



FACULTAD DE
INGENIERÍA
UDELAR



UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Generación de poses de lengua de señas a partir de descripciones en lenguaje natural de símbolos HamNoSys

Santiago Máximo Caram

Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Marzo de 2025



FACULTAD DE
INGENIERÍA
UDELAR



UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Generación de poses de lengua de señas a partir de descripciones en lenguaje natural de símbolos HamNoSys

Santiago Máximo Caram

Tesis de Maestría presentada al Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ciencia de Datos y Aprendizaje Automático.

Director:

Luis Chiruzzo

Director académico:

Aiala Rosá

Montevideo – Uruguay

Marzo de 2025

Máximo Caram, Santiago

Generación de poses de lengua de señas a partir de descripciones en lenguaje natural de símbolos HamNoSys / Santiago Máximo Caram. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2025.

xv, 122 p.: il.; 29, 7cm.

Director:

Luis Chiruzzo

Director académico:

Aiala Rosá

Tesis de Maestría – Universidad de la República, Programa en Ciencia de Datos y Aprendizaje Automático, 2025.

Referencias bibliográficas: p. 90 – 96.

1. Producción de Lengua de Señas, 2. HamNoSys, 3. LLM, 4. VQ-VAE, 5. Instruction Tuning. I. Chiruzzo, Luis, . II. Universidad de la República, Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Guillermo Moncecchi

Gregory Randall

Horacio Saggion

Montevideo – Uruguay
Marzo de 2025

RESUMEN

Las lenguas de señas, utilizadas por millones de personas, son lenguajes naturales con una gramática y un léxico propios, independientes de los lenguajes orales. A pesar de los avances de los últimos años, la producción automática de lengua de señas sigue siendo un desafío.

Esta tesis investiga inicialmente el uso de modelos generativos para mejorar la producción de lengua de señas, para luego centrarse específicamente en la generación de secuencias de poses a partir de la notación The Hamburg Sign Language Notation System (HamNoSys).

El método propuesto consta de tres componentes: (1) un generador de descripciones que traduce símbolos HamNoSys a lenguaje natural, (2) un Large Language Model (LLM) ajustado con *Instruction Tuning* para convertir estas descripciones en secuencias discretas de poses y (3) una red Vector Quantized Variational Autoencoder (VQ-VAE) para codificar y decodificar las poses. Esta combinación transforma la traducción de *HamNoSys a poses* en una tarea de generación condicionada por texto.

Los resultados experimentales muestran que este enfoque logra un desempeño comparable a trabajos previos, validando así la metodología propuesta. No obstante, persisten desafíos en la precisión de las poses, especialmente en las configuraciones de las manos. Se proponen futuras líneas de investigación, como la exploración de arquitecturas alternativas, y el ajuste de las técnicas de discretización de poses.

Además, esta investigación contribuye a la creación de diccionarios en lenguaje natural de símbolos HamNoSys, mejorando la accesibilidad, y abre oportunidades para el desarrollo de aplicaciones multimodales en la generación de movimiento humano, como la síntesis de gestos.

Palabras claves:

Producción de Lengua de Señas, HamNoSys, LLM, VQ-VAE, Instruction Tuning.

Lista de figuras

1.1	Etapas de la producción de lengua de señas	3
2.1	Parametrización de LoRA, solo se entrenan A y B (Hu et al. 2021).	10
2.2	Arquitectura de ControlNet (L. Zhang et al. 2023).	11
2.3	Ejemplos de ControlNets sobre Stable Diffusion (L. Zhang et al. 2023).	12
2.4	Ejemplo de glosa en ASL (Gill, 2005).	14
2.5	Estructura de HamNoSys (Skobov y Lepage, 2020).	15
2.6	<i>Give</i> representado en SignWriting (Cripps, 2008).	15
2.7	Lengua de Señas Alemana para <i>Haus</i> (Shalev-Arkushin et al. 2023).	16
2.8	Arquitectura de Ham2Pose (Shalev-Arkushin et al. 2023). . .	19
2.9	<i>Emparejamiento</i> de dos series temporales (Salvador y Chan, 2004). 20	
2.10	Diagrama del proceso evaluación utilizando la métrica <i>Distance Ranks</i> para un elemento del conjunto de test.	22
2.11	Método de MotionGPT (Y. Zhang et al. 2023).	25
2.12	Arquitectura de <i>Text2Video-Zero</i> (Khachatryan et al. 2023). .	27
2.13	Método de <i>Follow Your Pose</i> (Ma et al. 2024).	28
2.14	Experimento con TemporalKit para imitar seña de un video con otro personaje.	29
3.1	Método propuesto	33
3.2	Secuencia de poses generada.	35
3.3	Secuencia de <i>frames</i> del video original.	35
4.1	Esquema de VQ-VAE (J. Zhang et al. 2023).	37
4.2	Arquitectura de la VQ-VAE de poses (J. Zhang et al. 2023). .	41

4.3	Evolución de $\mathcal{L}_{commit}$, Perplexity (PPL), $\mathcal{L}_{re}$ y $nDTW-MJE$ durante el entrenamiento de la red VQ-VAE.	44
5.1	Estructura de una secuencia HamNoSys (Skobov y Lepage, 2020).	46
5.2	Secuencia HamNoSys acompañada de la seña real.	47
5.3	Ejemplo de seña con dos manos que incluye excepciones.	47
5.4	Formas básicas de la mano.	48
5.5	Forma base combinada con inclinaciones de los dedos.	48
5.6	Forma base combinada con posiciones del dedo pulgar.	48
5.7	Ubicaciones con respecto a la cabeza.	49
5.8	Sub-bloque de <i>localización</i> : $\cap^{(\chi \uparrow)}$	49
5.9	Bloque de <i>movimiento</i> : $\downarrow^{(\uparrow \rightarrow)}$	50
5.10	Posición intermedia entre dos formas de la mano.	51
5.11	Efectos de usar especificaciones sobre dedos de la mano.	51
5.12	<i>Frames</i> del video original para la secuencia $\prec \wedge 0 \}^{(\chi \uparrow)} \rightarrow \downarrow^{++}$	55
5.13	<i>Frames</i> del video original para la secuencia $\mathfrak{M} \setminus \mathfrak{M} \mathfrak{r} \setminus \mathfrak{r} \ominus \mathfrak{O}^{\square} \}^{(\swarrow \times)}$	64
6.1	Evolución de $nDTW-MJE$ para diferentes valores de temperatura	71
6.2	Evolución de $nDTW-MJE$ para los diferentes técnicas de reducción de repeticiones	72
6.3	Evolución de $nDTW-MJE$ para las diferentes modalidades de descripción	73
6.4	Evolución de $nDTW-MJE$ para los diferentes valores del rango de LoRA.	74
6.5	Evolución de $nDTW-MJE$ para diferentes tamaños del <i>codebook</i> de la red VQVAE.	75
6.6	Evolución de $nDTW-MJE$: LLM vs OpenNMT	76
6.7	Ejemplo de una secuencia de poses generada de forma precisa.	80
6.8	Ejemplo de una secuencia de poses con el movimiento de brazos correcto pero con errores en las configuraciones de las manos.	80
6.9	Ejemplo de una secuencia de poses con la localización de la mano imprecisa. La mano derecha se lleva al pecho en lugar del hombro derecho.	80
6.10	Ejemplo donde las imprecisiones en la configuración de las manos de la secuencia generada, ya están presentes en la codificación de las poses	81

6.11	Ejemplo donde la secuencia generada muestra errores en la configuración de la mano, a pesar de que las poses son codificadas correctamente	82
6.12	Ejemplo donde la secuencia generada muestra errores al representar la mano izquierda en reposo.	82

Lista de tablas

4.1	Composición por lengua para cada partición del <i>dataset</i> utilizado para entrenar la red VQ-VAE.	41
4.2	K-Means vs VQ-VAE	42
4.3	$nDTW-MJE$ para diferentes <i>tasas de downsampling</i>	43
4.4	Resultados para modelos entrenados con <i>tasa de downsampling</i> igual a 2.	43
4.5	Métricas de validación y testeo, para modelos entrenados con <i>tasa de downsampling</i> igual a 2.	43
5.1	Ejemplos de fuente HamNoSys para partes del cuerpo	52
5.2	Descripciones según modalidad para la secuencia $\prec \wedge 0 \succ (\chi \uparrow) \rightarrow \Downarrow \#$	53
5.3	Ejemplo utilizando modalidad <i>secuencia HamNoSys</i>	55
5.4	Ejemplo utilizando modalidad <i>secuencia HamNoSys sin prefijo</i>	56
5.5	Ejemplo utilizando modalidad <i>secuencia con descripciones</i>	56
5.6	Descripciones simples.	57
5.7	Descripciones con aclaraciones.	57
5.8	Descripciones con múltiples significados.	57
5.9	Descripciones de símbolos de signos de puntuación.	58
5.10	Ejemplo utilizando modalidad <i>secuencia con descripciones refinadas</i>	58
5.11	Descripciones con múltiples significados refinadas.	59
5.12	Descripciones con múltiples significados, conservadas.	59
5.13	Descripciones refinadas con signos de puntuación.	60
5.14	Descripciones refinadas sin el uso de determinados signos.	60
5.15	Ejemplo utilizando modalidad <i>descripción en lenguaje natural</i>	61
5.16	Grupos de símbolos definidos.	63

6.1	Composición por lengua para cada partición del <i>dataset</i> utilizado para ajustar el LLM.	68
6.2	Métricas <i>nDTW-MJE</i> y <i>Distance Ranks</i> para los conjuntos de validación y testeo.	77
6.3	Comparación de <i>Distance Rnks</i> entre Ham2Pose, OpenNMT y LLM (considerando que los conjuntos de datos no son exactamente comparables).	78
6.4	<i>Distance Ranks</i> por lengua de señas (Ham2Pose / LLM). . .	79
1.1	Diccionarios Parte I	98
2.1	Diccionarios Parte II	106
3.1	Diccionarios Parte III	117

Lista de siglas

- ASL** Lengua de Señas Americana [1](#), [14](#), [15](#)
- ASR** Reconocimiento Automático del Habla [2](#)
- BSL** Lengua de Señas Británica [1](#)
- CSLR** Reconocimiento de Lenguas de Señas Continuas [1](#)
- DDPM** Denoising Diffusion Probabilistic Models [10](#)
- DGS** Lengua de Señas Alemana [19](#)
- DTW** Dynamic Time Warping [20](#), [21](#)
- DTW-MJE** Dynamic Time Warping - Mean Joint Error [20](#), [21](#)
- EMA** Exponential Moving Average [13](#), [39](#)
- GAN** Generative Adversarial Network [26](#)
- GPT** Generative Pre-trained Transformer [8](#), [24](#)
- GSL** Lengua de Señas Griega [19](#)
- HamNoSys** The Hamburg Sign Language Notation System [v](#), [vi](#), [vii](#), [13](#), [14](#), [15](#), [16](#), [17](#), [18](#), [46](#)
- LDM** Latent Diffusion Models [10](#)
- LLM** Large Language Model [v](#), [24](#), [25](#)
- LSE** Lengua de Señas Española [14](#)
- LSF** Lengua de Señas Francesa [19](#)
- LoRA** Low Rank Adaptation [6](#)
- MDN** Mixture Density Network [17](#)
- PJM** Lengua de Señas Polaca [19](#)
- PLN** Procesamiento del Lenguaje Natural [9](#)
- PPL** Perplexity [vii](#), [42](#), [43](#), [44](#)
- RLHF** Reinforcement Learning from Human Feedback [9](#)
- RNN** Red Neuronal Recurrente [8](#)
- SLP** Producción de Lengua de Señas [2](#)

SLT Traducción de Lengua de Señas [1](#)

VAE Variational Autoencoder [7](#), [12](#), [24](#)

VQ-VAE Vector Quantized Variational Autoencoder [v](#), [vi](#), [vii](#), [ix](#), [4](#), [7](#), [12](#),
[13](#), [17](#), [24](#), [25](#), [36](#), [37](#), [38](#), [39](#), [41](#), [42](#), [44](#), [68](#), [70](#)

WFD Federación Mundial de Sordos [1](#)

Tabla de contenidos

Lista de figuras	VI
Lista de tablas	IX
Lista de siglas	XII
1 Introducción	1
1.1 Motivación	1
1.2 Objetivo	2
1.3 Aportes	4
1.4 Organización del documento	5
2 Fundamentos teóricos	7
2.1 LLMs y LoRA	8
2.2 Modelos de difusión	10
2.2.1 ControlNet	11
2.3 <i>Autoencoders</i> y VQ-VAE	12
2.4 Notaciones de lenguas de señas	13
2.4.1 Glosa	14
2.4.2 Sistema de Notación de Hamburgo (HamNoSys)	14
2.4.3 SignWriting	15
2.4.4 Comparación de notaciones	16
2.5 Generación de poses de lengua de señas	16
2.6 Ham2Pose	18
2.6.1 Dataset	18
2.6.2 Preprocesamiento de los datos	19
2.6.3 Métrica nDTW-MJE	20
2.6.4 Métrica Distance Ranks	21

2.7	Generación de poses a partir de texto	23
2.7.1	MotionGPT	24
2.8	Generación de avatares de lengua de señas	26
2.9	Modelos de difusión para la generación de video	27
2.9.1	Text2Video-Zero	27
2.9.2	Follow Your Pose	28
2.9.3	TemporalKit	28
2.9.4	Resultados preliminares utilizando modelos de difusión	29
2.10	Conclusión	30
3	Método	32
3.1	Ejemplo	33
4	VQ-VAE de poses	36
4.1	Descripción general de la red VQ-VAE	36
4.2	Optimización	38
4.3	Cuantización	39
4.3.1	EMA	39
4.3.2	Code reset	40
4.4	Arquitectura	40
4.5	Detalles de implementación	40
4.6	Dataset	41
4.7	Resultados	42
4.8	Conclusiones	44
5	Generador de descripciones HamNoSys	45
5.1	Estructura de la notación HamNoSys	46
5.1.1	Operador de simetría	47
5.1.2	Configuración inicial de la mano	47
5.1.3	Movimiento	49
5.2	Complejidades de la notación HamNoSys	50
5.3	Preprocesamiento de los datos	51
5.4	Modalidades de descripciones	53
5.4.1	Secuencia HamNoSys	55
5.4.2	Secuencia HamNoSys sin prefijo	55
5.4.3	Secuencia con descripciones	56
5.4.4	Secuencia con descripciones refinadas	58

5.4.5	Descripción en lenguaje natural	60
6	LLM	66
6.1	Modelo Mistral-7B-Instruct-v0.2	66
6.1.1	Formato de las instrucciones	67
6.2	Instrucciones definidas	67
6.3	<i>Dataset</i>	68
6.4	Entrenamiento	69
6.5	Posprocesamiento de las salidas generadas	69
6.6	Experimentos	70
6.6.1	Elección de temperatura para inferencia	70
6.6.2	Reducción de repeticiones consecutivas	71
6.6.3	Modalidad de descripción	73
6.6.4	Rango de LoRA	74
6.6.5	Tamaño del <i>codebook</i>	75
6.7	Comparación con <i>baseline</i>	76
6.8	Comparación con Ham2Pose	77
6.9	Análisis cualitativo	78
6.10	Conclusiones	82
7	Conclusión	85
7.1	Estado del arte y metodología	85
7.2	Método propuesto	85
7.3	Resultados obtenidos	86
7.4	Limitaciones identificadas	87
7.5	Agenda para investigaciones futuras	88
7.6	Conclusión final	89
	Referencias bibliográficas	90
	Apéndices	97
Apéndice 1	Diccionarios Parte I	98
Apéndice 2	Diccionarios Parte II	106
Apéndice 3	Diccionarios Parte III.	117

Capítulo 1

Introducción

1.1. Motivación

Según la Federación Mundial de Sordos (WFD), hay más de 70 millones de personas sordas en todo el mundo. Más del 80 % de ellas viven en países en desarrollo. Colectivamente, utilizan más de 300 lenguas de señas diferentes (United Nations, [2023](#)).

Las lenguas de señas, que utilizan la modalidad visual-gestual para transmitir significado, son lenguajes naturales con su propia gramática y léxicos, que funcionan de forma independiente a los lenguajes orales. Por ejemplo, la Lengua de Señas Americana (ASL), utilizada predominantemente en los Estados Unidos, y la Lengua de Señas Británica (BSL), empleada mayormente en el Reino Unido, son completamente diferentes, a pesar de que el inglés es el lenguaje oral predominante en ambos países (Shalev-Arkushin et al. [2023](#)).

Durante las últimas tres décadas, la traducción computacional de lengua de señas ha sido un área de investigación continua. El estudio inicial se centró en el reconocimiento de lenguas de señas aisladas. Luego, los avances en el campo llevaron al Reconocimiento de Lenguas de Señas Continuas (CSLR), que requiere tanto la segmentación de un video en sus signos constituyentes como su clasificación. Desde entonces, se introdujo una tarea más desafiante: la Traducción de Lengua de Señas (SLT), donde secuencias continuas de lengua de señas se convierten directamente en oraciones de lenguaje oral. SLT es más desafiante que CSLR debido a las diferencias en gramática y orden entre la lengua de señas y el lenguaje oral (Walsh et al. [2024](#)).

