Generating Sign Language Poses from HamNoSys and Natural Language Descriptions

Santiago Máximo, Luis Chiruzzo
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Uruguay
santiagomaximoc@gmail.com, luischir@fing.edu.uy

Abstract

One of the steps involved in the process of sign language generation is generating a sequence of poses that represent the signs. This paper presents a method for using textual information to improve the translation of signs in HamNoSys format into sequences of poses. The method comprises a description generator that translates HamNoSys into a textual description, an LLM fine-tuned to the task of predicting a pose sequence from a HamNoSys description, and a VQ-VAE network that encodes and decodes pose sequences as a list of discrete symbols. Our experiments found that even using simple dictionary descriptions of HamNoSys, it is possible to improve the predictions of pose sequences by leveraging the information from a pretrained LLM.

Keywords: Sign Language Generation, HamNoSys, VQ-VAE, LLM

1. Introduction

Sign Languages (SL) are natural languages used daily by millions of people, especially in the Deaf and Hard-of-Hearing (DHH) community, which are produced in the visual-spatial modality instead of the auditory modality of spoken languages (Baker, 2016). This makes these languages much harder to process from a computational perspective, and despite huge advances in NLP in recent years, the automatic generation of SL is still a challenge.

One possible approach to translate from an oral language into a SL could involve the following steps:

1. Translating the oral language into an intermediate notation.
2. Generating a sequence of poses (sets of spacial keypoints indicating the position of the body, joints, fingers, etc.) from the intermediate notation.
3. Generating a full video avatar from the sequence of poses.

In this project we work on the second step, and we propose to improve the process of automatic production of SL using state of the art generative methods. We focus on the translation from sign descriptions noted in the Hamburg Sign Language Notation System (HamNoSys) into sequences of poses.

The proposed method comprises three components: (1) a description generator that translates HamNoSys symbols into text descriptions; (2) a Large Language Model (LLM) fine-tuned with instructions to convert these descriptions into sequences of discretized poses; and (3) a Vector Quantized Variational Autoencoder (VQ-VAE) that encodes and decodes these poses. Following this schema, we cast the problem of translating HamNoSys to poses into a sequence-to-sequence task that can be seen as a sequence-to-sequence generation task conditioned by text.

2. Related Work

The tridimensional nature of SL and other complexities such as cotemporality of signs make it harder to represent graphically in the same way as oral languages. There have been many proposals to develop writing systems for SL that capture all its complexities, such as HamNoSys (Hanke, 2004) or SignWriting (Sutton and Frost, 2008), but so far there is no universally agreed standard (Goyal, 2015). In this work, we focus on the HamNoSys notation, which describes a sign as a sequence of symbols denoting hand shape, movement, and other characteristics such as non-manual features (see section 4).

Unlike glossing, which relies on semantic identifiers specific to each language, HamNoSys describes the actual phonology of signs to provide a universal and language agnostic mapping directly to physical gestures. Also, compared to SignWriting and its 2D spatial layouts, the linear and sequential order of HamNoSys is more compatible with computational pipelines.

The first description of the three-step approach to translate oral languages into SL, using pose-based outputs, appears in (Stoll et al., 2018). Research on the second step of the pipeline has generally been approached using glosses as intermediate no-

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 9358–9367
11–16 May 2026.

