., 2017), which uses self-attention mechanisms to handle dependencies in data.

2.2.1 Adding Motion Tokens and Embeddings

Since T5 was originally trained only on text data, we extended its tokenizer to include motion tokens representing the VQ-VAE codebook, as well as start and stop motion tokens. These tokens were added to the embedding layer. To maintain the integrity of the pre-trained text embeddings, we froze the original embeddings during training, which prevents catastrophic forgetting. The newly added motion embeddings and the rest of the model's layers were fine-tuned to learn the new task.

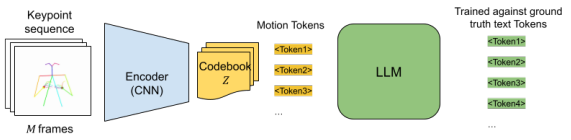

Figure 2: The learnt motion codes for each sign are used as input to the language model and trained against the corresponding shape descriptions for that sign.

2.2.2 LLM Fine-tuning

The language model was trained in a supervised manner where a tokenized motion sequence for a sign served as input and the corresponding tokenized shape description was the target. The training was conducted with a batch size of 8, over 10

epochs, using a learning rate of 2×10^{-4}. We employed cross-entropy loss to guide the training process .

2.3 Evaluation Metrics

The performance of the language model was evaluated on test data using Rouge score (Lin, 2004), BERTScore (Zhang et al., 2020) alongside the test set loss to provide a comprehensive evaluation of the model's performance, covering both literal overlap and semantic accuracy, as well as the model's prediction accuracy.

Rouge Score: Rouge (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of text summaries by comparing them with reference summaries. We will use Rouge-1 that focuses on unigram (single word) overlap.

BERTScore: BERTScore evaluates text generation quality using contextual embeddings from the BERT model. Unlike Rouge, which relies on exact n-gram matches, BERTScore uses cosine similarity between BERT embeddings of the generated and reference text, capturing semantic similarity even when different words or phrases are used.

Test Set Loss: For the T5 model, the test set loss is calculated using cross-entropy loss, which measures the difference between the predicted probability distribution and the true distribution over the vocabulary.

3 Results

3.1 Comparison with a Naive Baseline

To evaluate the effectiveness of the fine-tuned LLMs we included a naive baseline based on random codes where a code generator outputs a code sequence of correct length for the given sample but randomly selects codes from the codebook. This gives the language model information regarding sequence length but does not provide any information on the actual motions performed. These random motion codes were used to train an LLM in a similar fashion as the other models and the results will be referred to as *Naive*. The purpose of this baseline is to understand how much of the final performance simply comes from training a language model to output reasonable descriptions compared to providing it with the information encoded in the learned motion codes coming from a VQ-VAE.

3.2 Results for different VQ-VAE setups

3.2.1 Loss Function

To evaluate the impact of the loss function used to train the VQ-VAE we compare the base model (Table 1) to a model using MSE loss as well as the naive baseline (see Table 2). The VQ-VAE trained with MSE lets the LLM achieve a better test loss and F1 for BERTScore but the same is not true for the F1 score for Rouge-1.

Loss function	Loss	F1 (Rouge-1)	F1(Bert)
MSE	**0.4170**	0.3683	**0.8846**
Euclidean	0.4201	**0.3751**	0.8826
Naive	0.4373	0.3611	0.8714

Table 2: Performance for different loss functions for the VQ-VAE training sorted by LLM Test Loss.

3.2.2 Codebook Size K

The impact of modifying the codebook is presented in Table 3 and somewhat contrary to intuition it can be seen that the performance of the LLM is not strictly increasing with increasing codebook size.

K	Loss	F1 (Rouge-1)	F1 (Bert)
1024	**0.4097**	**0.3852**	**0.8849**
2048	0.4146	0.3797	0.8845
512	0.4201	0.3751	0.8826
Naive	0.4373	0.3611	0.8714

Table 3: Performance for different codebook sizes sorted by LLM Test Loss.

3.2.3 Depth of Encoder and Decoder

As can be seen in Table 4 the impact of the depth of the encoder and decoder is ambiguous and a deeper model does not necessarily produce better codes.

Depth	Loss	F1 (Rouge-1)	F1 (Bert)
3	**0.4201**	0.3751	**0.8826**
2	0.4235	0.3717	0.8773
4	0.4360	**0.3819**	0.8754
Naive	0.4373	0.3611	0.8714

Table 4: Performance for different depth of the encoder and decoder sorted by LLM Test Loss.

3.2.4 Embedding Dimension d

Decreasing the embedding dimension from the base model's 512 improves the learnt representations up to a point as can be seen in Table 5.

d	Loss	F1 (Rouge-1)	F1 (Bert)
128	**0.4049**	**0.3987**	**0.8905**
64	0.4196	0.3689	0.8826
512	0.4201	0.3751	0.8826
256	0.4253	0.3834	0.8791
Naive	0.4373	0.3611	0.8714

Table 5: Performance for different embedding dimensions for the codes sorted by LLM Test Loss.