Aunque la investigación sobre la traducción de lenguas de señas a lenguajes

orales ha avanzado rápidamente en los últimos años, la traducción de lenguajes orales a lenguas de señas, también conocida como Producción de Lengua de Señas (SLP), sigue siendo un desafío. Esto se debe en parte a la idea errónea de que las personas sordas se sienten cómodas leyendo el lenguaje transcrito y no requieren traducción a la lengua de señas. Sin embargo, no hay garantía de que alguien cuya primera lengua es una lengua de señas, por ejemplo, BSL, muestre una alta alfabetización en inglés escrito (Shalev-Arkushin et al. 2023).

El *pipeline* para la traducción de lenguajes orales a lengua de señas es un proceso complejo que, en general, no se implementa de manera *end-to-end*, sino que se divide en varias etapas, como se muestra en la Figura 1.1.

La primera etapa consiste en el procesamiento de videos, donde la lengua oral es transcrita a texto. Este proceso ha sido ampliamente estudiado y aplicado en diversos contextos más allá de la traducción a lengua de señas, ya que forma parte del campo del Reconocimiento Automático del Habla (ASR).

Para las etapas subsiguientes, las técnicas más estudiadas han seguido un enfoque estructurado en dos fases (Walsh et al. 2022). Primero, se traduce el texto transcrito a un sistema de notación intermedia, como las glosas o HamNoSys. Luego, se genera una secuencia en lengua de señas a partir de esta representación intermedia. Este último paso suele dividirse en dos subetapas: la conversión de la notación intermedia en una secuencia de poses y el mapeo de dicha secuencia a una representación visual en lengua de señas, generalmente mediante un avatar, es decir, un personaje animado por computadora que reproduce las señas.

Recientemente, los avances en *deep learning*, y en particular en los modelos generativos, como los modelos de difusión y los LLMs, han abierto nuevas oportunidades para mejorar la producción de la lengua de señas, impulsando la accesibilidad y la inclusión. En este contexto, existe un gran potencial para explorar el uso de modelos generativos en la etapa de producción de lengua de señas, un proceso de naturaleza multimodal que involucra tanto la manipulación de representaciones intermedias de la lengua de señas como la generación de imágenes y videos para la producción de poses y avatares.

1.2. Objetivo

El objetivo principal de este trabajo es contribuir al desarrollo del *pipeline* para la producción de lengua de señas mediante un uso novedoso de los méto-

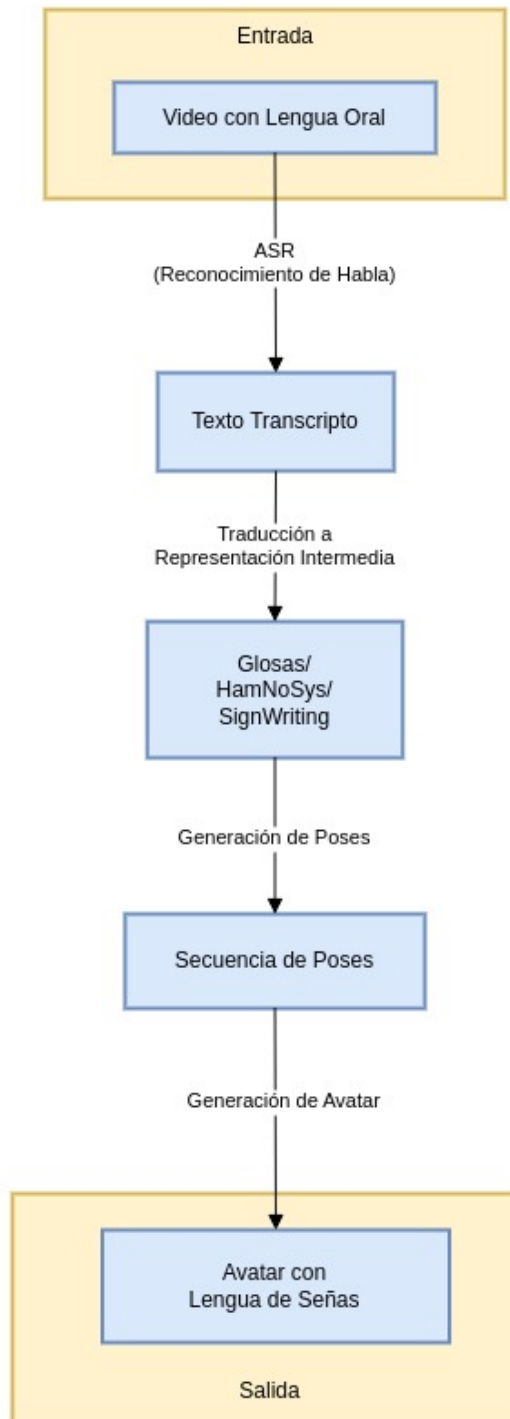


Figura 1.1: Etapas de la producción de lengua de señas

dos generativos, promoviendo así una mayor accesibilidad e inclusión para las personas sordas. Específicamente, se busca: (1) implementar una arquitectura VQ-VAE para la representación eficiente de poses en lengua de señas; (2) crear recursos lingüísticos que permitan traducir símbolos HamNoSys a lenguaje natural; y (3) desarrollar un método basado en LLMs para la transformación de descripciones HamNoSys en secuencias de poses.

1.3. Aportes

Las contribuciones de este trabajo son las siguientes:

- **Análisis del estado del arte en producción de lengua de señas.** Se realiza un estudio exhaustivo de los enfoques existentes para la producción de lengua de señas. Este análisis incluye también la comparación entre las representaciones intermedias más populares, proporcionando un marco de referencia para futuras investigaciones en el área.
- **Un nuevo método de conversión de secuencias HamNoSys a secuencias de poses:** Se propone un enfoque novedoso basado en modelos generativos que permite transformar descripciones en notación HamNoSys en secuencias de poses.
- **Desarrollo de un modelo basado en la arquitectura VQ-VAE.** Se implementa una red VQ-VAE para la codificación y decodificación de poses de lengua de señas. El modelo propuesto facilita la representación discreta de las poses y puede ser reutilizado en contextos de traducción de señas a lenguaje oral y viceversa.
- **Demostración del potencial de los LLMs para la generación de secuencias de poses a partir de descripciones textuales.** Se ilustra cómo estos modelos, entrenados con *Instruction Tuning*, pueden utilizarse en modalidades *texto a imagen* o *texto a video*, abriendo la puerta diversas aplicaciones que requieran la representación de movimientos detallados de las manos, más allá del ámbito de la lengua de señas.
- **Creación de diccionarios con explicaciones en lenguaje natural de los símbolos HamNoSys.** Estos diccionarios pueden contribuir a la accesibilidad, permitiendo, por ejemplo, que personas con discapacidad visual comprendan de manera clara cada símbolo y, además, facilitar la

enseñanza y difusión de la lengua de señas.

1.4. Organización del documento

[Capítulo 2: Fundamentos teóricos](#) expone los fundamentos teóricos y la revisión de la literatura relevante para la investigación. Se examinan distintas notaciones para la lengua de señas y se presentan enfoques para la generación de poses a partir de una representación intermedia, así como su relación con técnicas de generación de movimiento a partir de texto. Además, se revisan metodologías para la creación de avatares de lengua de señas, analizando el potencial de estrategias recientes para la generación de video. Como cierre, se exponen las conclusiones preliminares y las decisiones adoptadas que guiarán el desarrollo de la tesis.

[Capítulo 3: Método](#) describe la arquitectura propuesta para la generación de secuencias de poses en lengua de señas a partir de descripciones en HamNoSys. Se presentan los módulos que componen el sistema y la forma en que interactúan para transformar estas descripciones en secuencias de movimiento.

[Capítulo 4: VQ-VAE de Poses](#) describe la implementación de la red VQ-VAE utilizada para la codificación y decodificación de secuencias de poses de lengua de señas. Comienza ofreciendo una visión general de los componentes de la red, los objetivos de optimización y las estrategias de cuantización empleadas. Luego, se detallan la arquitectura, los hiperparámetros y el conjunto de datos utilizados. El capítulo concluye con la presentación de los experimentos realizados con esta implementación, el análisis de los resultados obtenidos y la exposición de las conclusiones derivadas de este trabajo.

[Capítulo 5: Generador de descripciones HamNoSys](#) describe el proceso de generación de descripciones de secuencias HamNoSys, fundamentales para el entrenamiento del LLM especializado en la generación de poses de lengua de señas. Se detalla la estructura de la notación HamNoSys con ejemplos y visualizaciones, seguido de un análisis sobre los desafíos en la conversión de sus símbolos a lenguaje natural, y de la descripción del preprocesamiento empleado. Finalmente, se presentan las cuatro moda-

lidades implementadas para la generación de descripciones, destacando sus características y diferencias.

Capítulo 6: LLM detalla la implementación del ajuste del LLM para la generación de *tokens* representativos de poses en lengua de señas. Se describe el modelo base utilizado, junto con el formato de las instrucciones definidas para este trabajo. Luego se explica la composición del conjunto de datos, el proceso de ajuste mediante *Instruction Tuning* y Low Rank Adaptation (LoRA), y se introduce el posprocesamiento empleado para extraer secuencias de *tokens* tras cada inferencia. Por último, se presentan los experimentos realizados, se analizan los resultados obtenidos y se exponen las conclusiones del capítulo.

Capítulo 7: Conclusión ofrece un análisis final del trabajo realizado. Se inicia con un resumen de la revisión del estado del arte y la metodología aplicada, seguida de la descripción del enfoque desarrollado. Luego se detallan los hallazgos principales y los desafíos identificados. Finalmente, se plantean posibles líneas de investigación futura y una conclusión general.

Capítulo 2

Fundamentos teóricos

Este capítulo presenta los fundamentos teóricos y la revisión de la literatura relevante para la presente investigación. Se explicará la trayectoria seguida en el análisis de modelos y técnicas relevantes, así como las decisiones tomadas con base en esta revisión.

En primer lugar, se introducen los modelos generativos significativos en el contexto de este trabajo. Se analizan los LLMs y la técnica de adaptación LoRA, así como los modelos de difusión y la arquitectura ControlNet. Además, se presentan los *autoencoders*, en particular las redes Variational Autoencoder (VAE) y VQ-VAE.

Posteriormente, se examinan las principales notaciones utilizadas en la representación de lengua de señas: glosas, HamNoSys y SignWriting. Se comparan sus características con el fin de determinar cuál resulta más adecuada para alimentar el *pipeline* de producción de lengua de señas propuesto en esta tesis.

Seguidamente se revisa el estado del arte de la primera etapa contemplada en esta revisión bibliográfica: la generación de poses de lenguas de señas a partir de una notación intermedia. Como se justificará más adelante, en el presente proyecto se opta por trabajar con la representación HamNoSys. En consecuencia, se analiza en profundidad el proyecto Ham2Pose (2022), pionero en abordar la generación de poses de lengua de señas a partir de HamNoSys. Se describe su arquitectura y se identifican los elementos que serán incorporados en el presente trabajo: su conjunto de datos, su metodología de preprocesamiento de poses, así como las métricas *nDTW-MJE* y *Distance Ranks* que permitirán establecer comparaciones con la implementación desarrollada en esta tesis.

Para explorar la incorporación de modelos generativos recientes en la generación de poses de lengua de señas, se investiga su aplicación en un campo con características similares pero más desarrollado: la generación de movimiento a partir de texto. En este contexto, se profundiza en el proyecto MotionGPT, que demuestra el potencial del uso de LLMs y redes VQ-VAE en la generación de secuencias de poses humanas. Este enfoque servirá como referencia para el método propuesto en este trabajo.

Posteriormente, se aborda la otra etapa del flujo de producción de lengua de señas considerada en esta revisión bibliográfica, la generación de avatares de lengua de señas, revisando la evolución de las técnicas. Con el objetivo de examinar las oportunidades de utilizar modelos de difusión en este campo, se estudia el estado del arte en la generación de vídeos a partir de este tipo de modelos. Se analizan tres proyectos de referencia que serán examinados en esta fase exploratoria inicial de la tesis: Text2Video-Zero (Khachatryan et al., 2023), Follow Your Pose (Ma et al., 2024) y TemporalKit (Rowles, 2023).

Finalmente, se presentan las conclusiones preliminares derivadas de esta revisión y las decisiones metodológicas adoptadas para el desarrollo de esta tesis.

2.1. LLMs y LoRA

Un modelo de lenguaje es una distribución de probabilidad sobre secuencias de palabras o fragmentos de palabras (o más específicamente *tokens*).

Los modelos de lenguaje se pueden clasificar de manera general en dos categorías principales: modelos de lenguaje estadísticos, que utilizan enfoques estadísticos tradicionales, y modelos de lenguaje neuronales, que se basan en arquitecturas de redes neuronales como las Redes Neuronales Recurrentes (RNNs) o Transformers (Vaswani et al. 2017).

Los modelos de lenguaje basados en Transformers, introducidos por primera vez por Google Brain en 2017, representan el estado del arte en la mayoría de las tareas de procesamiento de lenguaje natural. Estos modelos se basan principalmente en el mecanismo de *atención*, que ayuda a manejar de mejor manera las dependencias de larga distancia en los textos, y permite más *paralelización* en comparación a otras arquitecturas neuronales.

La serie de modelos Generative Pre-trained Transformer (GPT) de Ope-

nAI¹ ejemplifica los avances en los modelos de lenguaje. GPT-3, en particular, demostró el potencial de los grandes modelos de lenguaje en diversas aplicaciones, desde la escritura creativa hasta la asistencia en codificación. Estos modelos fueron entrenados de forma autoregresiva, lo que significa que aprenden a predecir el siguiente *token* (palabra o parte de palabra) basándose en los *tokens* anteriores de la secuencia.

Más recientemente, en noviembre de 2022, ChatGPT² mostró mejoras significativas en la calidad de la generación de texto en comparación con sus predecesores, produciendo respuestas similares a las humanas con resultados sorprendentes.

ChatGPT se entrenó utilizando una combinación de Reinforcement Learning from Human Feedback (RLHF) e *Instruction Tuning* (Ouyang et al. 2022). RLHF combina el aprendizaje por refuerzos con retroalimentación humana para alinear las salidas del modelo con las preferencias humanas. *Instruction Tuning* en cambio, implica ajustar el modelo preentrenado utilizando *datasets* que contienen instrucciones humanas y las salidas deseadas, lo que permite al modelo aprender cómo interpretar y seguir instrucciones.

En los últimos tiempos han surgido alternativas de código abierto a ChatGPT, como LLaMA (Touvron, Lavril et al. 2023) o modelos provistos por Mistral AI³, que buscan democratizar el acceso a tecnologías avanzadas de Procesamiento del Lenguaje Natural (PLN). Estos modelos permiten a investigadores y desarrolladores experimentar y construir sobre arquitecturas existentes sin las restricciones de los sistemas propietarios.

Pero a medida que se preentrenan modelos cada vez más grandes, realizar un ajuste (*fine tuning*) completo los mismos, es decir volver a entrenar con todos los parámetros del modelo, se vuelve menos factible.

LoRA (Hu et al. 2021), se presenta como un método eficiente para el ajuste fino de modelos de lenguaje, ya que preserva los pesos preentrenados originales ($W \in \mathbb{R}^{d \times d}$) mientras introduce matrices entrenables de bajo rango ($A \in \mathbb{R}^{r \times d}$, $B \in \mathbb{R}^{d \times r}$), donde $r \ll d$, como se ilustra en la Figura 2.1. Durante el entrenamiento, solo estas matrices de rango reducido son actualizadas, manteniendo congelados los pesos originales. La actualización se expresa como $h = Wx + BA(x)$, donde x representa la entrada y h la salida del modelo.

¹<https://openai.com/>

²<https://chat.openai.com/>

³<https://mistral.ai/>

Esta aproximación reduce significativamente los recursos computacionales necesarios para el ajuste fino mientras mantiene un rendimiento comparable al ajuste tradicional, lo que la hace particularmente valiosa para la adaptación de modelos de gran escala a tareas específicas. Por ejemplo, tomando como referencia GPT-3 de 175 billones de parámetros, LoRA puede reducir la cantidad de parámetros entrenables en 10.000 veces y los requerimientos de GPU en 3 veces.

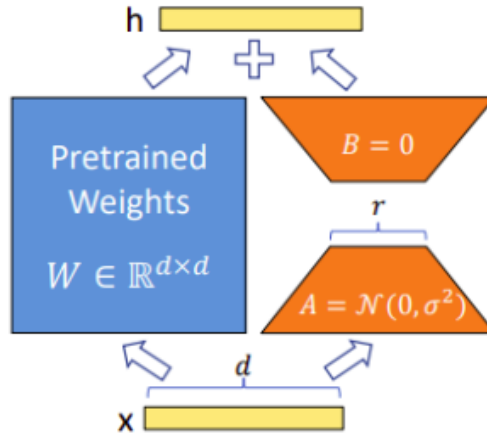


Figura 2.1: Parametrización de LoRA, solo se entrenan A y B (Hu et al. 2021).