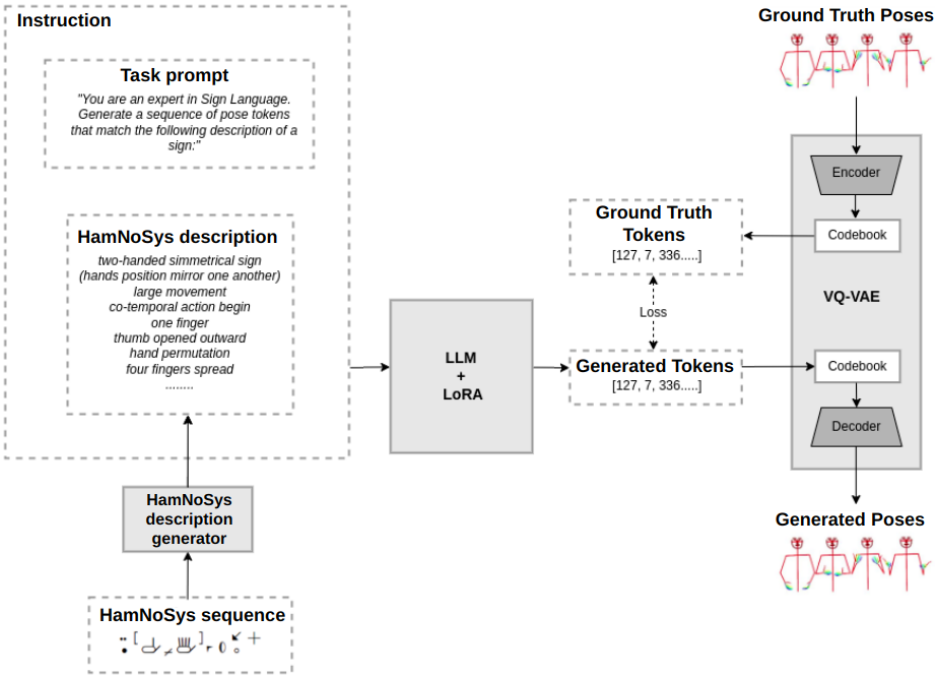


Figure 1: Our approach to poses generation from HamNoSys symbols.

tation[1], and different ways of generating the poses sequence: a simple look-up table (Stoll et al., 2018), GANs and neural machine translation (Stoll et al., 2020), transformer-based approaches (Saunders et al., 2020a,b), and VQ-VAE and discrete diffusion approach (Xie et al., 2024).

Although HamNoSys has previously been used in the context of avatar-based SL generation (Zwitserlood et al., 2005; Bangham et al., 2000), Ham2Pose (Shalev-Arkushin et al., 2023) was the first work to adopt it as an intermediate representation within a pose-based generation framework. It comprises a pipeline of two parts: first they use a transformer encoder to get a representation of the HamNoSys input symbols, then they use a diffusion-based model that uses this representation to generate a sequence of poses.

As HamNoSys is designed to be independent from any particular SL and tries to capture all types of signs, they built a dataset containing examples from four different SL: Polish Sign Language (PJM), German Sign Language (DGS), Greek Sign Language (GSL), and French Sign Language (LSF). The data was collected from the DGS Corpus Hanke (2009), Dicta-Sign Matthes et al. (2012), and the corpus-based dictionary of Polish Sign Languag Łacheta and Rutkowski (2014). In total there are around 5700 videos of signs with their corresponding HamNoSys codification in the dataset.

[1]A simpler notation that represents signs as words from an oral language, loosing much of the information present in fine-grained descriptions of hand configurations and movements.

They proposed a new metric to compare sequences of poses of different length, which belong to a continuous space. The metric, called **nDTW-MJE**, based on the Dynamic Time Warping (DTW) - Mean Joint Error (MJE) metric (Huang et al., 2021). It aligns two sequences of keypoints that may differ in speed or duration, comparing the movement of each keypoint separately. Their main contribution is the handling of missing keypoints (joints that are undetected in the ground-truth or ungenerated by the model): the metric assigns a distance of 0 when a keypoint is absent in the reference and applies a penalty when it is missing in the prediction. This metric is a distance, so lower values are better, with the best possible value being 0.

On top of that, Ham2Pose report their results according to another metric, also based on nDTW-MJE. As small differences between poses might not be that noticeable, it is more important to be able to tell if the prediction is more similar to the reference than to other possible sequences. To do this, they sample 20 random sequences from the reference set, 20 from the prediction set, and the real reference. Then they calculate nDTW-MJE between the prediction and the other 41 sequences, and see the rank of the prediction according to this distance. This is called the **prediction distance rank**, as it is calculated from the prediction point of view. Analogously, they calculate it from the reference point of view, creating the **ground truth distance rank**. In this work we use the same dataset as Ham2Pose and we report the same metrics.

Our approach is inspired by another work, Mo-

tionGPT (Zhang et al., 2024), that does not produce SL signs specifically but general poses from a text description, for example a jump or dance move. They leverage the pretrained capabilities of LLMs and fine-tune them into the problem of predicting a pose sequence from a movement description. The poses are represented by a sequence of codes from a codebook learned using VQ-VAE (Van Den Oord et al., 2017). The objective of a VQ-VAE (Vector Quantized Variational Autoencoder) is to build an encoder and decoder that obtain a discrete latent representation of a continuous space, such as sets of pose keypoints. This discrete space consists of a learned *codebook* of K vectors in $\mathbb{R}^d$, where d is the dimension of each codebook vector. During encoding, each continuous latent vector is replaced by its nearest codebook vector, and during decoding, these discrete vectors are used to reconstruct the original data.

3. Overview of our Approach

Ham2Pose trained a transformer model to predict a pose sequence from HamNoSys symbols. Our hypothesis is that if we could start from a pretrained LLM instead of training a transformer from scratch, like MotionGPT does, we would leverage the language understanding and motion patterns that may already be present in the LLM to improve the prediction results. The problem is that LLMs are trained using oral languages such as English, no SL is significantly present in the training data, and probably no or very little HamNoSys information is present. Because of this, we experimented with different ways of translating HamNoSys symbols into a textual representation that could actually be used by the LLM, and then fine-tune the model to try to predict our pose sequence.