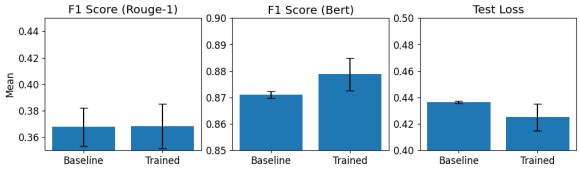

Figure 3: The best performing model (Embedding dimension 128) compared to the naive baseline, showing mean and standard deviation over 10 VQ-VAE and LLM training runs.

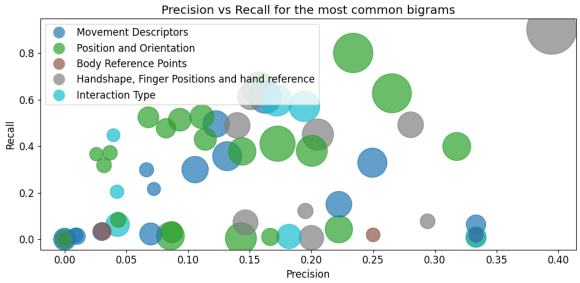

Figure 4: Precision Vs. Recall on the test data for the 100 most common bigrams in the training data. Size by occurrence and color according to type.

4 Discussion

The selection of parameters for the VQ-VAE, such as embedding dimension and codebook size, clearly impacts the usefulness of the learnt representations while changing loss function and modifying the depth of the VQ-VAE did not lead to an improvement in all the metrics. It might be reasonable to believe that a more powerful encoder and decoder should learn more useful representations. However, the depth of the network impacts the length of the motion code sequence. For a deeper architecture the compression increases by a factor of two which means a deeper architecture needs to capture twice as much information per token in which case the codebook, rather than the architecture, likely becomes the bottleneck. As for the bigger codebook size that increased our scores, it shows that using Lookup Free Quantization (Yu et al., 2024), that instead of vectors uses scalars, while ensuring bigger codebook size on the same resources, could work on our data after grid searching for the optimal hyperparameters.

Interestingly the shallow architecture reached a much lower reconstruction loss on the test set for the VQ-VAE while also creating subjectively more responsive reconstruction videos. Regardless of these results, the learnt motion codes for this model performed worse when used with the LLM (see Table 4). As the improvement over the naive baseline was limited we retrained the best performing model and the random baseline 10 times to measure changes in performance. As can be seen in Figure 3 the difference between the models is meaningful for BERTScore and test loss but not for Rouge-1.

We were also interested in understanding which types of shape descriptions where easier or harder for the model to identify and tested the model's performance relating to precision and recall for bigrams in the test dataset. However, as can be seen in Figure 4 there is no clear pattern to which bigram types the model accurately captured.

5 Conclusion

The comparison between the language models fine-tuned on learned motion codes and the naive baseline indicates that the learned motion codes do convey useful information related to the textual shape descriptions of the signs, but more importantly that the degree of usefulness of these codes can be used to guide the design of the representation learning setup. Even though the performance of the models trained on learnt codes is only slightly better than that of the one trained on random codes, it signals that using this way of benchmarking motion codes, and thereby the learnt representations, could be a valuable addition to the toolbox when trying to create the best possible representation learning setup.

This study evaluated VQ-VAE models by varying the setup along one parameter at the time. Future research could benefit from combining these results and mix varying decoder architectures, quantizers, codebook sizes, and codebook embedding dimensions to find even better setups for representation learning while continuously guiding the improvement using the proposed LLM evaluation. However, it should be noted that with the current setup, several training runs per model are needed to ensure a proper measure of its performance which somewhat limits the usefulness of the method.

References

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A generative model for music. *Preprint*, arXiv:2005.00341.

Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2023. Motiongpt: Human motion as a foreign language. *Preprint*, arXiv:2306.14795.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Fredrik Malmberg, Anna Klezovich, Johanna Mesch, and Jonas Beskow. 2024. Exploring latent sign language representations with isolated signs, sentences and in-the-wild data. In *Proceedings of the LREC-COLING 2024 11th Workshop on the Representation and Processing of Sign Languages: Evaluation of Sign Language Resources*, pages 219–224.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *Preprint*, arXiv:1906.00446.

Svenskt teckenspråkslexikon. 2024. Swedish Sign Language Dictionary online. Department of Linguistics, Stockholm University. teckenspraklexikon.su.se.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2018. Neural discrete representation learning. *Preprint*, arXiv:1711.00937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Will Williams, Sam Ringer, Tom Ash, John Hughes, David MacLeod, and Jamie Dougherty. 2020. Hierarchical quantized autoencoders. *Preprint*, arXiv:2002.08111.

Pan Xie, Qipeng Zhang, Zexian Li, Hao Tang, Yao Du, and Xiaohui Hu. 2022. Vector quantized diffusion model with codeunet for text-to-sign pose sequences generation. *arXiv preprint arXiv:2208.09141*.

Zhendong Yang, Ailing Zeng, Chun Yuan, and Yu Li. 2023. Effective whole-body pose estimation with two-stages distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4210–4220.

Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. 2024. Language model beats diffusion – tokenizer is key to visual generation. *Preprint*, arXiv:2310.05737.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.