2.2. Modelos de difusión

Los modelos de difusión son otra clase de modelos generativos muy utilizados para la generación de imágenes o video. Están inspirados en la termodinámica del no equilibrio. Típicamente definen una cadena de Markov de pasos de difusión para agregar lentamente ruido aleatorio a los datos, y luego aprenden a revertir el proceso de difusión para construir muestras de datos a partir del ruido. Los modelos Denoising Diffusion Probabilistic Models (DDPM) (Ho et al. 2020) fueron los primeros en demostrar que los modelos de difusión son capaces de generar imágenes de alta calidad, superando en ocasiones a otros tipos de modelos generativos. Más recientemente los Latent Diffusion Models (LDMs) (Blattmann et al. 2023) proponen el *denoising* (eliminación de ruido) en un espacio latente, logrando un equilibrio entre efectividad y eficiencia.

Existen diferentes familias de modelos de difusión. Stable Diffusion (Rombach et al. 2021), por ejemplo, es un LDM que utiliza un codificador de texto,

CLIP (Radford et al. 2021), para condicionar las *prompts*. Aunque los modelos de difusión superan significativamente a sus predecesores en las tareas de *texto a imagen*, la ambigüedad en la generación de imágenes partiendo de descripciones textuales, ha motivado investigaciones relacionadas a la generación imágenes condicionadas por otros tipos de entradas, como es el caso de ControlNet.

2.2.1. ControlNet

ControlNet (L. Zhang et al. 2023) es una arquitectura (ver Figura 2.2) de redes neuronales que agrega controles de condicionamiento espacial a modelos de difusión previamente entrenados. ControlNet preserva la calidad y capacidades del modelo de difusión al bloquear sus parámetros, y a su vez crea una copia entrenable de sus *encoders*. Esta arquitectura trata al modelo preentrenado como un potente *backbone* para aprender diversos controles condicionales. La copia entrenable y el modelo original se conectan por medio de *convoluciones cero* (capas de convolución inicializadas en cero) que incrementan progresivamente los parámetros desde cero durante el entrenamiento.

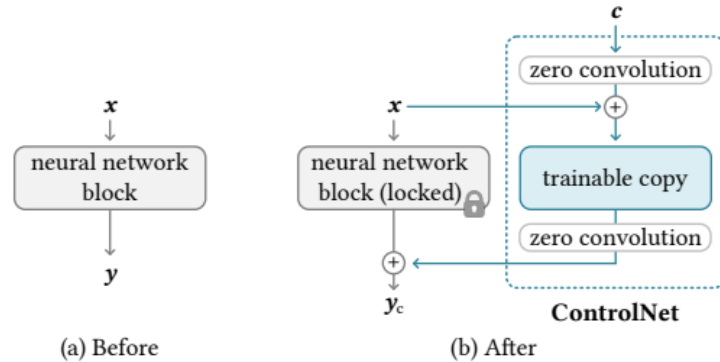


Figura 2.2: Arquitectura de ControlNet (L. Zhang et al. 2023).

Se han implementado diferentes ControlNets (ver Figura 2.3) utilizando como modelo preentrenado a Stable Diffusion (Rombach et al. 2021), de forma de probar diferentes condicionantes, incluyendo *keypoints* (puntos clave de poses) de Openpose (Cao et al. 2018). El ControlNet de Openpose se entrenó con un conjunto de alrededor de 200k imágenes, con sus respectivas poses y leyendas, recopiladas de Internet. Se pueden combinar múltiples modelos de ControlNet de forma simultánea para crear una MultiControlNet.



Figura 2.3: Ejemplos de ControlNets sobre Stable Diffusion (L. Zhang et al. 2023).

2.3. *Autoencoders* y VQ-VAE

Los *autoencoders* son redes neuronales diseñadas para aprender representaciones eficientes de datos, generalmente con el propósito de reducción de dimensionalidad o aprendizaje de características. Constan de dos componentes principales: un *encoder* que comprime la entrada en un espacio latente y un *decoder* que reconstruye la entrada a partir de esta representación latente.

Sin embargo, los *autoencoders* convencionales tienen limitaciones cuando se trata de tareas generativas. El espacio latente puede no estar bien estructurado, lo que dificulta el control del proceso de generación. Aquí es donde entran en juego los VAEs (Kingma y Welling, 2013). Los VAEs extienden el concepto de los autoencoders tradicionales al introducir un modelado probabilístico.

En lugar de codificar las entradas en un vector latente fijo, los VAEs modelan el espacio latente como una distribución. El *encoder* genera los parámetros de esta distribución (media y varianza), y el *decoder* toma muestras de esta distribución para generar salidas. Este enfoque permite a los VAEs generar nuevos datos tomando muestras del espacio latente aprendido, proporcionando un modelo generativo más robusto en comparación con los autoencoders convencionales.

Los Vector Quantized Variational Autoencoders (VQ-VAEs) (van den Oord et al. 2017) son una variante de los VAEs que utilizan la cuantización vectorial para crear una representación discreta del espacio latente. En lugar de codificar las entradas como distribuciones continuas en el espacio latente, los VQ-VAEs mapean las entradas a un conjunto finito de vectores latentes discretos a partir de un *codebook* (o diccionario) aprendido. Esto es beneficioso

para tareas donde las representaciones discretas son más naturales, logrando resultados prometedores en áreas como la síntesis de imágenes, generación de música o gestos del habla.

Un entrenamiento *naive* de un VQ-VAE es propenso al *colapso* del codebook, es decir, que solo se active un número reducido de códigos, lo que limita significativamente el rendimiento de la reconstrucción y la generación. Para mitigar este problema, se pueden utilizar diversas técnicas durante el entrenamiento, como ser Exponential Moving Average (EMA) (Williams et al. 2020) para la actualización del *codebook*, o *code reset* (Williams et al. 2020) que consiste en el restablecimiento de códigos inactivos durante el entrenamiento (J. Zhang et al. 2023).

En el Capítulo 4 se profundiza en la arquitectura, optimización y cuantización de una red VQ-VAE, al describir la implementación del *encoder* y del *decoder* de poses.

2.4. Notaciones de lenguas de señas

Las lenguas de señas son lenguas viso-espaciales en la que el señante utiliza el espacio tridimensional alrededor de su cuerpo para describir un evento. Las señas pueden contener componentes manuales así como componentes no manuales. Los componentes manuales pueden comprender la forma, la orientación, y los movimientos de las manos. Los componentes no manuales incluyen la utilización de cejas, los párpados, expresiones faciales, el movimiento de la cabeza o los hombros.

La naturaleza tridimensional de la lengua de señas impide que se escriba de igual manera que otros idiomas orales como el inglés o el español. Para plasmar los gestos tridimensionales en forma escrita, los investigadores han creado diversas notaciones, pero hasta ahora no existe una forma escrita estándar (Goyal, 2015).

La producción de lengua de señas se realiza frecuentemente utilizando alguna de estas notaciones intermedias, destacándose los sistemas de notación semánticos, como por ejemplo las glosas, o los sistemas de notación léxicos, como HamNoSys o SignWriting (Sutton y Frost, 2009).

2.4.1. Glosa

Las glosas permiten transcribir la lengua de señas a una forma textual, palabra por palabra. Es decir, una glosa representa una seña utilizando una palabra disponible en la lengua oral más habitual del lugar. La palabra elegida para representar cada seña usualmente representa uno de los significados de la seña (aunque no necesariamente el significado de la seña en contexto), lo que permite que personas que no están familiarizadas con el lenguaje de señas puedan entender mejor lo que se está signando (Goyal, 2015).

En algunos sistemas de glosas es común que las características no manuales se indiquen mediante una línea horizontal situada encima de la glosa, como en el caso de la ASL (ver Figura 2.4).

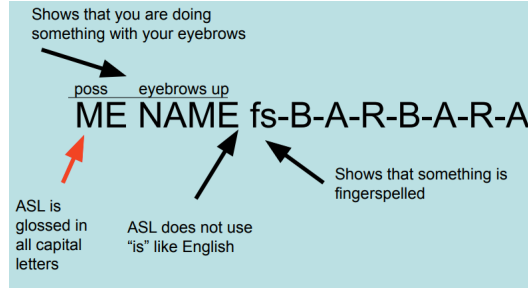


Figura 2.4: Ejemplo de glosa en ASL (Gill, 2005).

A continuación se muestra un ejemplo, extraído del corpus de iSignos (Cabeza y García-Miguel, 2019), donde se muestra la representación en Lengua de Señas Española (LSE) para las manos derecha (D) e izquierda (I), correspondiente al texto *¿Cómo te llamas?*:

T: ¿Cómo te llamas?
D: INDX.PRO(1):2sg NOMBRE
I: NOMBRE

En la mano derecha se indica al inicio un índice de tipo pronombre (INDX.PRO) que remite a la segunda persona del singular (2sg), para luego efectuar con ambas manos la seña correspondiente a la glosa NOMBRE.

2.4.2. Sistema de Notación de Hamburgo (HamNoSys)

HamNoSys es un sistema de notación fonética desarrollado por la Universidad de Hamburgo. La notación HamNoSys tiene sus raíces en un sistema

anterior denominado Stokoe (Stokoe, 1961), con más detalle en el manejo del componente no manual de la seña. La estructura de esta notación (como puede verse en la Figura 2.5) contiene principalmente cuatro componentes: un operador de simetría, el componente NMF (características no manuales), la configuración inicial de la mano, y un bloque de acción. La estructura y características de esta notación se describen en detalle en el Capítulo 5.

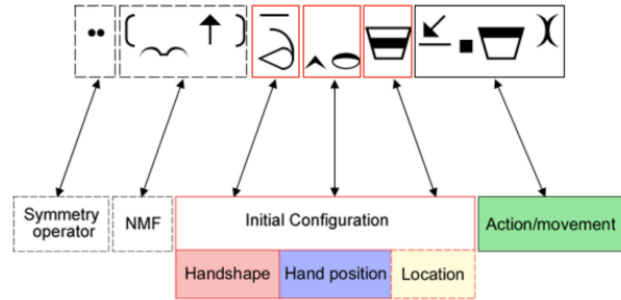


Figura 2.5: Estructura de HamNoSys (Skobov y Lepage, 2020).

2.4.3. SignWriting

SignWriting (Sutton y Frost, 2009) es un sistema que utiliza símbolos visuales para representar las formas de las manos, las expresiones faciales y los movimientos de las manos de la lengua de señas. La notación contiene 672 símbolos gráficos que se escriben en un diseño bidimensional.

La palabra signada *give* en ASL utilizando los símbolos de SignWriting puede verse en la Figura 2.6. El símbolo superior representa el parámetro de movimiento y el movimiento específico *alejándose del señante*. La primera parte del símbolo inferior representa la forma de la mano que se utiliza en la seña (dedos curvados y apretados juntos sobre el pulgar). El relleno negro sólido en la primera parte y el cuadro en la segunda parte del símbolo inferior representan la orientación de la palma (hacia arriba).



Figura 2.6: *Give* representado en SignWriting (Cripps, 2008).

2.4.4. Comparación de notaciones

Una ventaja del sistema de glosas es su capacidad para describir el componente no manual de las señas. Si bien HamNoSys destina una sección específica con estos fines, no resulta fácil describir las características no manuales, como las expresiones faciales y los movimientos corporales (Goyal, 2015).

Dado que las glosas son específicas de cada lengua de señas, cada tarea de traducción es única y requiere datos diferentes. En los sistemas HamNoSys y SignWriting en cambio, hay una correspondencia directa entre los símbolos (glifos) con aspectos de los gestos, como la ubicación, orientación, forma y movimiento de las manos. Dado que estos sistemas no son específicos de ningún idioma, permiten la transcripción invariante de signos en diferentes lenguas de señas, dándoles un carácter universal.

A diferencia de HamNoSys, SignWriting se escribe en una disposición espacial que no sigue un orden secuencial. Aunque esta disposición fue diseñada para una mejor legibilidad humana, es menos *amigable* para las computadoras, que esperan un orden de texto secuencial (Shalev-Arkushin et al. 2023).

En la Figura 2.7 puede verse la Lengua de Señas Alemana para *Haus* (casa). La representación mediante glosas es un identificador semántico único, mientras que HamNoSys y SignWriting describen la fonología de la seña: “*Dos manos extendidas con las palmas hacia abajo, una hacia la otra, tocándose, y luego moviéndose simétricamente en diagonal hacia abajo*”.



Figura 2.7: Lengua de Señas Alemana para *Haus* (Shalev-Arkushin et al. 2023).

2.5. Generación de poses de lengua de señas

Como se mencionó anteriormente, los enfoques de producción de lengua de señas pueden dividirse en las siguientes etapas:

1. Traducción del lenguaje oral a un sistema de notación intermedia.
2. Generación de una secuencia de poses a partir de la notación intermedia.

3. Generación de un avatar a partir de la secuencia de poses.

La investigación en cuanto al segundo paso, correspondiente a la generación de poses, se ha centrado en la utilización de las glosas como notación intermedia. En Stoll et al. 2018 se propone el primer enfoque que adopta este *pipeline* en 3 pasos, implementando la generación de poses a través de una tabla que contiene las correspondencias entre glosas y poses.

Text2sign (Stoll et al. 2020) sugiere combinar modelos generativos con un grafo de movimientos y traducción automática neuronal. En este enfoque, una red de tipo *encoder-decoder* genera una secuencia de probabilidades de glosas a partir del lenguaje oral. Esta secuencia se emplea para condicionar un grafo de movimientos, permitiendo así la obtención de una secuencia de poses que represente la entrada.

Otro enfoque, Progressive Transformers (Saunders et al. 2020b), sugiere una arquitectura basada en Transformers para la generación de poses a partir de glosas, de manera autorregresiva. Un trabajo similar (Saunders et al. 2020a) amplió el modelo Progressive Transformer a través de una red de tipo Mixture Density Network (MDN) para una generación de señas más expresiva.

También se han propuesto modelos no autorregresivos, por ejemplo en Xie et al. 2022 se explora un método no autorregresivo con una estrategia de difusión en tres etapas:

1. En una primera etapa se representa la secuencia de poses mediante códigos latentes usando un autoencoder de tipo VQ-VAE
2. Luego se propone el modelo G2P-DDM, una extensión de los modelos de difusión discretos, que modela las correspondencias entre las secuencias de glosas y poses.
3. Finalmente, desarrollan un método de clustering para predecir longitudes variables de secuencias de poses.

Si bien ha habido algunos trabajos para la generación de avatares utilizando como notación intermedia HamNoSys (con resultados poco satisfactorios), escasos han sido los estudios sobre la generación de poses a partir de HamNoSys. Ham2Pose (Shalev-Arkushin et al. 2023) es el primer intento, proponiendo un *pipeline* compuesto por dos partes: un *Procesador de Texto* y un *Generador de Poses*, que se explica en detalle a continuación.

2.6. Ham2Pose

Dada una seña escrita en HamNoSys, el modelo Ham2Pose genera una secuencia de poses que representan la seña deseada.

Arquitectura

El modelo consta de dos partes: el *Procesador de Texto*, responsable de la codificación de texto en HamNoSys y la predicción de la longitud de la secuencia de poses a generar; y el *Generador de Poses*, encargado de la generación de la secuencia de poses.

El modelo recibe una secuencia HamNoSys y una pose inicial. Esta pose de referencia (que puede ser una pose en reposo) es el punto de partida de la secuencia a generar. El *pipeline* del modelo es el siguiente:

- El texto HamNoSys de entrada es tokenizado, embebido y codificado por el *Procesador de Texto*. Luego, se pasa al *Predictor de Longitud de Secuencia* que predice la longitud de la secuencia.
- A continuación, la pose de referencia se multiplica hasta alcanzar la longitud de pose deseada (durante el entrenamiento es la longitud del *ground truth*, mientras que en la inferencia, es la longitud predicha) y se pasa al *Generador de Poses* junto con el texto HamNoSys codificado.
- Finalmente, utilizando el texto codificado y la pose de referencia extendida, el *Generador de Poses* refina gradualmente la secuencia de poses durante T pasos para obtener la pose deseada (inspirándose en los modelos de difusión).

La arquitectura (que puede verse en la Figura 2.8) utiliza *encoders* de Transformers para la codificación de la secuencia HamNoSys y la codificación de las poses.

2.6.1. Dataset

El *dataset* utilizado en Ham2Pose consta de 5754 videos de signos de lenguas de señas con sus transcripciones en HamNoSys. Cada video muestra a una persona de frente haciendo un solo signo. Los datos se recopilaron del *corpus* DGS (Hanke, 2009), de Dicta-Sign (Matthes et al. 2012) y de un *corpus* de la Lengua de Señas Polaca (Hanke, 2009).

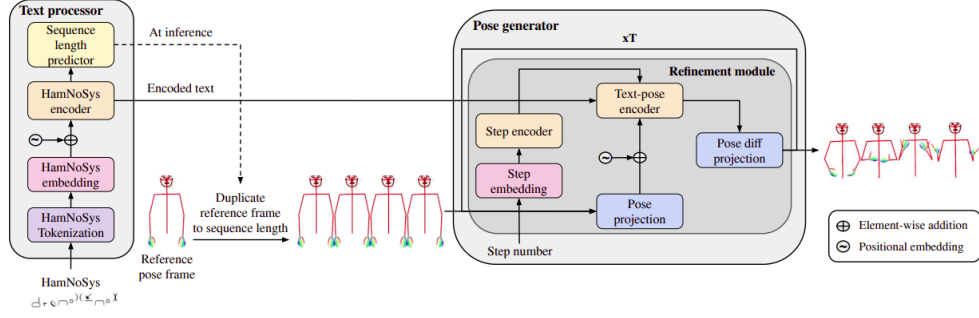


Figura 2.8: Arquitectura de Ham2Pose (Shalev-Arkushin et al. 2023).