Fig. 1 shows an overview of our approach, which can be divided in the following steps:

1. Generate a text description from a HamNoSys sequence. For example, given , the dictionary transcription generates this description:

   one finger

   extended finger upward and away from the body

   palm down

   co-temporal action begin

   straight movement down (parallel to body)

   finger movements resulting from handshape change | change of handshape

   and orientation

   extended finger away from the body

   co-temporal action end

2. Generate a sequence of pose codes from the HamNoSys description. In this step, the fine-tuned LLM uses the description and generates this code sequence:

   [401, 464, 409, 170, 491, 170, 491, 409, 497, 497, 497, 225, 497, 33, 141, 401, 401]

3. Reconstruct the pose sequence given the codes. The VQ-VAE decoder module uses the code sequence to generate a reconstructed pose sequence as in Fig. 2.

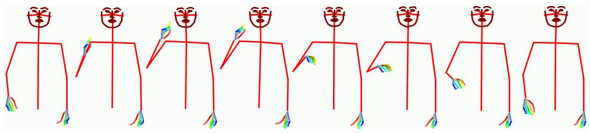


Figure 2: Reconstructed poses.

For step 1, we tried several different ways of encoding HamNoSys into a textual description (see section 4): using the HamNoSys code names, using two dictionaries we created for describing the symbols, or creating a more fluid natural language translation with an LLM.

For steps 2 and 3, note that the poses codes are a way of reducing all possible body and hand configurations, that belong to a continuous space, into a discrete number of symbols, somewhat analogous to capturing all possible phonemes in oral language as discrete units. We approach this in an automatic way by using a VQ-VAE network.

3.1. Dataset

We used the same dataset as (Shalev-Arkushin et al., 2023). It contains a set of videos of signers doing signs in different SLs together with their HamNoSys notations. We used the same keypoints and preprocessed the data in the same way (including masking low-confidence joints, removing leg keypoints, and normalizing pose scale). We split the dataset in roughly 81%-9%-10% for training, dev and test. Table 1 shows the composition of the dataset and our partition and the proportion of each language that is present in each partition.

	PJM	DGS	GSL	LSF
Total signs	2568	1911	889	381
Signers	2	8	2	2
Train	2075	1536	739	307
Dev	231	181	72	33
Test	262	194	78	41

Table 1: Breakdown by language of each dataset split used for training VQ-VAE, reported in number of signs.

3.2. VQ-VAE for Poses

Our VQ-VAE is based on the implementation of (Zhang et al., 2024), also based on (Zhang et al., 2023). The network was trained with the following hyperparameters:

- Encoder input/Decoder output size: 274 (2D coordinates of the 137 Openpose keypoints)
- Codebook size: 512x512
- Batch size: 256
- Optimizer: AdamW with $[\beta_1, \beta_2] = [0.9, 0.99]$
- EMA ($\gamma = 0.99$) and code reset
- 1000 warm-up steps with learning rate 2×10^{-4} and update: $\text{lr}^{\text{current}} = \text{lr}^{\text{prev}} \times \frac{(\#\text{iterations}+1)}{(\#\text{step}+1)}$.
- After warm-up, 50000 iterations without changing the learning rate

Besides these fixed parameters, we experimented varying minimum sequence length (32 or 64) and downsampling rate (1, 2 or 4). We found that the best results were always obtained with a downsampling rate of 2. The best reconstruction loss was found using seq length 32, while the best nDTW-MJE was for length 64. However, as using seq length 64 covered only about 40% of the training corpus, and a lot of examples would be discarded, we decided to go with the model trained with seq length 32, covering 94.5% of the corpus. This VQ-VAE has reconstruction loss of 0.00168 and nDTW-MJE of 7.99 measured on the test set.

4. HamNoSys Descriptions

The structure of a HamNoSys sequence can be divided in four blocks: symmetry operator, non-manual features, initial hand configuration, and movement. Only the last two are mandatory. Fig. 3 shows an example of a sign in HamNoSys format with three blocks (the optional non-manual features are not used in this sign):

- Symmetry operator (¨): it is a two-handed sign that is performed symmetrically with respect to the vertical axis.
- Initial hand configuration: the right hand is open (∩), oriented upwards with a 45º inclination (◟), and with the palm facing down (⌣). The left hand is located symmetrically because of the previous symmetry operator.
- Movement: both hands start the movement in contact (ˣ) and then slide diagonally downwards (ˬ).