En conjunto, los datos contienen señas correspondientes a cuatro lenguas de señas, realizadas por 14 señantes:

- Lengua de Señas Polaca (PJM): 2560 signos, 2 señantes.
- Lengua de Señas Alemana (DGS): 1926 signos, 8 señantes
- Lengua de Señas Griega (GSL): 887 signos, 2 señantes.
- Lengua de Señas Francesa (LSF): 381 signos, 2 señantes.

Con el fin usar estos videos como *ground truth* para la generación de las secuencias de poses, se estimaron *keypoints* usando el modelo de estimación de pose de OpenPose (Cao et al. 2018). Cada *keypoint* $k_i \in K$ consta de una localización en coordenadas 2D (x, y) , y una confianza c_i en la localización del *keypoint*. El número de *keypoints* extraídos es de 137 por *frame* de video, abarcando el cuerpo (25), la cara (70) y las manos (21 por mano).

2.6.2. Preprocesamiento de los datos

El conjunto de datos se preprocesó teniendo en cuenta las siguientes consideraciones:

- **Enmascarar *keypoints* de baja confianza:** se enmascaran aquellos *keypoints* con una confianza menor al 20 %.
- **Recortar *frames* sin sentido:** algunos videos, por ejemplo videos que se desvanecen (*fade in*, *fade out*) contienen *frames* iniciales o finales que no contienen suficiente información relevante. Además, en posiciones de reposo, las manos no siempre son visibles en el *frame*. Por lo tanto, se eliminan los *frames* iniciales y finales en los cuales no se identifica la cara o ambas manos (en base a determinados umbrales predefinidos).

- **Enmascarar las piernas:** dado que las piernas no son significativas, se eliminan, estableciendo en 0 la confianza de cada *keypoint* desde la cintura hacia abajo.
- **Invertir señas con la mano izquierda.** Las señas de una sola mano generalmente se hacen con la mano dominante. Una persona zurda espejaría una seña hecha por una persona diestra. Dado que la mano que realiza la seña no está especificada en HamNoSys, y que algunos videos del *dataset* incluyen señantes zurdos, para mantener la consistencia, estos videos se invierten para producir señas con la mano derecha.
- **Normalización de la pose.** Los *keypoints* de la pose se normalizan por los hombros de la pose, utilizando la biblioteca *pose-format* (Moryossef et al. 2023), de modo que todas las poses tengan la misma escala. Se define al cuello como centro de la pose, calculado como el punto medio promedio entre los hombros en todos los *frames*. Luego, se ajustan todos los *keypoints*, moviendo el centro a (0, 0), y se escala la pose de modo que la distancia promedio entre los hombros sea 1.

2.6.3. Métrica nDTW-MJE

Actualmente, no existe un método estándar de evaluación de la producción de lenguajes de señas. Ham2Pose propuso una nueva medida de distancia, *nDTW-MJE*, para medir la distancia entre secuencias de poses, que considera los *keypoints* faltantes. Para comprender *nDTW-MJE* es necesario previamente entender Dynamic Time Warping (DTW) y Dynamic Time Warping - Mean Joint Error (DTW-MJE).

DTW (Kruskal, 1983) es un algoritmo que puede medir la distancia entre dos series temporales, ya que es capaz de ignorar desplazamientos globales y locales en la dimensión temporal.

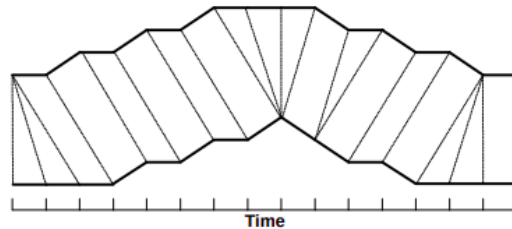


Figura 2.9: *Emparejamiento* de dos series temporales (Salvador y Chan, 2004).

En la Figura 2.9, cada línea diagonal conecta un punto en una serie temporal con su punto correspondiente en la otra serie temporal. La diferencia entre ambas series se mide mediante la suma de las distancias de cada par de puntos unidos por estas líneas. Si ambas series temporales fueran idénticas, las líneas serían estrictamente verticales.

DTW-MJE (Huang et al. 2021) mide la distancia media entre *keypoints* de poses después de alinearlos temporalmente usando DTW.

Como en la definición original de DTW-MJE no está claro cómo manejar los *keypoints* faltantes, Ham2Pose propone *nDTW-MJE*, definiendo la siguiente función de distancia para cada par de *keypoints*:

$$\text{dist}(\text{ref}, \text{other}) = \begin{cases} 0 & \text{ref} = \text{NaN} \\ \frac{\|\text{ref}\|_2}{2} & \text{ref} \neq \text{NaN} \ \& \ \text{other} = \text{NaN} \\ \|\text{ref} - \text{other}\|_2 & \text{otherwise} \end{cases} \quad (2.1)$$

De esta manera, solo se comparan *keypoints* existentes en ambas secuencias de poses y se penalizan aquellos *keypoints* que existen en la pose de referencia pero no en la pose predicha. Para medir la distancia entre ambas secuencias de poses, se normalizan previamente utilizando el procedimiento descrito en la Sección 2.6.2.

En la implementación de *nDTW-MJE* se utiliza FastDTW (Salvador y Chan, 2004), una versión más eficiente (de orden lineal) pero aproximada de DTW. FastDTW evita el enfoque de programación dinámica del algoritmo estándar de DTW, usando un enfoque multinivel que proyecta recursivamente una solución desde una resolución más baja y refina la solución proyectada.

2.6.4. Métrica Distance Ranks

El proyecto Ham2Pose presenta la métrica ***Distance Ranks*** para evaluar la precisión de un método de animación o generación de secuencias de poses. Esta métrica establece dos modalidades de comparación:

- **Predicción como referencia:** Se utiliza la secuencia generada por el modelo como referencia.
- **Ground Truth (GT) como referencia:** Se emplea la secuencia real como referencia.

Para cada elemento del conjunto de test se realiza el siguiente proceso (ver Figura 2.10):

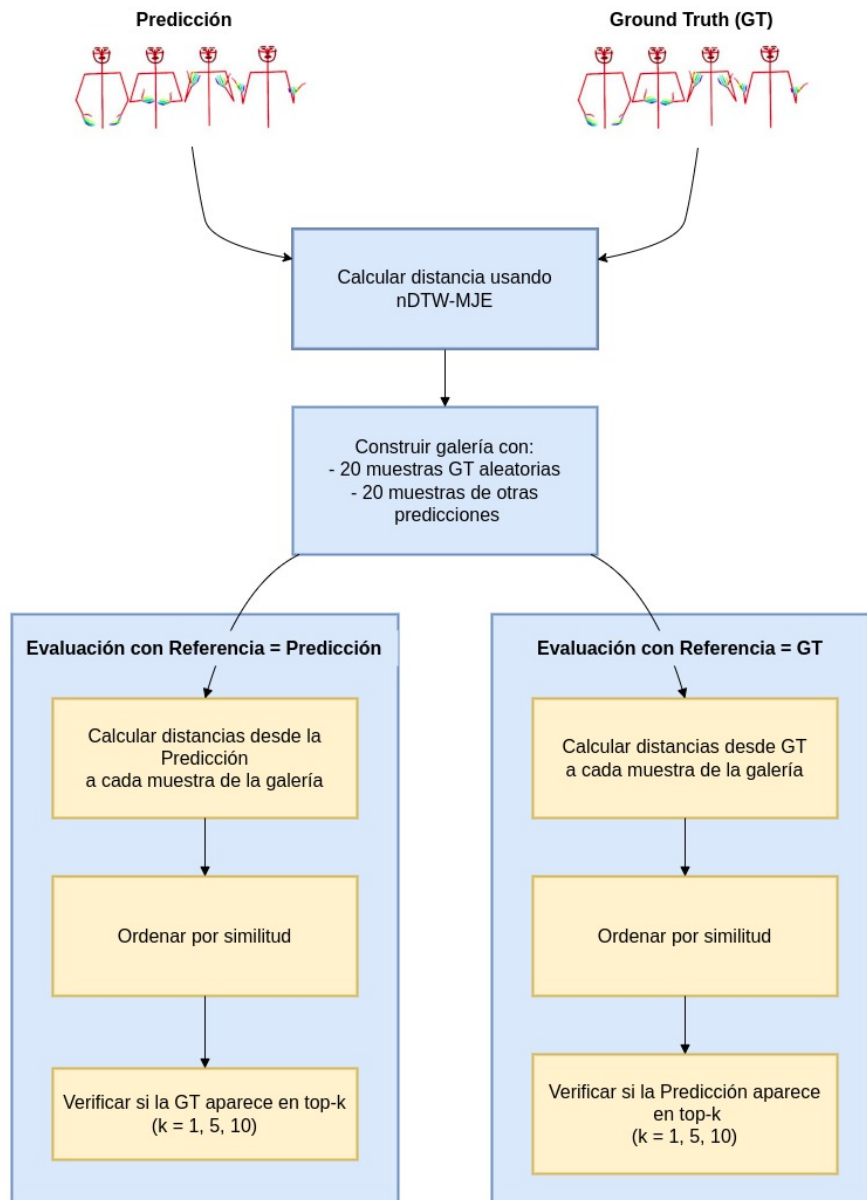


Figura 2.10: Diagrama del proceso evaluación utilizando la métrica *Distance Ranks* para un elemento del conjunto de test.

1. Cálculo de la distancia inicial:

- Se predice la secuencia de poses para el elemento utilizando el modelo implementado.
- Se calcula la distancia entre la secuencia predicha y la secuencia de GT utilizando $nDTW-MJE$.

2. Construcción de la galería de muestras:

- Se seleccionan 20 secuencias GT aleatorias del conjunto de test.
- Se seleccionan 20 secuencias predichas aleatorias del conjunto de test.
- Se conforma una galería compuesta por las 40 secuencias seleccionadas.

3. Evaluación del ranking (según modalidad):

- Se calcula la distancia, mediante $nDTW-MJE$, entre una **referencia** (la secuencia de poses predicha o la secuencia GT , según la modalidad) y cada una de las secuencias de la galería.
- Se ordenan estas distancias de forma ascendente, ubicando en las primeras posiciones a las secuencias con menor distancia (mayor similitud).
- Se verifica si la secuencia de interés (la secuencia de poses predicha o la secuencia GT , según la modalidad) aparece entre las k secuencias más similares ($k = 1, 5, 10$).

El proceso anterior se repite para cada elemento del conjunto de test. Posteriormente, se promedian los resultados obtenidos para obtener las métricas finales (por modalidad): **Rank 1**, **Rank 5** y **Rank 10**, definidas como el porcentaje de elementos del conjunto de test para los cuales la secuencia de interés aparece entre las k secuencias más similares.

2.7. Generación de poses a partir de texto

Generar secuencias de poses a partir de descripciones textuales puede utilizarse en numerosas aplicaciones, como en la industria de los videojuegos o la producción cinematográfica. Pero es una tarea desafiante ya que las poses y el texto provienen de modalidades diferentes. Se espera que un modelo aprenda una correspondencia entre el espacio del lenguaje y el espacio de las poses.

Muchos trabajos proponen generar *embeddings* unificados para el lenguaje y las poses utilizando *autoencoders*. Por ejemplo, MotionCLIP (Tevet, Gordon et al. 2022), alinea el espacio de las poses con el espacio de CLIP (Radford et al. 2021). Este enfoque muestra capacidad de generar movimientos fuera de la distribución del entrenamiento pero las secuencias generadas no son de alta calidad. TEMOS (Petrovich et al. 2022) en cambio, propone el uso de VAEs basados en Transformers para la generación de poses a partir de texto, pero se basa en oraciones cortas y tiene dificultades para trabajar con descripciones fuera de distribución.

Más recientemente, modelos basados en difusión, como MDM (Tevet, Raab et al. 2022) y MotionDiffuse (M. Zhang et al. 2022), han pasado a dominar la tarea de generación de secuencias de poses a partir de texto. Sin embargo, enfoques más clásicos, como T2M-GPT (J. Zhang et al. 2023), que emplea una combinación de una red VQ-VAE y un GPT para la generación de secuencias, han demostrado, en algunos casos, mejores rendimiento que enfoques basados en difusión.

Finalmente, MotionGPT (Y. Zhang et al. 2023), inspirado en T2M-GPT, propone un *fine tuning* de un LLM siguiendo instrucciones específicamente diseñadas para la generación de secuencias de poses.

2.7.1. MotionGPT

MotionGPT puede utilizar señales de control multimodales, como texto y poses, para generar movimientos humanos consecutivos, al tratar estas señales multimodales como *tokens* especiales de LLMs.

Como motivación de este trabajo se sostiene la idea de que los desafíos relacionados con la generación de movimiento humano podrían resolverse de manera natural mediante adaptaciones de LLMs por las siguientes razones:

- Estudios recientes han demostrado que los LLMs pueden entender entradas multimodales, como MiniGPT-4 (Zhu et al. 2023) y VideoChat (Li et al. 2024), a través de un adaptador liviano como LoRA (Hu et al. 2021), y por lo tanto los LLMs también podrían entender secuencias de movimiento con un adaptador adecuado.
- Los LLMs pueden proporcionar contextos diversos de movimiento humano para la generación de poses porque han codificado patrones de

movimiento a partir de las grandes cantidades de textos con las que han sido entrenados.

Concretamente, en el proyecto MotionGPT (ver Figura 2.11):

- Se mapean poses humanas a códigos a través de una red VQ-VAE entrenada específicamente con este fin.
- Se generan *prompts* con instrucciones combinando texto y códigos de poses.
- Se adaptan LLMs utilizando LoRA para que respondan de manera eficiente con secuencias de poses humanas, a las instrucciones recibidas.

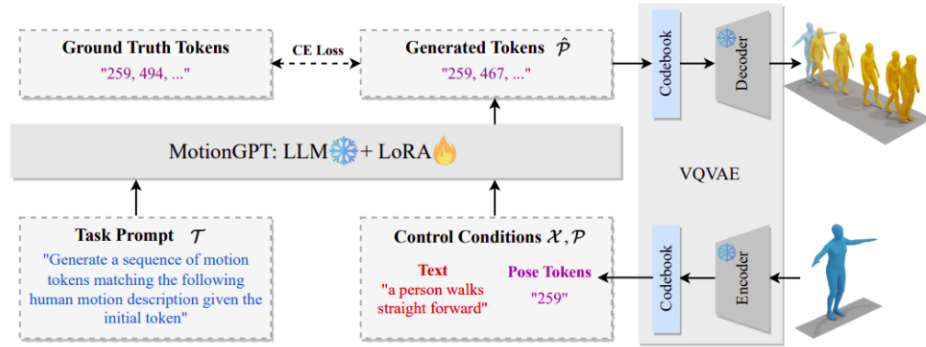


Figura 2.11: Método de MotionGPT (Y. Zhang et al. 2023).

Se experimentó con diversos modelos base preentrenados, incluyendo LLaMA-7B y LLaMA-13B, e inclusive un LLaMA sin preentrenamiento. Los resultados mostraron una fuerte correlación entre el nivel de preentrenamiento de los LLMs y el rendimiento del modelo en la tarea de generación de secuencias de poses. Esto destaca la influencia significativa del conocimiento previo extraído de los LLMs.

En cuanto al preentrenamiento de la red VQ-VAE, se sigue la estructura y la estrategia de entrenamiento utilizada por T2M-GPT (J. Zhang et al. 2023).

MotionGPT exhibió un rendimiento competitivo con los métodos de última generación para la generación de movimiento a partir de texto, logrando resultados comparables para los datasets HumanML3D (Guo et al. 2022) y KIT-ML (Plappert et al. 2016).

2.8. Generación de avatares de lengua de señas

Uno de los problemas más desafiantes vinculados a la lengua de señas es la generación de videos sintéticos que permitan a sus usuarios experimentar una comunicación natural y fluida, similar a la comunicación entre humanos. Antes de la era del *deep learning*, el problema de la producción de lenguaje de señas se abordaba utilizando avatares animados (por ej: VisiCast (Bangham et al. 2000) o JASigning (Elliott et al. 2010)). Sin embargo, en términos de apariencia y movimiento estos avatares no resultaban realísticos, afectando la aceptación y el involucramiento de los usuarios con este tipo de tecnologías.

Con la llegada del *deep learning*, se han introducido nuevos métodos que permiten la generación de videos con avatares que tienen la apariencia de personas reales. Para mejorar la comprensión de las señas, los enfoques más recientes van un paso más allá y aplican técnicas de *retargeting* (transferencia de movimiento de un personaje o intérprete a otro) para transformar secuencias de poses de lengua de señas a un video de un actor humano fotorrealista.