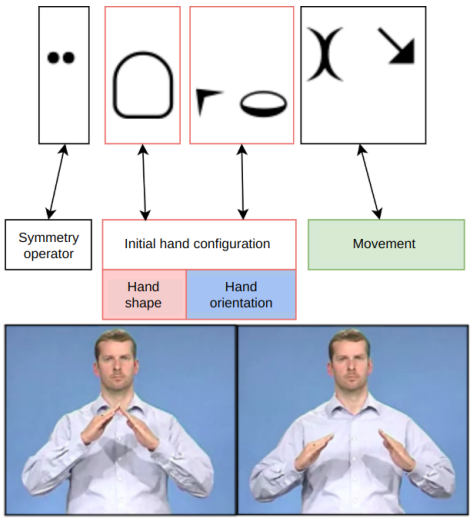


Figure 3: HamNoSys sequence with its structure and snapshots of the real sign representing the movement. (Image based on Shalev-Arkushin et al. (2023))

Implementing a complete description of all possible HamNoSys sequences is a challenging task as there are more than 200 symbols, with multiple combinations and possible interactions between them. The complexity and abstraction of symbols, which represent hand configurations or specific movements, are difficult to summarize without losing important details. There is also interdependence in some subsets of symbols: some only acquire meaning when they are combined with others, so it is very difficult to explain in isolation when trying to generate a unique description, e.g. ⧵ is used to generate intermediate positions between two other symbols. There are also numeric symbols that refer to different fingers affecting other configurations: ☝ represents the extended index finger, but ☝^{5} represents the extended little finger. The ✕ symbol by default represents cross hands but can be combined with the symbols of other body parts such as arms, wrists, or fingers indicating that those are crossed. Although there are some didactic materials and examples online, they are generally insufficient to cover all possible cases and disambiguate the hard ones.

4.1. Dictionaries

We created a dictionary of descriptions of the HamNoSys symbols. These descriptions were done by carefully reviewing the HamNoSys documentation (Hanke, 2004), a HamNoSys 4.0 user guide (Smith, 2013), TeX package[2], and examples

[2] https://github.com/DGS-Korpus/HamNoSys4TeX

available[3], trying to find all the possible meanings for each symbol. This dictionary of HamNoSys meanings and descriptions is a result of this project and is publicly available[4].

Some symbols have concise and clearly defined meanings, such as body parts. Others need some clarifications or additional comments regarding position or movement. However, others are much more complex and might have different meanings depending on context, such as location markers, and we need to give all the alternative meanings.

In a second stage, we decided to refine these definitions so as to make them clearer. For symbols that vary their meaning according to context to the point of being contradictory, we tried to reduce the meaning to its bare form having the common semantic sense that unifies all senses, in order not to introduce noise. This is not possible in all cases, e.g. in cases like *"curved movement towards body or curved movement to the left"* that could not be reduced and were kept as a disjunction. We also simplified some of the symbols for alternatives and blocks using appropriate unicode symbols like |, [or],

Table 2 shows examples of HamNoSys symbols with our simple descriptions and our refined descriptions.

Symbol	Name	Simple description	Refined description
	hameyebrows	eyebrows	
	hamear	ear	
	hampinchone-two	closed pinch between thumb and index - rest of the fingers in fist form	
	hamlrbeside	left | left side of | right | right side of	side
	hamcross	crossing hands (also for arms - wrists and fingers)	crossing
	hammetaalt	alternatives begin	|
[	hamparbegin	co-temporal action begin	co-temporal action [
]	hamparend	co-temporal action end	]
	hamsymmlr	two-handed symmetrical sign (hands position mirror one another)	two-handed symmetrical sign - hands position mirror one another
	hamextfin-gerur	curved movement towards body | curved movement to the left	curved movement towards body or curved movement to the left

Table 2: Examples of HamNoSys symbols with their original name and our simple and refined descriptions. When there is no refined description, the simple one is used for both.

4.2. Descriptions of sequences

Using this dictionary, we designed five ways to represent a HamNoSys sequence of symbols as plain text. The first two are baseline representations that just use the corresponding names for the HamNoSys symbols separated by commas. All HamNoSys names use a “ham” prefix, so we created two variants of this representation: with or without the prefix.

The third way of representing a HamNoSys sequence is a direct lookup of each symbol in our dictionary, concatenating their simple descriptions (meanings), one per line. The fourth way is similar, but using the refined descriptions, which generates a sort of semi-formal language for describing co-temporal or sequential actions.

The last representation method is trying to create a compact description in natural language of the HamNoSys sequence. In this method, the sequence is first encoded using the refined descriptions, then GPT-3.5-turbo (Brown et al., 2020) is told to rewrite it in a coherent way using a prompt composed of four parts:

1. A detailed description of the blocks and sub-blocks present in the HamNoSys sequence, such as ”symmetry block”, ”initial configuration block”, and ”actions block” (see section 4).
2. An explanation of the task, telling the system to rewrite the sequence in a coherent paragraph in natural language.
3. An one-shot example of HamNoSys with its expected natural language description.
4. The refined description encoding of the HamNoSys sequence, so that the model should complete the natural language description after it.

Table 3 shows the representation of the HamNoSys sequence according to our five description methods.

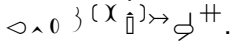
5. Translation into Poses with LLM

The main objective of our experiments is to transform HamNoSys symbols, which describe a sign representing a word or concept in a sign language, into a sequence of poses. In order to do this, we fine-tuned the Mistral 7B model (Jiang et al., 2023) with instructions to generate a sequence of poses from our codebook given a HamNoSys description. The fine-tuning was done using QLoRA (Dettmers et al., 2023). We did several rounds of experiments

[3] https://vhg.cmp.uea.ac.uk/tech/hamnosys/Introduccion%20a%20HamNosys_castellano.pdf

[4] https://github.com/santiagomc/sl-poses-from-hamnosys-nl

Method	Generated description
HamNoSys names	hampinch12, hamextfingeru, hampalml, hamcheek, hamseqbegin, hamtouch, hamfingertip, hamseqend, hamreplace, hamfinger2, hamthumbopenmod, hamrepeatfromstartseveral
HamNoSys names without prefix	pinch12, extfingeru, palml, cheek, seqbegin, touch, fingertip, seqend, replace, finger2, thumbopenmod, repeatfromstartseveral
Simple descriptions	closed pinch between thumb and index - rest of the fingers in fist form extended finger up (parallel to body) palm left cheek sequential action begin contact with fingertip sequential action end finger movements resulting from handshape change | change of handshape and orientation one finger thumb extended repeated a couple of times
Refined descriptions	closed pinch between thumb and index and rest of the fingers in fist form extended finger up - parallel to body palm left cheek sequential action (contact with fingertip) finger movements resulting from handshape change or a change of handshape and orientation one finger thumb extended repeated a couple of times
Natural language description	A closed pinch with the thumb and index finger, other fingers in a fist, pointing upwards parallel to the body, palm facing left, making contact with the cheek, followed by sequential actions of touching fingertips, then individual finger movements and repetitions.

Table 3: Generated description according to description method for the sequence  .

varying different parameters and ways of enconding and decoding the data.

5.1. Hyperparameter search

We used the `Mistral-7B-Instruct-v0.2` model and fine-tuned it with the training samples in the following format:

```
<s>[INST] You are an expert in
Sign Language. Generate a sequence
of pose tokens that match the
following description of a sign:
{HamNoSys description} [/INST]
{Sequence of human poses} </s>
```

These hyperparameters were fixed during all the experiments:

- Number of steps: 22000, checkpoints every 2000 steps
- Batch size: 4
- Optimizer: Paged Adamw 32bit
- Learning rate/Weight Decay: 0.0002/0.0001
- LoRA Alpha/Dropout: 16/0.1

We made different rounds of experiments varying different features and aspects of the training process. As the number of possible configurations was too large, we decided to tune only one aspect at a time in this order:

Inference temperature: We tried different temperatures from 0.3 to 0.9 to see their impact in the decoding process. This experiment used the simple dictionary descriptions. We used the nDTW-MJE metric to compare the results, finding that lower temperatures yielded worse results mainly on the first training steps, but all stabilized in the end. However, the best results were achieved by temperature 0.9, so we used that value in the rest of the experiments.

LoRA range: The first experiments used a LoRA range of 64, and we wanted to see if using different ranges could achieve better results, so we tried with a smaller (32) and a larger (128) range. We could not get improvements in the best result according to nDTW-MJE, and we kept our range of 64 for the rest of the experiments.

Codebook size: We initially used a codebook size of 512 for the VQ-VAE network. Knowing that a larger codebook size could help the reconstruction process, we tried with a larger codebook of size 1024. This had indeed a slight improvement of about 1% (reduction in the nDTW-MJE metric of 1%), but on the other hand it made the whole process much slower, and the gains obtained in this way were not enough to justify its use, so the rest of the experiments were done using the 512 symbols codebook.

Output format repetitions: In our preliminary experiments, we noticed that the model was generating sequences with too many repetitions of the same pose codes. This could happen because in the training data it is very common that some pose sequences have long streaks of the same pose, as when a sign takes a long time in the same or similar hand configurations, compared to the rest of the movements. It could also be an artifact of the VQ-VAE quantization process, because similar consecutive poses are mapped to the same discrete code, generating sequences with repeated codes. In any case, this was harder for the model to learn, and it tended to generate too long sequences of

repeating poses. Because of this, we decided to try different approaches to reduce the number of repetitions. If a subsequence of the same pose was repeated n times, we might reduce n to: 1, $min(n, 6)$, $log_2(n)$, or $\sqrt{n}$ repetitions.