La generación de vídeos humanos realistas a partir de señales de control, como *keypoints* de una pose humana, ha sido estudiada extensamente en los últimos años, especialmente con el éxito de los modelos basados en Generative Adversarial Networks (GANs) para la generación condicionada de videos. Por ejemplo, Vid2vid (Wang et al. 2018) presenta un enfoque utilizando GANs condicionales, incorporando flujo óptico y restricciones de consistencia temporal para generar secuencias de video de poses que son tanto realistas como diversas.

SignGAN (Saunders et al. 2020a) fue el primer modelo de producción de lengua de señas en generar videos continuos fotorrealistas condicionando la síntesis a una secuencia de poses obtenidas previamente, y a la imagen de un signante de referencia (Tze et al. 2023).

Aunque los avances provistos por los modelos GAN son significativos, los recientes avances en generación de imágenes utilizando modelos de difusión ofrecen nuevas perspectivas en lo que respecta a la generación de videos a partir de poses y, en particular, en la producción de lengua de señas.

2.9. Modelos de difusión para la generación de video

El éxito de los modelos de difusión para generación de imágenes a partir de texto, ha hecho que se utilice su estructura como inspiración en la investigación de la generación de videos a partir de texto. A continuación, se describen los enfoques propuestos por *Text2Video-Zero* (Khachatryan et al. 2023), *Follow Your Pose* (Ma et al. 2024) y *TemporalKit* (Rowles, 2023).

2.9.1. Text2Video-Zero

Este trabajo explora el problema de la generación de un video a partir de un texto bajo la modalidad *zero shot*, que es la tarea de generar videos a partir de indicaciones textuales sin requerir ninguna optimización o finetuning. Un punto clave de este enfoque es la modificación de un modelo de difusión preentrenado (por ejemplo Stable Diffusion), enriqueciéndolo con una generación temporalmente consistente. Para garantizar la consistencia temporal, presentan dos modificaciones innovadoras y ligeras:

1. Enriquecer los códigos latentes de los frames generados con información de movimiento para mantener tanto la escena global como el fondo temporalmente consistentes.
2. Usar atención cruzada de cada frame sobre el primero para preservar el contexto, la apariencia y la identidad del objeto en primer plano a lo largo de toda la secuencia.

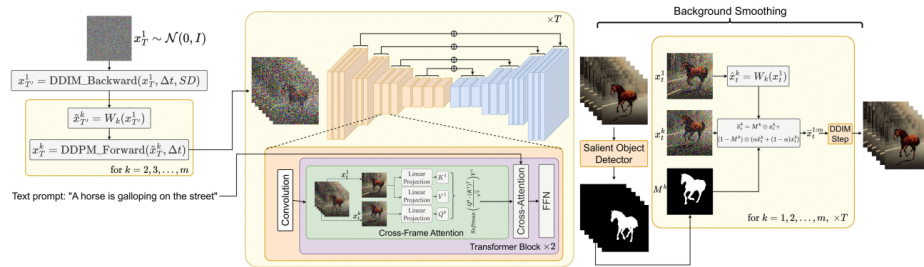


Figura 2.12: Arquitectura de *Text2Video-Zero* (Khachatryan et al. 2023).

2.9.2. Follow Your Pose

El objetivo de este trabajo es crear videos de personajes a partir de descripciones textuales y también controlar su pose mediante una señal de control dada, por ejemplo esqueletos humanos.

Proponen una estrategia de entrenamiento en dos etapas, con bloques ajustados a través de un modelo preentrenado que genera imagen a partir de texto (por ejemplo Stable Diffusion), utilizando datasets fácilmente obtenibles: pares de imágenes-poses y videos sin poses. Concretamente, en la primera etapa, solo se utilizan los pares imágenes-poses para la generación controlada de una imagen a partir de texto, entrenando un *encoder* convolucional para codificar la información de la pose. En la segunda etapa, ajustan el movimiento de la red anterior utilizando videos sin poses, mediante la adición de bloques de *self-attention* temporal y *self-attention* entre *frames* (inspirados en Text2Video-Zero).

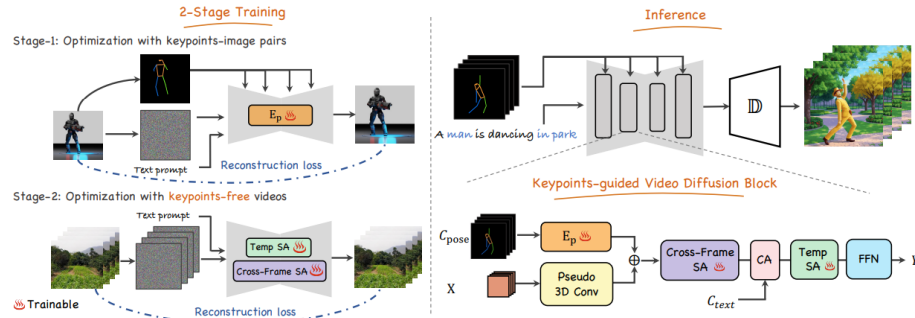


Figura 2.13: Método de *Follow Your Pose* (Ma et al. 2024).

2.9.3. TemporalKit

TemporalKit es una extensión de **Stable Diffusion WebUI**⁴, que agrega capacidades de procesamiento de video y estabilidad temporal a Stable Diffusion. Stable Diffusion WebUI o Automatic1111, desarrollada por la comunidad, es la interfaz web de facto para ejecutar modelos de Stable Diffusion.

TemporalKit permite a los usuarios:

- Descomponer un video en segmentos más pequeños para su procesamiento.

⁴<https://github.com/AUTOMATIC1111/stable-diffusion-webui>

- Aplicar generación de imagen a imagen usando Stable Diffusion a cada *frame* del video, por ejemplo, para reemplazar el personaje del *frame* por otro.
- Combinar los *frames* transformados para generar un nuevo video.

TemporalKit logra esto a través del procesamiento en *batches* y de integración con otras herramientas como ControlNet.

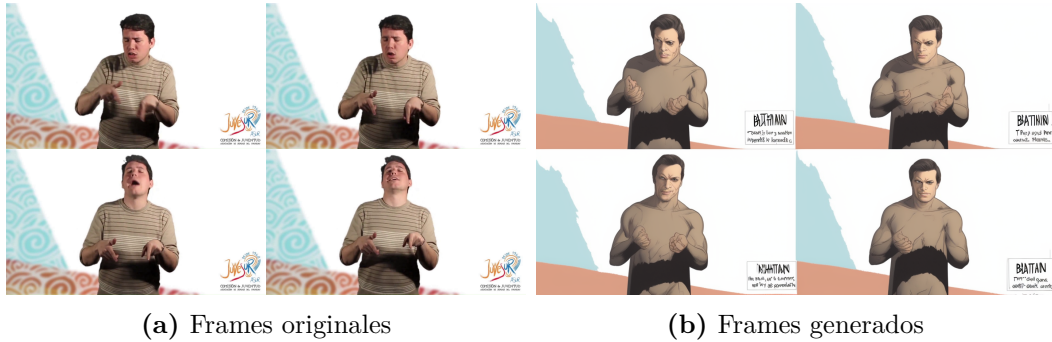


Figura 2.14: Experimento con TemporalKit para imitar seña de un video con otro personaje.

2.9.4. Resultados preliminares utilizando modelos de difusión

Las pruebas iniciales realizadas en el contexto de esta tesis con los modelos presentados en esta sección muestran mejoras respecto a trabajos previos; sin embargo, aún presentan deficiencias significativas que dificultan la generación de avatares humanos convincentes. Esto se evidencia, por ejemplo, en la Figura 2.14 utilizando TemporalKit. Entre los principales problemas identificados se encuentran:

- Es necesario mejorar la consistencia temporal entre *frames* para lograr transiciones más fluidas.
- Los personajes generados en cada frame presentan variaciones, lo que impide una continuidad visual precisa.
- La imitación de la pose de las manos muestra errores, en muchos casos debido a que los modelos ControlNet fueron entrenados con *datasets* de OpenPose que no incluyen *keypoints* de las manos.

2.10. Conclusión

A continuación se exponen las conclusiones derivadas de esta revisión de los conceptos y del estado del arte, así como las decisiones adoptadas para el desarrollo de la tesis.

Entre las posibles líneas de investigación, la producción de lengua de señas se presenta como un área menos desarrollada y con grandes desafíos, por lo que ha sido elegida para el contexto de este trabajo.

En la fase exploratoria, se analizaron 2 etapas de la producción de lengua de señas: la generación de poses a partir de la representación intermedia, y la generación de avatares a partir de las poses. Las pruebas realizadas con las técnicas más recientes, analizadas en la Sección 2.9, parecen indicar que el estado del arte aún no cuenta con la madurez necesaria para la generación de videos de calidad condicionados a poses humanas. Por lo tanto, se opta por concentrar esfuerzos en la generación de poses a partir de una representación intermedia.

A pesar de que, paralelamente al desarrollo de este trabajo, se están realizando esfuerzos para generar un *corpus* de Lengua de Señas Uruguay, aún no se dispone de datos suficientes y unificados. Esta limitante lleva a descartar los sistemas de notación semánticos, como las glosas, que son dependientes de cada lengua en particular. En su lugar, se opta por trabajar con sistemas de notación léxicos debido a su carácter universal, lo que permite que el trabajo aquí presentado pueda ser potencialmente reutilizado para la producción de lengua de señas en el contexto de Uruguay.

Dentro de los sistemas de notación léxicos analizados, HamNoSys se escribe en una disposición secuencial, siendo más adecuado para el procesamiento computacional que SignWriting. Por consiguiente, HamNoSys es el sistema elegido para este proyecto.

Definido el objetivo de centrar el trabajo de esta tesis en la generación de poses de lengua de señas a partir de HamNoSys, resta establecer el método a utilizar. Como se mencionó, Ham2Pose es el primer intento de abordar este problema, proponiendo un método que consta de dos módulos: un *Procesador de Texto*, responsable de la codificación del texto HamNoSys, y un *Generador de Poses*, utilizando *encoders* basados en Transformers para ambos módulos. Estos Transformers son entrenados desde cero, sin utilizar en ninguna etapa modelos preentrenados como LLMs, modelos de difusión o *encoders* como CLIP

o BERT.

En el proyecto Ham2Pose se indica que no se utilizaron modelos preentrenados debido a que HamNoSys no es un lenguaje común y se disponen de recursos limitados para realizar reentrenamientos. Sin embargo, al ser HamNoSys una notación fonética donde existe correspondencia entre los símbolos y aspectos de los gestos, como la ubicación u orientación de la mano, esta tesis propone la idea de generar un diccionario basado en los símbolos de HamNoSys para traducir las secuencias a lenguaje natural. De esta forma, el problema se transforma en una tarea de generación de poses a partir de texto, lo que permite aprovechar el conocimiento y las capacidades encapsuladas en los grandes modelos de lenguaje preentrenados.

Dentro de la exploración realizada en la Sección 2.5, se opta por utilizar el método propuesto por MotionGPT. Este método genera secuencias de poses a partir de descripciones textuales de movimientos mediante el ajuste de un LLM con instrucciones, y utilizando una red VQ-VAE para el mapeo de secuencias de poses a códigos discretos.

Capítulo 3

Método

En este capítulo se introduce el método planteado para la generación de secuencias de poses que representen signos en lengua de señas, a partir de descripciones HamNoSys. Para ello se propone una arquitectura compuesta por tres módulos, implementados como parte de este proyecto: un generador de descripciones HamNoSys, un LLM ajustado mediante *Instruction Tuning*, y una red VQ-VAE para la codificación y decodificación de poses.

A grandes rasgos, el método puede dividirse en los siguientes pasos que pueden apreciarse en la Figura 3.1:

1. **Generación de la descripción de una secuencia HamNoSys:**

El proceso comienza con la entrada de una secuencia de símbolos HamNoSys. El primer módulo, el generador de descripciones HamNoSys, fue implementado para recibir esta secuencia y generar descripciones en diferentes formatos: desde representaciones basadas directamente en los símbolos HamNoSys hasta descripciones más detalladas en lenguaje natural. En el diagrama se ilustra un ejemplo donde se genera una descripción de la secuencia HamNoSys a partir de la explicación en lenguaje natural de cada símbolo que forma la secuencia.

2. **Generación de una secuencia de códigos de poses a partir de una descripción:**

En un siguiente paso, un LLM recibe una instrucción que contiene la descripción de una secuencia HamNoSys y genera una secuencia de *tokens* de poses de lengua de señas. El LLM, que ha sido ajustado como parte de este trabajo mediante *Instruction Tuning* y LoRA, recibe instrucciones compuestas por una descripción de la tarea, que define el objetivo que

el modelo debe cumplir (en este caso, generar *tokens* de pose correspondientes a una seña), y la descripción HamNoSys generada por el primer módulo.

3. Reconstrucción de las Poses a partir de una secuencia de códigos:

Finalmente, se utiliza una red VQ-VAE, entrenada en el contexto de este trabajo, para la decodificación de los *tokens* generados por el LLM.

Cabe destacar que la red VQ-VAE es también utilizada para codificar las poses del *dataset* de Ham2Pose a *tokens* discretos. Estos *tokens* codificados actúan como el *ground truth*, proporcionando la referencia necesaria para que el LLM aprenda a generar los *tokens* que representen correctamente las poses descritas por el generador de descripciones HamNoSys.

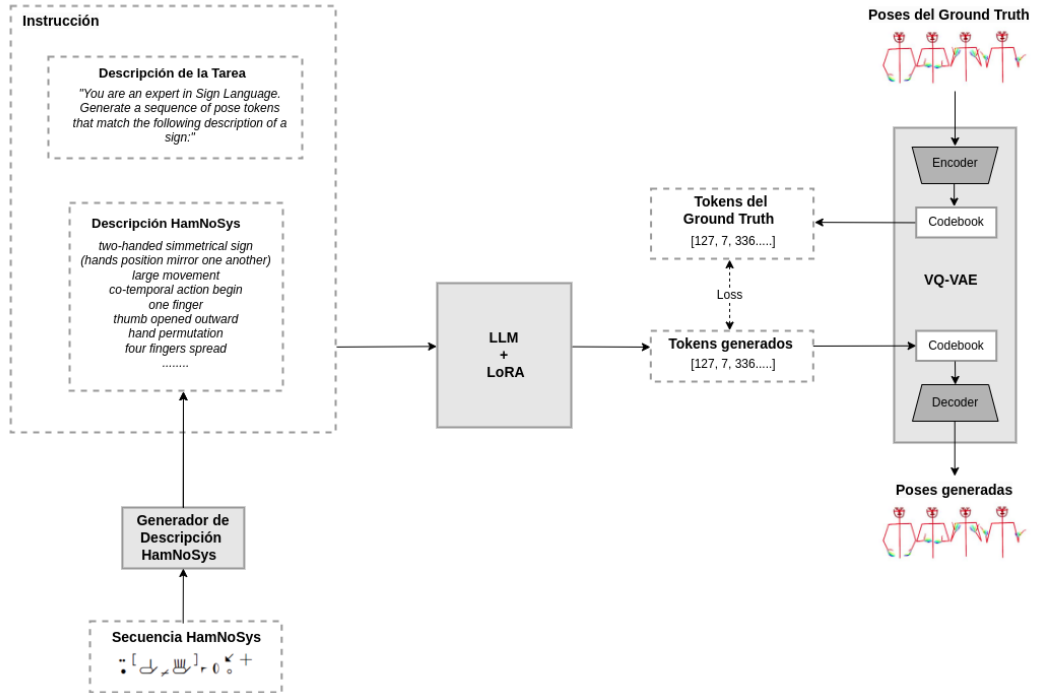


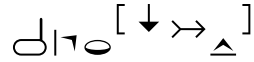
Figura 3.1: Método propuesto

3.1. Ejemplo

A continuación, se ejemplifica el proceso descrito anteriormente, generando la secuencia de poses en lengua de señas a partir de una secuencia HamNoSys

del conjunto de datos de validación, utilizando los componentes desarrollados en esta tesis:

1. Se parte de una secuencia de **símbolos HamNoSys**:



2. El **generador de descripciones HamNoSys**, utilizando la modalidad *secuencia con descripciones* descrita en la Sección 5.4.3, genera la siguiente secuencia con la descripción de cada símbolo:

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

3. A partir de la secuencia con descripciones de los símbolos HamNoSys se genera una **instrucción**, siguiendo el formato definido en la Sección 6.2:

Instrucción

```

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

```

4. Se realiza una inferencia utilizando un **LLM**, ajustado mediante *Instruction Tuning* según lo descrito en el Capítulo 6, para generar la siguiente salida:

Inferencia del LLM

```
[401, 464, 409, 170, 491, 170, 491, 409, 497, 497, 497,  
225, 497, 33, 141, 401, 401] </s>
```

5. Se utiliza el *decoder* de la red **VQ-VAE**, detallada en el Capítulo 4, para generar la secuencia de poses de lengua de señas a partir de la secuencia de *tokens* generada por el LLM. La secuencia generada se puede visualizar en la Figura 3.2 mientras que la secuencia de frames del video original se muestra en la Figura 3.3.

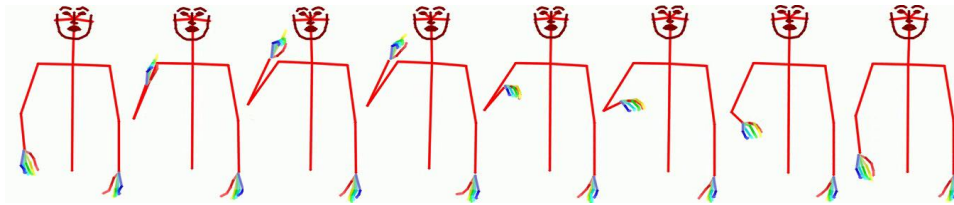


Figura 3.2: Secuencia de poses generada.

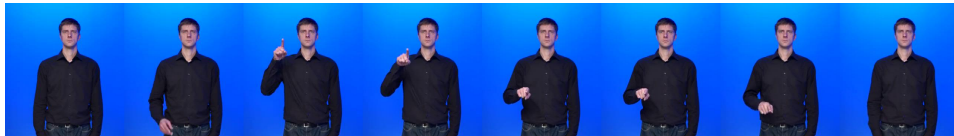


Figura 3.3: Secuencia de *frames* del video original.

Capítulo 4

VQ-VAE de poses

En este capítulo se describe la implementación de la red VQ-VAE, utilizada para la codificación y decodificación de secuencias de poses de lengua de señas dentro del *pipeline* propuesto en esta tesis.

En primer lugar, se presentan las características y los componentes principales de la red VQ-VAE. A continuación, se aborda el objetivo de la optimización de esta red neuronal, con especial énfasis en la versión utilizada en este trabajo.

Posteriormente, se detallan las estrategias de cuantización empleadas, específicamente EMA y *code reset*, describiendo sus fundamentos y su impacto en el modelo.

Luego, se expone la arquitectura de la red junto con los hiperparámetros utilizados y el conjunto de datos empleado.

En la sección de resultados, se introduce el método *baseline* utilizado como referencia y se presentan los resultados experimentales obtenidos. Se validan distintas configuraciones para la longitud de las secuencias y la *tasa de down-sampling*, analizando el comportamiento de las distintas métricas de evaluación.

Finalmente, se discuten las conclusiones derivadas de la implementación de este componente, resaltando los hallazgos más relevantes.

4.1. Descripción general de la red VQ-VAE

Como se mencionó anteriormente, el objetivo de una red VQ-VAE es realizar un proceso de codificación y compresión de datos que permita obtener

representaciones latentes discretas en lugar de continuas.

La VQ-VAE utilizada en el contexto de este proyecto, se basa en implementación de (Y. Zhang et al. 2023), que a su vez está basada en (J. Zhang et al. 2023), y su esquema general puede verse en la Figura 4.1.

Dada una secuencia de poses $X = [x_1, x_2, \dots, x_T]$ con $x_t \in \mathbb{R}^d$, donde T es el número de *frames* y d es la dimensión de la pose, el objetivo es recuperar la secuencia de poses a través de un *autoencoder* y un *codebook* (que se aprende) que contiene K códigos $C = \{c_k\}_{k=1}^K$ con $c_k \in \mathbb{R}^{d_c}$, donde d_c es la dimensión de los códigos.

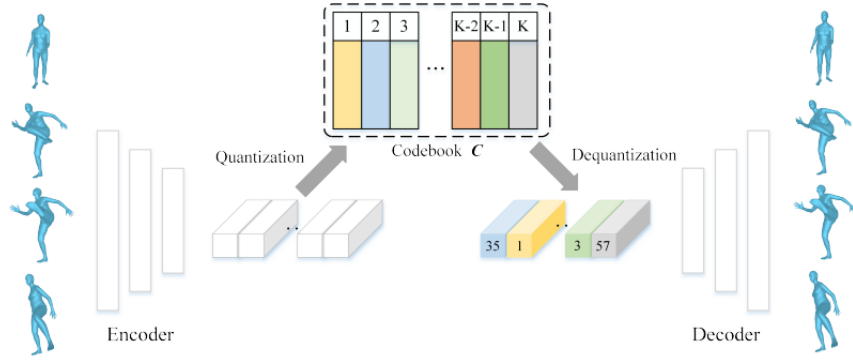


Figura 4.1: Esquema de VQ-VAE (J. Zhang et al. 2023).

Con el *encoder* y el *decoder* del *autoencoder* denotados por E y D respectivamente, la representación latente, Z , se puede calcular como $Z = E(X)$ con $Z = [z_1, z_2, \dots, z_{T/l}]$ y $z_i \in \mathbb{R}^{d_c}$, donde l representa la *tasa de downsampling temporal* del *encoder* E . Esta tasa representa el factor por el cual se reduce la cantidad de *frames* de la secuencia original para generar la representación latente, es decir, que la secuencia original de T *frames* se comprime a T/l pasos temporales.

Durante la cuantización, cada representación (vector) latente se asigna al elemento más cercano dentro del *codebook*, lo que permite convertir la representación continua en una representación discreta. Lo anterior puede expresarse de la siguiente manera:

$$\hat{z}_i = \arg \min_{c_k \in C} \|z_i - c_k\|_2$$

La decuantización transforma los códigos discretos cuantizados de nuevo en vectores continuos, que representan de manera aproximada las características

latentes originales. Finalmente, el *decoder* recibe estos vectores continuos y genera una secuencia que intenta reconstruir la secuencia original de poses.

4.2. Optimización

El objetivo estándar de optimización de una red VQ-VAE contiene tres componentes: un *loss de reconstrucción* ($\mathcal{L}_{re}$), un *loss de embeddings* ($\mathcal{L}_{embed}$) y un *loss de commitment* ($\mathcal{L}_{commit}$).

El $\mathcal{L}_{embed}$, penaliza la distancia entre la salida del *encoder* y el vector de *embedding* seleccionado del diccionario, forzando a que el *codebook* se alinee con las representaciones del *encoder*. En este cálculo se detiene el gradiente en la salida del *encoder* para actualizar únicamente los *embeddings* del diccionario. Como en la implementación utilizada en este proyecto se emplean las técnicas *code reset* y EMA para actualizar el diccionario, se omite el uso de $\mathcal{L}_{embed}$.

En consecuencia, en el contexto de este trabajo:

$$\mathcal{L}_{vqvae} = \mathcal{L}_{re} + \mathcal{L}_{commit}$$

El objetivo de $\mathcal{L}_{re}$ es optimizar tanto el *encoder* como el *decoder*, evaluando qué tan bien el *decoder* es capaz de reconstruir la entrada original. Para implementarlo se sigue la línea de (J. Zhang et al. 2023), al utilizar *L1* suavizada ($\mathcal{L}_1^{\text{smooth}}$) (no se incluye un término extra de regularización basado en la velocidad, no aplicable al *dataset* de coordenadas 2D con el que se trabaja en este proyecto).

Por lo tanto:

$$\mathcal{L}_{re} = \mathcal{L}_1^{\text{smooth}}(X, X_{re})$$

Por su parte, $\mathcal{L}_{commit}$ incentiva al *encoder* a *comprometerse* con el vector asignado del diccionario, penalizando la diferencia entre la salida del *encoder* y el vector del *codebook*. Esto ayuda a estabilizar el entrenamiento y mantener una representación latente coherente. Es decir,

$$\mathcal{L}_{commit} = \beta \|Z - \text{sg}[\hat{Z}]\|_2$$

donde:

- Z es la salida del *encoder* antes de cuantizar (*embedding* continuo),

- $\hat{Z}$ es el *embedding* cuantizado (el vector seleccionado del *codebook*),
- β es un hiperparámetro que controla la magnitud de la penalización,
- $\|\cdot\|_2^2$ indica la norma L_2 al cuadrado,
- $\text{sg}[\cdot]$ es el operador *stop-gradient*, que **no propaga gradientes** hacia $\hat{Z}$.

El propósito del operador *stop-gradient* es asegurarse de que las actualizaciones de gradiente que este término produce solo afecten al *encoder* y no modifiquen el diccionario.

4.3. Cuantización

Dos estrategias de cuantización son utilizadas en la implementación de la VQ-VAE de poses: EMA y *code reset*.

4.3.1. EMA

Exponential Moving Average (EMA) (van den Oord et al. 2017) es una técnica mediante la cual los vectores del *codebook* son actualizados aplicando una media móvil exponencial de las salidas del *encoder*. Esto evita utilizar los gradientes para actualizar el *codebook*, resultando en un entrenamiento más estable.

Para cada vector e_i del *codebook* C , se mantienen:

- n_i : la media móvil exponencial del número de veces que el código i ha sido el vecino más cercano (es decir, seleccionado durante la cuantización).
- m_i : la media móvil exponencial de la suma de las salidas del *encoder* asignadas al código i .

Los conteos y las sumas son actualizados de la siguiente manera:

$$n_i^{(t)} = \gamma \cdot n_i^{(t-1)} + (1 - \gamma) \cdot n_i^{\text{actual}} \quad (4.1)$$

$$m_i^{(t)} = \gamma \cdot m_i^{(t-1)} + (1 - \gamma) \cdot m_i^{\text{actual}} \quad (4.2)$$

donde γ es la tasa de decaimiento, y n_i^{actual} y m_i^{actual} son los conteos y las sumas del batch actual.

El vector e_i del *codebook* es finalmente actualizado mediante:

$$e_i = \frac{m_i}{n_i} \quad (4.3)$$

4.3.2. Code reset

El objetivo principal es prevenir problemas como el colapso del *codebook*, en el cual algunos códigos se vuelven inutilizados o subutilizados. Para ello, se *resetea* el vector correspondiente a aquellos códigos que presentan baja frecuencia de uso durante el entrenamiento, favoreciendo que todos los códigos contribuyan activamente al aprendizaje. Concretamente, en la implementación utilizada en este proyecto, cuando se detecta que un código no ha sido utilizado, se le asigna un nuevo vector aleatorio. Este vector se genera a partir de la distribución del *batch* de vectores de poses en ese paso de entrenamiento.

4.4. Arquitectura

Como se observa en la Figura 4.2, se usa una arquitectura convolucional compuesta por convoluciones 1D, bloques residuales (He et al. 2015), y activaciones ReLU para introducir no linealidad. Los bloques residuales también están compuestos por convoluciones 1D y activaciones ReLU.

Se utiliza convolución con *stride* 2 para el *downsampling temporal*, e interpolación por vecinos más cercanos para el *upsampling temporal*.

La *tasa de downsampling* es $l = 2^L$, siendo L el número de bloques residuales.

4.5. Detalles de implementación

Se configuraron los siguientes hiperparámetros en base a los trabajos que se tomaron como referencia:

- Tamaño del *codebook*: 512 x 512.
- Tamaño de *batch*: 256.
- Optimizador *AdamW* con $[\beta_1, \beta_2] = [0.9, 0.99]$.
- Tasa de decaimiento para EMA, $\gamma = 0.99$.
- 1000 pasos iniciales de *warm-up* con *learning rate* inicial igual a 2×10^{-4} y paso de actualización: $\text{lr}^{\text{actual}} = \text{lr}^{\text{prev}} \times \frac{(\#\text{iteraciones}+1)}{(\#\text{paso}+1)}$.
- Luego del *warm-up* se ejecutan 50000 iteraciones sin modificación del *learning rate*.

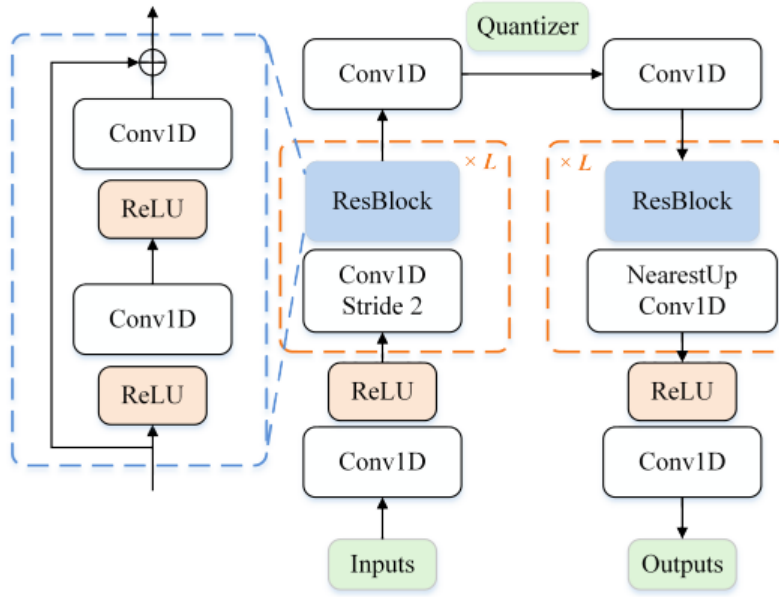


Figura 4.2: Arquitectura de la VQ-VAE de poses (J. Zhang et al. 2023).

Cada pose es transformada a un *embedding* de dimensión 274, correspondiente a las dimensiones 2D de los 137 *keypoints* de Openpose, siendo esta la dimensión de entrada del *encoder*.

4.6. Dataset

El *dataset* se generó extrayendo las poses del *dataset* de Ham2Pose, ver Sección 2.6.1, aplicando el procedimiento descrito en la Sección 2.6.2. Los 5749 ejemplos del *dataset* original se dividieron en particiones de entrenamiento (81 %), validación (9 %) y testeo (10 %). La distribución por lengua dentro cada partición mantiene la distribución del conjunto total (ver Tabla 4.1).

Tabla 4.1: Composición por lengua para cada partición del *dataset* utilizado para entrenar la red VQ-VAE.

	pjm	dgs	gsl	lsf
entrenamiento	44,5 %	33 %	15,9 %	6,6 %
testeo	44,7 %	35 %	13,9 %	6,4 %
validación	45,6 %	33,7 %	13,6 %	7,1 %

4.7. Resultados

Con el propósito de comparar con un enfoque más sencillo, se implementó un método *baseline* para la codificación de poses. En particular, se empleó el algoritmo de *clustering* K-Means con una tolerancia de 1×10^{-4} y un máximo de 300 iteraciones. Cada pose es asignada al centroide del *cluster* más cercano, el cual se considera como la versión reconstruida de la pose original. Esta estrategia permite calcular el *loss de reconstrucción* así como también *nDTW-MJE*, y de esta forma comparar con VQ-VAE. El número de *clusters* utilizado se corresponde al tamaño del *codebook* de la red VQ-VAE (512).

Para calcular *nDTW-MJE* se siguió la implementación de Ham2Pose (Shalev-Arkushin et al. 2023). Para el caso de VQ-VAE, las métricas reportadas se corresponden a aquellas obtenidas en la iteración en la cual se alcanzó el valor más bajo de *nDTW-MJE* (para el conjunto de validación).

En la Tabla 4.2 se observa la comparación entre los modelos entrenados con K-Means y VQ-VAE. Para que la comparación sea “justa” se reportan los resultados de las redes VQ-VAE entrenadas con *tasa de downsampling* igual a 1, lo que es equivalente a no aplicar *downsampling*. Se observan mejores resultados para VQ-VAE, ya que se obtienen valores más bajos tanto para el *loss de reconstrucción* como para el error *nDTW-MJE*.

Tabla 4.2: K-Means vs VQ-VAE

Largo de Secuencia	Modelo	Recons ↓	nDTW-MJE ↓
32	K-Means	0.00241	10.28499
	VQ-VAE	0.00188	8.85521
64	K-Means	0.00224	9.66192
	VQ-VAE	0.00188	8.22592

Se realizaron experimentos con valores 32 y 64 para el largo de secuencias de poses (se truncan las secuencias mayores, y se descartan las secuencias menores), y con valores 1, 2 y 4 para la *tasa de downsampling* (L).

Como se reporta en la Tabla 4.3, los mejores valores de *nDTW-MJE* se obtienen en los modelos entrenados con *tasa de downsampling* igual a 2.

En la Tabla 4.4, para los modelos entrenados con *tasa de downsampling* igual a 2, se detallan los valores correspondientes a $\mathcal{L}_{re}$, $\mathcal{L}_{commit}$, PPL y *nDTW-MJE*. En este contexto, PPL sirve para medir qué tan eficazmente se está utilizando el *codebook*. Un valor más alto de PPL indica una distribución más

Tabla 4.3: $nDTW-MJE$ para diferentes *tasas de downsampling*.

Largo de Secuencia	Tasa de downsampling	nDTW-MJE ↓
32	1	8.85521
	2	8.13396
	4	8.68896
64	1	8.22592
	2	7.73611
	4	8.24214

uniforme en el uso de los códigos, lo que significa que el modelo está utilizando una mayor parte del diccionario de códigos. Se obtienen métricas similares para las particiones validación y testeo (ver Figura 4.5).

Tabla 4.4: Resultados para modelos entrenados con *tasa de downsampling* igual a 2.

Largo de Secuencias	Iteración	Commit ↓	PPL ↓	Recons ↓	nDTW-MJE ↓
32	15000	0.02919	363.73	0.00179	8.13396
64	7000	0.02157	344.73	0.00186	7.73611

Tabla 4.5: Métricas de validación y testeo, para modelos entrenados con *tasa de downsampling* igual a 2.