Fig. 4 shows the results of this experiment. The reduction of pose repetitions is only applied to the training data, the test data still has as many repetitions as necessary, however it is clear that the reduction methods have always better results. The best values of nDTW-MJE were achieved by log and square root reductions, and the best one was 11.91 on step 14000 using square root, so we used this technique in the rest of the experiments.

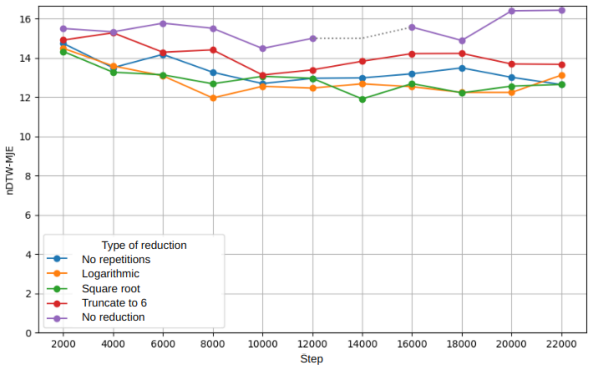


Figure 4: nDTW-MJE during training calculated over the development data varying the maximum number of pose repetitions in output sequences seen during training.

Input format descriptions: We tried with the five different ways of transforming HamNoSys symbols into textual description we described in Section 4: HamNoSys names, HamNoSys names without prefix, simple dictionary description, refined dictionary description, and natural language description.

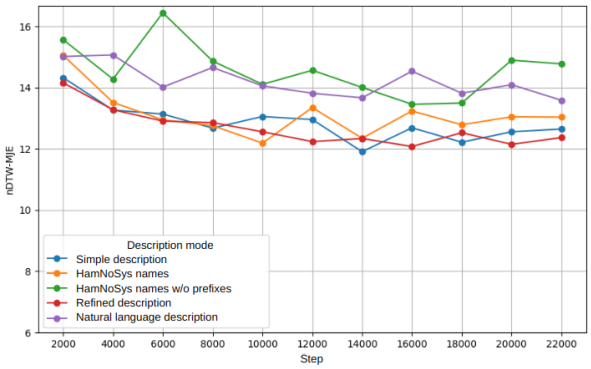


Figure 5: nDTW-MJE during training calculated over the development data varying the method for describing the HamNoSys symbols in text format.

The simple and refined dictionary descriptions are the ones that obtain the best nDTW-MJE results. The refined descriptions show a more stable performance throughout all the training, but the simple ones have the lowest nDTW-MJE value. Using an LLM to generate natural language descriptions, on the other hand, shows the worst results, similar to using the names of the HamNoSys symbols without prefix. It is interesting that just using HamNoSys names, including the prefix, shows a very competitive result, which might indicate that the pretrained LLM might have already seen some HamNoSys notation during its training. The low performance observed when using natural language descriptions generated by an LLM may be related to limitations in prompt design and in the description generation process. Due to the scope of this work, alternative prompting strategies (e.g., more precise instructions, few-shot examples, or fine-tuning) were not explored.

5.2. Results

Table 4 shows the results for our best model on the dev and test sets. This model is the one that uses temperature 0.9, LoRA range 64, codebook size 512, square root repetitions on the output, and simple dictionary descriptions on the input.

As this is a sequence-to-sequence task, we compared to a simpler sequence-to-sequence method as baseline. We trained an OpenNMT (Klein et al., 2017) model that uses the descriptions as input and predicts the pose codebooks as output. OpenNMT is used in its default configuration: an LSTM encoder-decoder model with simple attention mechanism. Our system outperforms this OpenNMT baseline on every metric, but not by a large margin. It is interesting to see that a much simpler method can also have very competitive results, and since we did not do a hyperparameter search on this baseline method, it would be interesting to see if even better results can be obtained in this way.

The table also shows a comparison to the original Ham2Pose results presented in Shalev-Arkushin et al. (2023), and in this case our method seems to outperform this previous work on every metric. But note that the results are not completely comparable because the actual data split used in Ham2Pose could not be found, and the published scripts regenerate the splits on every run.

Table 5 shows a breakdown of methods performance by language, comparing our LLM and the Ham2Pose results. Our method improves especially on PJM and DGS, the two languages with most examples in the dataset, while underperforming for LSF, which is the smallest language in the sample.

We also manually analyzed the generated poses to identify general errors, and try to detect which

Split	Model	nDTW-MJE↓	Prediction			Ground truth		
			Rank1↑	Rank5↑	Rank10↑	Rank1↑	Rank5↑	Rank10↑
Dev	LLM	**11.91**	**0.10**	**0.40**	**0.61**	**0.26**	**0.57**	**0.75**
	OpenNMT	12.87	**0.10**	0.38	0.58	0.24	0.56	0.73
Test	LLM	**12.38**	**0.09**	**0.41**	**0.60**	0.25	**0.58**	**0.74**
	OpenNMT	12.77	**0.09**	0.39	0.58	**0.26**	0.56	0.72
	Ham2Pose	-	*0.08*	*0.20*	*0.35*	*0.21*	*0.44*	*0.56*

Table 4: nDTW-MJE and Distance Ranks on the dev and test sets. We also show the results of Ham2Pose for comparison, although due to differences in the dataset split they are not completely comparable.

Reference	Lang	Rank 1	Rank 5	Rank 10
Prediction	PJM	0.03 / **0.14**	0.15 / **0.58**	0.27 / **0.77**
	DGS	**0.14** / 0.07	0.23 / **0.32**	0.39 / **0.55**
	GSL	0.00 / 0.00	0.10 / **0.18**	0.25 / **0.38**
	LSF	**0.22** / 0.08	**0.60** / 0.18	**0.80** / 0.21
Ground truth	PJM	0.08 / **0.19**	0.25 / **0.52**	0.41 / **0.68**
	DGS	0.26 / **0.29**	0.49 / **0.65**	0.58 / **0.81**
	GSL	0.27 / **0.37**	**0.67** / 0.64	**0.80** / 0.76
	LSF	**0.70** / 0.13	**0.95** / 0.47	**0.97** / 0.63

Table 5: Performance breakdown by language. Each cell contains results for our LLM method compared to Ham2Pose (Ham2Pose/LLM).

of the sign features were more easily reproduced by our method and which ones were harder. First we noticed that the length of our generated poses was generally shorter than the original ones, which could happen due to our reduction in pose repetitions during training. More research is needed to try other ways of reducing repetitions that have less impact on the duration of the sign, or finding a better way to avoid over-repetition during decoding.

About the types of features our method is able to reproduce from the original signs, we found that there is a natural progression according to the complexity and the number of possible configurations in each level: The features most easily captured by our model are the number of hands involved and symmetry (information provided by the HamNoSys symmetry block), distinguishing one-handed from two-handed signs and reproducing the specified symmetry. There was also a good accuracy in reproducing arm movements, trajectories and repetitions. Hand configuration is more challenging, out of this block of characteristics, hand location seems to be the most accurate. However, the shape and orientation of hands had more heterogeneous results, while finger configuration and inclination is the feature that is the most inaccurate. This seems to correspond to the complexity of the problem, as the number of possible configurations in hands and fingers is much larger than in arms or body positions.

Note that there are also two possible sources of errors in our process. The first one is the reconstruction error: due to the VQ-VAE process there

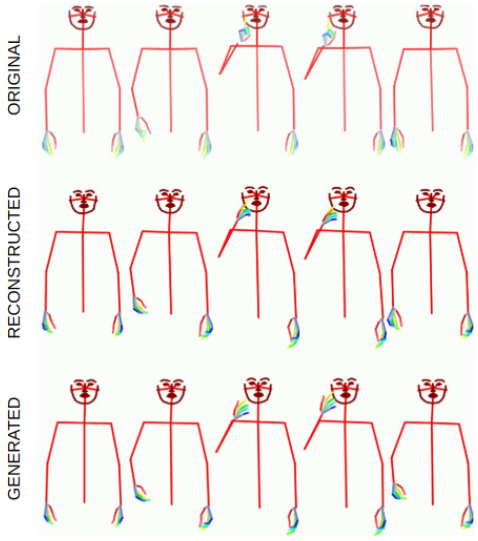


Figure 6: Example of a pose sequence generated with our system. We show the gold standard one, the version we could reconstruct after enconding and decoding with our VQ-VAE, and the version that our LLM generated from HamNoSys.

are some signs that already lose some precision when encoding and decoding them. On top of this, for the ones that can be accurately reconstructed using the VQ-VAE codes, the LLM is a second source of error in that it generally predicts a code sequence different from the expected one. See for example Fig. 6, where the reconstructed poses (second row) are already different from the original

sequence, and then the sequence generated by the LLM (third row) tries its best to reproduce the reconstructed sequence but adding a little more errors.

6. Conclusions

We presented an experiment on translating from a sign written in HamNoSys notation into a sequence of poses that depict the sign movement. Our approach uses an LLM to translate a text description of the HamNoSys sequence into a sequence of pose codes from a codebook created with a VQ-VAE network, representing a discrete latent space of the poses. The hypothesis is that we can leverage the LLM pretraining knowledge about natural language to improve the results.