Largo de Secuencias	Validación		Testeo	
	Recons ↓	nDTW-MJE ↓	Recons ↓	nDTW-MJE ↓
32	0.00179	8.13396	0.00168	7.99319
64	0.00186	7.73611	0.00170	7.73941

Los modelos que se entrenan con secuencias de largo 64 tienen una cobertura del 40 % del *dataset* original, un valor significativamente menor al 94.5 % de cobertura que tienen aquellos modelos que descartan solamente secuencias de largo menor a 32. Es por este motivo que a pesar de que el modelo con largo de secuencias 64 presenta un valor de $nDTW-MJE$ más bajo, se opta por elegir al modelo que utiliza largos de secuencia 32. Cabe destacar que este último reporta el error de reconstrucción más bajo.

Al analizar en detalle la evolución de las métricas (ver Figura 4.3) durante el entrenamiento del modelo seleccionado, se observa que tanto $\mathcal{L}_{re}$, PPL como $nDTW-MJE$, mejoran consistentemente hasta un número de iteraciones

similar, para luego estabilizarse. En cambio $\mathcal{L}_{commit}$ sigue mejorando de forma continua durante las 50000 iteraciones.

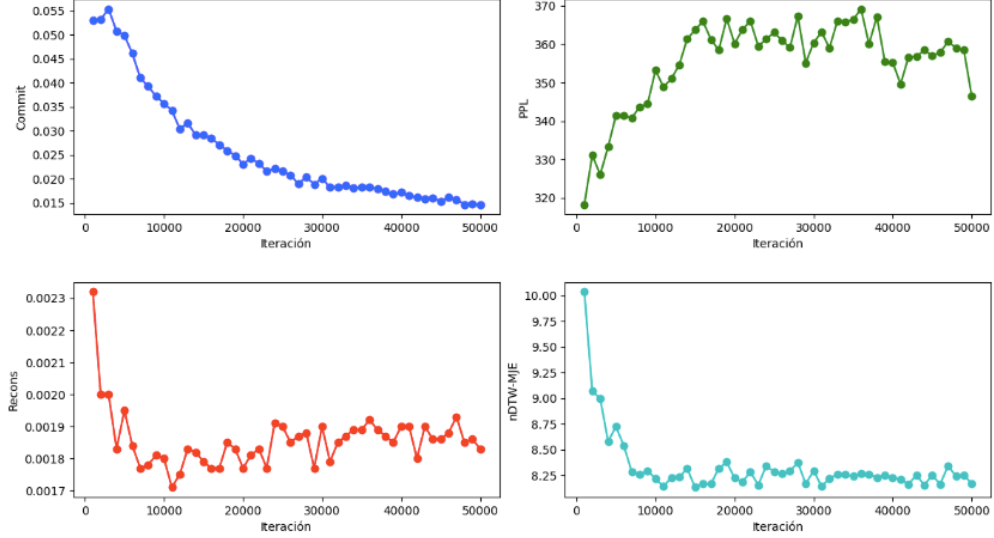


Figura 4.3: Evolución de $\mathcal{L}_{commit}$, PPL, $\mathcal{L}_{re}$ y $nDTW-MJE$ durante el entrenamiento de la red VQ-VAE.

4.8. Conclusiones

Los resultados de reconstrucción alcanzados se consideran aceptables para poder utilizar la VQ-VAE implementada en el contexto de este proyecto. Como trabajo futuro se puede realizar una mayor exploración de hiperparámetros, experimentando con diferentes tamaños del *codebook*, probando diversas combinaciones de las técnicas de cuantización, así como también realizando ajustes en las arquitecturas convolucionales elegidas. Inclusive se puede experimentar con múltiples *codebooks*, siguiendo la implementación de (Xie et al. 2022) donde se utilizan 3 *codebooks* independientes: uno para la mano derecha, otro para la mano izquierda, y el tercero para el resto de los *keypoints*.

Capítulo 5

Generador de descripciones HamNoSys

En este capítulo se presenta detalladamente el proceso de generación de descripciones de secuencias HamNoSys, utilizadas posteriormente para el entrenamiento e inferencia del LLM especializado en la generación de poses de lengua de señas.

Inicialmente, se ofrece una explicación sobre los bloques y sub-bloques que componen una secuencia HamNoSys, complementada con ejemplos y visualizaciones para facilitar su comprensión.

A continuación, se profundiza en las complejidades inherentes a la tarea de generar descripciones en lenguaje natural de los símbolos de HamNoSys, proporcionando ejemplos específicos que ilustran estas dificultades. Las explicaciones y ejemplos incluidos en este capítulo se han extraído de diferentes recursos disponibles, destacándose la publicación de la Universidad de Hamburgo donde se presenta la notación HamNoSys (Hanke, 2004), la guía de usuario oficial de HamNoSys 4.0 (Smith, 2013) y la introducción a HamNoSys 4.0 elaborada por el Grupo de Tecnología del Habla de la Universidad Politécnica de Madrid (Universidad Politécnica de Madrid, 2010).

Seguidamente, se examina la fuente Unicode⁵ de HamNoSys y se describe el preprocesamiento de los datos llevado a cabo antes de la generación de las descripciones.

Finalmente, se detallan las cuatro modalidades implementadas para la generación de estas descripciones, resaltando sus características y diferencias.

⁵<https://home.unicode.org/>

5.1. Estructura de la notación HamNoSys

La estructura de una secuencia de símbolos HamNoSys se puede dividir en 4 bloques principales (ver Figura 5.1), siendo algunos de ellos opcionales:

- **Operador de simetría:** denota señas con dos manos y determina cómo se deben espejar los atributos en la mano no dominante, a menos que se especifique lo contrario. Es opcional.
- **NMF (Características no manuales):** representa características no manuales que pueden utilizarse para describir una seña dada. Es opcional.
- **Configuración inicial de la mano:** representa la posición y forma de la mano de la postura inicial.
- **Movimiento:** representa una combinación de movimientos de trayectoria que pueden especificarse como movimientos absolutos o relativos.

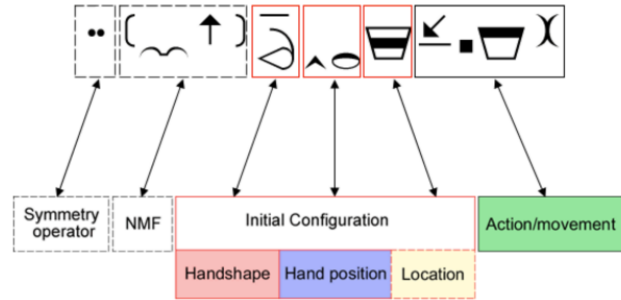


Figura 5.1: Estructura de una secuencia HamNoSys (Skobov y Lepage, 2020).

Por ejemplo, si analizamos la secuencia ($\sim \square \tau \ominus^{\chi} \succ$), ver Figura 5.2, los bloques presentes pueden interpretarse de la siguiente manera:

- **Operador de simetría ($\sim$):** indica que es una seña con las dos manos, con simetría con respecto al eje vertical.
- **Configuración inicial de la mano ($\square \tau \ominus$):** indica que la mano derecha está abierta ($\square$), orientada hacia arriba con una inclinación de 45° hacia la izquierda (τ), con la palma hacia abajo ($\ominus$). La mano izquierda se encuentra en posición simétrica con respecto al eje vertical.
- **Movimiento ($^{\chi} \succ$):** las manos comienzan tocándose ($^{\chi}$) y luego se desplazan diagonalmente hacia abajo ($\succ$).

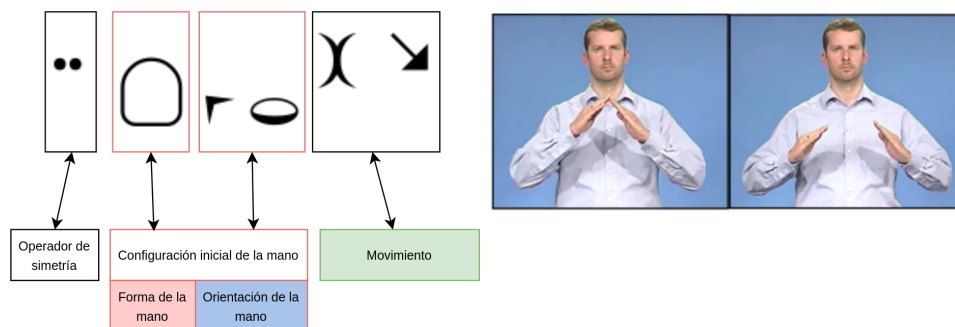


Figura 5.2: Secuencia HamNoSys acompañada de la seña real.

A continuación se describen de forma resumida cada uno de los bloques, salvo las características no manuales, que no son utilizadas en el presente trabajo.

5.1.1. Operador de simetría

El operador de simetría se utiliza para representar señas que involucran ambas manos. Determina que los atributos de la mano dominante deben reflejarse en la mano no dominante.

La simetría puede ser respecto al eje horizontal ($\cdot$) o respecto al eje vertical ($\cdot$).

Las excepciones siempre pueden especificarse describiendo por separado las configuraciones o acciones para cada mano siguiendo la siguiente estructura:

[**mano dominante** $\times$ **mano no dominante**]



Figura 5.3: Ejemplo de seña con dos manos que incluye excepciones.

5.1.2. Configuración inicial de la mano

La *configuración inicial de la mano* está compuesta por tres sub-bloques: *forma de la mano*, *orientación de la mano* y *localización*.

Forma de la mano

La *forma de la mano* se especifica en base a 3 componentes: la *forma base* (abierta, cerrada, en forma de C), la *inclinación de los dedos* y la *posición del dedo pulgar*, como se puede observar en las figuras 5.4, 5.5 y 5.6 respectivamente (Smith, 2013).

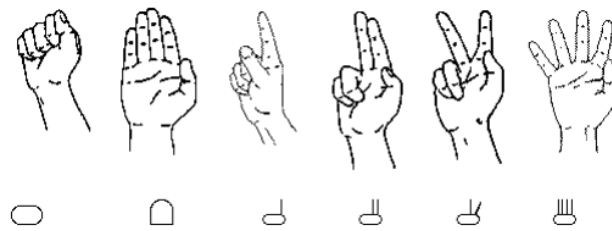


Figura 5.4: Formas básicas de la mano.



Figura 5.5: Forma base combinada con inclinaciones de los dedos.

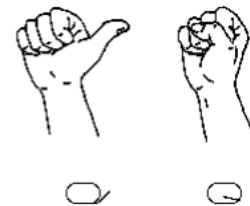


Figura 5.6: Forma base combinada con posiciones del dedo pulgar.

Orientación de la mano

La *orientación de la mano* está definida por dos parámetros:

- **Dirección de la base del dedo índice:** la dirección a la que estaría apuntando el dedo índice si estuviera extendido (dirección del eje).
- **Orientación de la palma de la mano:** orientación de la mano según el eje.

Por ejemplo, la *orientación de la mano* representada por los símbolos $\wedge 0$ indica la dirección del eje de la mano hacia arriba ($\wedge$), con la *orientación de la palma* de la mano hacia la izquierda (0).

Localización

La *localización* es opcional, y se define en base a dos componentes: la *parte del cuerpo* en dónde se sitúa la mano (ver Figura 5.7) y la *distancia* con respecto a dicha parte del cuerpo.

	A la izquierda	En la parte izquierda	Centro	En la parte derecha	A la derecha
Cabeza	◻○	▪○	○	○▪	○◻

Figura 5.7: Ubicaciones con respecto a la cabeza.

Los símbolos de *distancia* indican si se encuentra detrás del cuerpo ($\supset$), si se encuentra en contacto con una parte del cuerpo ($\supset$), o si en cambio se encuentra cerca a una distancia estándar ($\supset$) o máxima ($\supset$).

En la Figura 5.8 se muestra una seña con la mano localizada en la frente ($\supset$), tocando ($\supset$) el dedo pulgar ($\supset$).



Figura 5.8: Sub-bloque de *localización*: $\supset(\supset \supset)$

5.1.3. Movimiento

La descripción de los movimientos puede ser bastante compleja, incluyendo movimientos de las manos en el espacio, o movimientos de las manos sin cambiar su posición (por ej: flexión de los dedos), como también movimientos no manuales.

Los símbolos pueden describir, por ejemplo:

- Trayectorias en líneas rectas, curvas, zig zags, o círculos.
- Cambios que modifican la forma u orientación de la mano.
- Repeticiones de acciones.
- Acciones secuenciales o en paralelo.
- Modo del movimiento: rápido o lento.

En la Figura 5.9 se representa un movimiento hacia abajo ($\downarrow$) que se repite varias veces ($\uparrow$) mientras la mano se desplaza hacia la derecha ($\rightarrow$).

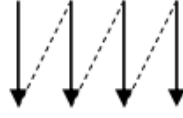


Figura 5.9: Bloque de *movimiento*: $\downarrow(\uparrow\rightarrow)$.

5.2. Complejidades de la notación HamNoSys

Implementar una transcripción en lenguaje natural que explique cada símbolo de la notación HamNoSys de forma concisa presenta algunas dificultades:

- La complejidad y abstracción de los símbolos, que representan configuraciones de mano o movimientos específicos, puede hacer difícil resumir su significado sin perder detalles esenciales.
- La cantidad y diversidad de símbolos requiere un profundo entendimiento de los mismos, lo que puede ser un proceso laborioso.
- La interdependencia de los símbolos, donde algunos adquieren pleno significado sólo en combinación con otros, complica su explicación cuando se describen de forma aislada o se pretenden generar descripciones únicas.
- La falta de recursos estandarizados, materiales didácticos, y consensos a la hora del etiquetado, dificulta la creación de definiciones universalmente aceptadas para cada símbolo.

A continuación se detallan algunos ejemplos de cambios en los significados de los símbolos en base a su contexto.

El símbolo $\setminus$ puede ser utilizado para representar configuraciones intermedias. Por ejemplo, puede indicar una forma intermedia de la mano en base a otras 2 formas dadas (ver Figura 5.10), que no estará explícitamente mencionada en la descripción en lenguaje natural.

Los dedos de la mano están numerados del 1 (pulgar) al 5 (meñique). La presencia del símbolo de un dedo puede modificar el significado de un símbolo adyacente. Por ejemplo, el símbolo $\hookleftarrow$ representa la mano con el dedo índice extendido por defecto, pero si está junto al símbolo del dedo meñique implica una seña con este último dedo extendido, como se muestra en la Figura 5.11.

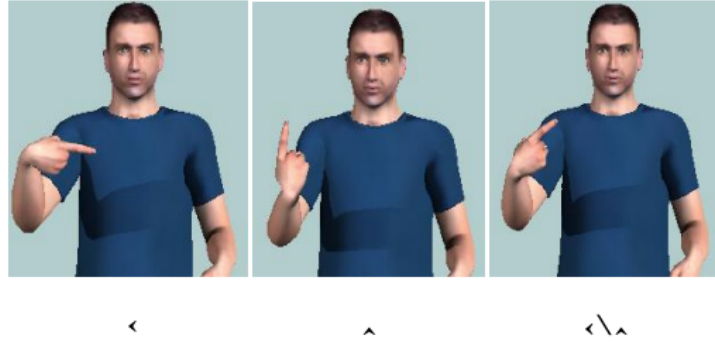


Figura 5.10: Posición intermedia entre dos formas de la mano.

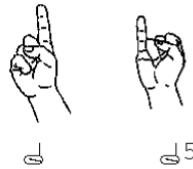


Figura 5.11: Efectos de usar especificaciones sobre dedos de la mano.

El símbolo de localización $\square$ cambia su significado dependiendo de su ubicación relativa con respecto a una parte del cuerpo. Por ejemplo, puede indicar una ubicación a la izquierda de la nariz ($\square \text{ 鼻}$) o a su derecha ($\text{鼻} \square$).

Por otra parte, el símbolo $\times$ puede representar por defecto manos cruzadas, pero también se puede usar en conjunto con una parte del cuerpo específica para indicar brazos, muñecas o dedos cruzados.

Asimismo, dependiendo de la dirección del eje de la mano, la interpretación del símbolo correspondiente a la palma puede ser diferente:

- $\triangle \circ$: dirección del eje de la mano hacia adelante, palma hacia abajo.
- $\wedge \circ$: dirección del eje de la mano hacia arriba, palma de frente.

5.3. Preprocesamiento de los datos

Los símbolos de HamNoSys están disponibles como una fuente Unicode. Cada glifo cuenta con un nombre oficial, un código Unicode y un símbolo correspondiente en la fuente.

Casi todos los caracteres de la fuente HamNoSys se encuentran en el área de uso privado de Unicode. Este es un grupo de caracteres para los cuales intencionalmente no se proporcionan significados, reservándolos para casos de

uso especiales que no están cubiertos por Unicode, como HamNoSys. Hay un pequeño número de caracteres relevantes para HamNoSys que se encuentran en las áreas regulares de Unicode, como el signo de interrogación (Schulder y Hanke, 2022).

En la tabla 5.1 se muestran algunos ejemplo para símbolos correspondientes a partes del cuerpo.