The experimental results show that this approach has a performance comparable to previous works and even improves on some metrics, validating the proposed methodology.

There are still many challenges about the precision of the generated poses, especially the hand configuration. Future research can focus on expanding the dataset and using more accurate pose estimation methods. We would also like to explore alternative architectures and tune the pose discretization methods for better representation. For example, we could use multiple VQ-VAE codebooks, with some dedicated exclusively to hand representations, and adjust the nDTW-MJE distance to prioritize hand keypoint fidelity.

As part of this project, we also compiled a dictionary that maps HamNoSys symbols to approximate natural language descriptions, contributing an additional resource to its documentation and opening opportunities to use this information on other applications that use human movement generation, like gesture synthesis.

Another limitation is that this research focuses on translating only the representation of one sign into poses, but translating a full SL sentence will involve many signs and also the interaction between their movements. Further research is needed to take these complexities into account.

7. Bibliographical References

Anne E Baker. 2016. Sign languages as natural languages. In *The linguistics of sign languages: An introduction*, pages 1–24. John Benjamins Publishing Company.

J. Andrew Bangham, Stephen Cox, Ralph Elliott, John R. W. Glauert, Ian Marshall, Siniša Rankov, and Mark Wells. 2000. Virtual signing: capture, animation, storage and transmission-an overview of the ViSiCAST project.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. *Advances in neural information processing systems*, 33:1877–1901.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in neural information processing systems*, 36:10088–10115.

L Goyal. 2015. Review and Comparison of Writing Notations of Sign Language. *International Journal of Engineering Sciences (IJoES)*.

Thomas Hanke. 2004. HamNoSys–representing sign language data in language resources and language processing contexts. In *sign-lang@ LREC 2004*, pages 1–6. European Language Resources Association (ELRA).

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*, 1(2):3.

Wencan Huang, Wenwen Pan, Zhou Zhao, and Qi Tian. 2021. Towards fast and high-quality sign language production. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3172–3181.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *ArXiv*, abs/2310.06825.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*.

Ben Saunders, Necati Cihan Camgöz, and R. Bowden. 2020a. Everybody Sign Now: Translating Spoken Language to Photo Realistic Sign Language Video. *ArXiv*, abs/2011.09846.

Ben Saunders, Necati Cihan Camgöz, and R. Bowden. 2020b. Progressive Transformers for End-to-End Sign Language Production. *ArXiv*, abs/2004.14874.

Rotem Shalev-Arkushin, Amit Moryossef, and Ohad Fried. 2023. Ham2pose: Animating sign language notation into pose sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21046–21056.

Robert Smith. 2013. HamNoSys 4.0 User guide.

Stephanie Stoll, Necati Cihan Camgoz, Simon Hadfield, and R. Bowden. 2020. Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks. *International Journal of Computer Vision*, 128:891 – 908.

Stephanie Stoll, Necati Cihan Camgöz, Simon Hadfield, and Richard Bowden. 2018. Sign language production using neural machine translation and generative adversarial networks. In *Proceedings of the 29th British Machine Vision Conference (BMVC 2018)*. British Machine Vision Association.

Valerie Sutton and Adam Frost. 2008. SignWriting: sign languages are written languages. *Center for Sutton Movement Writing*.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Pan Xie, Qipeng Zhang, Peng Taiying, Hao Tang, Yao Du, and Zexian Li. 2024. G2P-DDM: Generating sign pose sequence from gloss sequence with discrete diffusion model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 6234–6242.

J Zhang, Y Zhang, X Cun, S Huang, Y Zhang, H Zhao, H Lu, and X Shen. 2023. T2M-GPT: Generating Human Motion from Textual Descriptions with Discrete Representations. *arXiv preprint arXiv:2301.06052*.

Yaqi Zhang, Di Huang, Bin Liu, Shixiang Tang, Yan Lu, Lu Chen, Lei Bai, Qi Chu, Nenghai Yu, and Wanli Ouyang. 2024. MotionGPT: Finetuned LLMs are General-Purpose Motion Generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7368–7376.

Inge Zwitserlood, Margriet Verlinden, Johan Ros, and Sanny van der Schoot. 2005. Synthetic signing for the deaf: eSIGN.

8. Language Resource References

Thomas Hanke. 2009. DGS corpus project - Development of a corpus based electronic dictionary German Sign Language / German.

Silke Matthes, Thomas Hanke, Anja Regen, Jakob Storz, Satu Worseck, Eleni Efthimiou, Athanasia-Lida Dimou, Annelies Braffort, John R. W. Glauert, and Éva Sáfár. 2012. Dicta-Sign – Building a Multilingual Sign Language Corpus.

Joanna Łacheta and Paweł Rutkowski. 2014. A Corpus-based Dictionary of Polish Sign Language (PJM).