Tabla 5.1: Ejemplos de fuente HamNoSys para partes del cuerpo

Parte del cuerpo	Nombre	Símbolo	Código
cejas	hameyebrows	~	E043
ojos	hameyes	∞	E044
nariz	hamnose	‡	E045
fosas nasales	hamnostrils	‡	E046
oreja	hamear	?	E047

El dataset de Ham2Pose utilizado para este proyecto provee la secuencia HamNoSys como una lista de nombres HamNoSys y/o como una lista de códigos Unicode, por ejemplo para un dato en particular se tiene:

- **códigos unicode:** e006, e027, e03e, e04a, e0e2, e08c, e0c6, e0aa, e009, e010, e00e, e0e3, e0d0, e0d8
- **nombres HamNoSys:** hampinch12, hamextfingerul, hampalml, hamlips, hamparbegin, hammovei, hamsmallmod, hamreplace, hamcee12, hamfingerstraightmod, hamthumbopenmod, hamparend, hamclose, hamrepeatfromstart

Asimismo, los nombres HamNoSys pueden contener números escritos utilizando dígitos (ej: hamfinger2) o utilizando las palabras equivalentes en inglés (ej: hamfingertwo).

Por lo tanto se decide implementar como parte de este trabajo, un proceso previo de normalización de las secuencias, para convertirlas a secuencias de nombres HamNoSys que incluyen los números en forma de dígitos. Se sigue el siguiente proceso:

- Si un dato tiene la secuencia únicamente en formato Unicode, se utiliza un diccionario para mapear los códigos a nombres HamNoSys, en caso contrario se prioriza la secuencia en forma de nombres.

- Para cada nombre de la secuencia HamNoSys se verifica si contiene los números escritos con palabras, y en caso afirmativo se lo sustituyen con dígitos.

Los diccionarios implementados para realizar estos mapeos fueron generados utilizando la fuente Unicode de HamNoSys disponible en el paquete *HamNoSys4TeX*⁶.

5.4. Modalidades de descripciones

A continuación se describen las 4 modalidades de generación de descripciones HamNoSys que fueron implementadas como parte de esta tesis: *secuencia HamNoSys*, *secuencia HamNoSys sin prefijo*, *secuencia con descripciones*, *secuencia con descripciones refinadas* y *descripción en lenguaje natural*.

En la Tabla 5.2 se muestra la descripción generada, para cada una de las modalidades, para la secuencia $\prec \wedge 0 \succ \{ \text{X} \} \rightarrow \text{J}^{\text{H}}$. El video de la seña correspondiente a la secuencia se ilustra en la Figura 5.12.

Tabla 5.2: Descripciones según modalidad para la secuencia $\prec \wedge 0 \succ \{ \text{X} \} \rightarrow \text{J}^{\text{H}}$

Modalidad	Descripción generada
<i>Secuencia HamNoSys</i>	hampinch12, hamextfingeru, hampalml, hamcheek, hamseqbegin, hamtouch, hamfingertip, hamseqend, hamreplace, hamfinger2, hamthumbopenmod, hamrepeatfromstartseveral
<i>Secuencia HamNoSys sin prefijo</i>	pinch12, extfingeru, palml, cheek, seqbegin, touch, fingertip, seqend, replace, finger2, thumbopenmod, repeatfromstartseveral

⁶<https://github.com/DGS-Korpus/HamNoSys4TeX>

<i>Secuencia con descripciones</i>	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
------------------------------------	--

<i>Secuencia con descripciones refinadas</i>	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
--	---

<i>Descripción en lenguaje natural</i>	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.
--	--



Figura 5.12: *Frames* del video original para la secuencia $\odot \wedge 0 \} (\chi \uparrow) \rightarrow \odot^{\dagger}$

5.4.1. Secuencia HamNoSys

Esta modalidad *naive* puede utilizarse como línea base (*baseline*). Consiste simplemente en tomar la secuencia de nombres HamNoSys, separados por comas, obtenida tras aplicar la normalización previamente definida.

Esta modalidad permite evaluar el conocimiento previo que los grandes modelos de lenguaje puedan tener sobre HamNoSys, en base a los datos con los que fueron entrenados. A su vez, aprovecha el contenido semántico (aunque reducido) que contienen los nombres oficiales definidos por la notación.

Tabla 5.3: Ejemplo utilizando modalidad *secuencia HamNoSys*

Símbolos	Secuencia HamNoSys.
$\odot \wedge 0 \} (\chi \uparrow) \rightarrow \odot^{\dagger}$	hamfist, hamthumbacrossmod, hamextfingeru, hampalml, hamcheek, hamtouch, hammoved, hamsmallmod, hamarcr, hamchin, hamrat

5.4.2. Secuencia HamNoSys sin prefijo

Esta modalidad consiste en eliminar el prefijo *ham* de los nombres HamNoSys.

El objetivo de esta simplificación es evitar que el prefijo, que puede no tener un significado relevante para modelos de lenguaje que no han sido entrenados con datos de este dominio específico, introduzca ruido o confusión. Al eliminar

el prefijo, las secuencias resultantes contienen únicamente los extractos con verdadero contenido semántico.

Tabla 5.4: Ejemplo utilizando modalidad *secuencia HamNoSys sin prefijo*.

Símbolos	Secuencia HamNoSys sin prefijo
$\ominus \wedge 0 \} \chi \downarrow \mathfrak{D} \cup \blacksquare$	fist, thumbacrossmod, extfingeru, palml, cheek, touch, moved, smallmod, arcr, chin, lrat

5.4.3. Secuencia con descripciones

Esta modalidad consiste en sustituir cada símbolo de la secuencia HamNoSys por una descripción apropiada en lenguaje natural, en el idioma inglés.

Tabla 5.5: Ejemplo utilizando modalidad *secuencia con descripciones*.

Símbolos	Secuencia con descripciones
$\ominus \wedge 0 \} \chi \downarrow \mathfrak{D} \cup \blacksquare$	fist thumb across extended finger up (parallel to body) palm left cheek contact with straight movement down (parallel to body) small movement curved movement away from body curved movement to the right chin right of to the right of left of to the left of

Ante la ausencia de recursos específicos que proporcionen estas descripciones, en esta tesis se ha llevado a cabo una revisión de la documentación y otros materiales disponibles para recopilar definiciones y ejemplos que faciliten la comprensión de todos los posibles significados de cada símbolo. A continuación se presenta un resumen de los hallazgos obtenidos, en conjunto con ejemplos de las descripciones generadas para determinados símbolos.

Algunos símbolos presentan significados directos y concisos, como partes específicas del cuerpo (ver Tabla 5.6).

Tabla 5.6: Descripciones simples.

Símbolo	Descripción
~	eyebrows
∞	eyes
⊥	nose
⊥	nostrils
∪	ear

Otros símbolos, en cambio, requieren descripciones más detalladas, que incluyen aclaraciones o comentarios adicionales (ver Tabla 5.7).

Tabla 5.7: Descripciones con aclaraciones.

Símbolo	Descripción
◁	closed pinch between thumb and index - rest of the fingers in fist form
↗	extended finger up and to the right (parallel to body)
↗	repeated once (2nd iteration starting when first ended)
↔	two-handed symmetrical sign (hands position mirror one another)

Para símbolos con significados que varían considerablemente según el contexto, se proporcionan las diferentes acepciones aplicables, como se muestra en la Tabla 5.8.

Tabla 5.8: Descripciones con múltiples significados.

Símbolo	Descripción
◻	left left side of right right side of
▪	right of to the right of left of to the left of
↗	finger movements resulting from handshape change change of handshape and orientation
↺	curved movement towards body curved movement to the left
×	crossing hands (also for arms - wrists and fingers)

Para símbolos como los signos de puntuación, que intrínsecamente ya aportan significado semántico, igualmente se decide reemplazarlos con descripciones en lenguaje natural.

Tabla 5.9: Descripciones de símbolos de signos de puntuación.

Símbolo	Descripción
{	alternatives begin
}	alternatives end
	either/or

5.4.4. Secuencia con descripciones refinadas

Esta modalidad introduce algunos ajustes a la *secuencia con descripciones refinadas* descrita anteriormente.

Tabla 5.10: Ejemplo utilizando modalidad *secuencia con descripciones refinadas*.

Símbolos	Secuencia con descripciones refinadas
☞ ^ 0 } X ↓ 2 ∪ ▪	fist thumb across extended finger up - parallel to body palm left cheek contact with straight movement down - parallel to body small movement curved movement away from body or curved movement to the right chin adjacent

En primer lugar, para aquellos símbolos cuyo significado puede variar sustancialmente de acuerdo al contexto, hasta el punto de llegar a ser “contradictorios”, se evita detallar cada posible connotación. En su lugar, se genera una descripción más concisa que contenga un contenido semántico común a todos los significados posibles. El objetivo es evitar introducir “ruido” y dar lugar

a que el LLM capte las sutilezas dependientes del contexto a medida que se entrena con diferentes ejemplos.

Tabla 5.11: Descripciones con múltiples significados refinadas.

Símbolo	Descripción	Descripción refinada
◻	left left side of right right side of	side
▪	right of to the right of left of to the left of	adjacent
×	crossing hands (also for arms - wrists and fingers)	crossing

Sin embargo, en algunos casos, no es posible limitarse a un contenido semántico común, ya que esto podría causar confusión con otros símbolos similares, como se muestra en la Tabla 5.12.

Tabla 5.12: Descripciones con múltiples significados, conservadas.

Símbolo	Descripción refinada
◁	curved movement towards body or curved movement to the left
⤴	curved movement upwards or curved movement away from body
▷	curved movement away from body or curved movement to the right
⤵	curved movement downwards or curved movement towards body

Además, se opta por emplear directamente aquellos símbolos Unicode correspondientes a signos de puntuación que aportan un valor semántico propio. Por ejemplo, se eliminan las menciones a *begin* y *end*, reemplazándolas por paréntesis de apertura y cierre; y se decide usar directamente el símbolo | en lugar de la descripción *either/or*.

Tabla 5.13: Descripciones refinadas con signos de puntuación.

Símbolo	Descripción	Descripción refinada
	either/or	
[	co-temporal action begin	co-temporal action [
]	co-temporal action end	]
{	alternatives begin	alternatives {
}	alternatives end	}

Este último ajuste, para prevenir confusiones, conlleva a evitar el uso de dichos signos de puntuación dentro de otras descripciones.

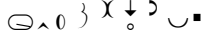
Tabla 5.14: Descripciones refinadas sin el uso de determinados signos.

Símbolo	Descripción	Descripción refinada
“	two-handed simmetrical sign (hands position mirror one another)	two-handed simmetrical sign - hands position mirror one another
↶	curved movement towards body curved movement to the left	curved movement towards body or curved movement to the left

5.4.5. Descripción en lenguaje natural

La última modalidad implementada no consiste en la generación de una secuencia, sino en una descripción compacta en lenguaje natural que interpreta y explica la secuencia de entrada.

Tabla 5.15: Ejemplo utilizando modalidad *descripción en lenguaje natural*.

Símbolos	Descripción en lenguaje natural
	The sign begins with a closed fist shape with the thumb placed across the fingers, the extended finger pointing upwards in line with the body. The palm of the hand is facing left, making contact with the cheek. The hand then moves straight down parallel to the body with a small modifier movement, followed by a curved movement either away from the body or to the right. The final movement is towards the chin, with a modifier indicating the position is adjacent.

Estas descripciones son generadas utilizando el modelo de lenguaje GPT-3.5-turbo de OpenAI, a partir de secuencias estructuradas previamente bajo la modalidad denominada *secuencia con descripciones refinadas*.

Para mejorar el contexto y la claridad, cada descripción refinada de un símbolo se acompaña de una etiqueta que especifica el *grupo de símbolos* al que pertenece el mismo.

A continuación, se detallan las instrucciones proporcionadas al modelo de OpenAI así como los grupos de símbolos definidos.

Estructura de las *prompts*

El modelo de OpenAI es invocado utilizando una *prompt* con el rol *sistema* y otra con el rol *usuario*. La *prompt* de tipo *sistema* es concisa, “*You are an expert in Sign Language and in HamNoSys notation.*”, y tiene por objetivo definir en alto nivel el rol asignado al asistente. Por su parte, la *prompt* de tipo *usuario* contiene la instrucción específica que guía al asistente para realizar la tarea de transformar una secuencia HamNoSys en una descripción en lenguaje natural.

La *prompt* de tipo *usuario* está compuesta por:

1. Una descripción detallada de los bloques y sub-bloques que conforman la estructura de una secuencia HamNoSys, proporcionando contexto al mo-

delo de lenguaje para interpretar adecuadamente la secuencia de entrada:

The structure of HamNoSys sequences used in this project has three main blocks with the following order:

“Symmetry block”, “Initial configuration block” and “Actions block”.

Symmetry block (optional): denotes two-handed signs and determines how attributes should be mirrored to the non-dominant hand unless otherwise specified. . If some of the parameters between the two hands are different, the HamNoSys description has the following structure:

.....

.....

Action block: represents a combination of path movements.

-
2. Una explicación de la tarea que debe realizar el asistente:

Given a HamNoSys sequence (some elements of the sequence are labeled to help understand their utility in the sequence), try to rewrite the sequence in a coherent paragraph that explains in natural language the description of the sign.

3. Un ejemplo de una secuencia HamNoSys en conjunto con su descripción en lenguaje natural, siguiendo el enfoque *few-shot learning*, que consiste en proporcionar ejemplos de entrada y salida al asistente, de forma de ayudarlo a generalizar para nuevas entradas.

For example:

Hamnosys sequence:

“two-handed symmetrical sign - hands position mirror one another” “flat hand (Handshape - Base from)” “extended finger up and to the left -

parallel to body (Hand position - Extended finger direction)” “palm down (Hand position - Palm orientation)” “contact with (Location - Distance)” “straight movement down and to the right - parallel to body (Movement)”

NLP description:

Two flat hands with fingers closed, rotated towards each other, touching, then symmetrically moving diagonally downwards.

-
4. La nueva secuencia HamNoSys y la pauta para generar la descripción:
-

For the Hamnosys sequence:

“flat hand (Handshape - Base from)” “thumb across (Handshape - Thumb position)” “extended finger up - parallel to body (Hand position - Extended finger direction)” “palm left (Hand position - Palm orientation)” “cheek (Location - Body part)” “contact with (Location - Distance)”

NLP description:

Grupos de símbolos

Los símbolos se agruparon de acuerdo al bloque o sub-bloque al que pertenecen, y de acuerdo a su funcionalidad dentro de los mismos. Para algunos símbolos en particular no se incluye etiqueta, por ejemplo símbolos que definen comienzos o fines de secuencias. En la Tabla 5.16 se listan los grupos definidos acompañados de un ejemplo representativo.

Tabla 5.16: Grupos de símbolos definidos.

Grupo	Ejemplo
Handshape - Base from	flat hand: $\square$
Handshape - Thumb position	thumb opened outward: $\swarrow$

- Los símbolos relacionados con la forma de la mano ($\text{𐤎} \setminus \text{𐤎}$), contienen al símbolo *between* ($\setminus$) que indica una posición intermedia entre dos formas específicas. La interpretación correcta de esta descripción sería una mano con el pulgar extendido y los otros cuatro dedos separados, con una flexión intermedia (ni completamente plana ni muy pronunciada). Sin embargo, la descripción generada por el LLM interpreta erróneamente esta información, considerándola como una seña de dos manos en lugar de una sola, y asumiendo que el símbolo *between* se refiere a la colocación de los dedos entre ambas manos, lo que altera el significado original.
- Algo similar ocurre con los símbolos que describen la orientación del eje de la mano ($\text{𐤎} \setminus \text{𐤎}$), el símbolo *between* debería indicar una posición intermedia, pero el LLM lo interpreta nuevamente como una referencia a dos manos distintas.
- En cuanto a la ubicación de la mano con respecto a la parte superior de la cabeza (𐤎), la descripción generada menciona que está “a un lado”, sin especificar que es a la derecha. Esta omisión se debe a que las descripciones refinadas de los símbolos utilizadas en la generación de la prompt no incluyen explícitamente este detalle.

Para mejorar la precisión de las descripciones en lenguaje natural, se podría implementar a futuro un *parseo* previo de la secuencia HamNoSys. Este preprocesamiento permitiría construir una *prompt* más precisa asignando descripciones dinámicas a los símbolos según su contexto.

Por ejemplo, el símbolo 𐤎 podría reemplazarse explícitamente por “a la izquierda”, cuando efectivamente indique dicha posición en relación con una parte del cuerpo, como en el caso de ($\text{𐤎} \text{𐤎}$).

Asimismo, al detectar la presencia del símbolo *between*, se podrían identificar las dos posiciones vinculadas y especificar en la *prompt* que se trata de una única posición intermedia entre ambas.

Capítulo 6

LLM

En este capítulo se detalla la implementación del ajuste del LLM para la generación de *tokens* representativos de poses de lengua de señas. En primer lugar, se presenta el modelo base utilizado, *Mistral-7B-Instruct-v0.2*, describiendo tanto sus características principales como el formato de sus instrucciones. A continuación, se explican las plantillas de instrucciones definidas específicamente para este trabajo.

Posteriormente se aborda la composición del conjunto de datos utilizado para el ajuste del modelo, seguido de una descripción de los detalles del entrenamiento mediante la técnica LoRA, junto con los parámetros básicos configurados. Además, se introduce la técnica de posprocesamiento utilizada después de cada inferencia para la extracción de las secuencias de *tokens*.

Más adelante, se describen los experimentos realizados, incluyendo el análisis la evolución de la métrica *nDTW-MJE* para cada *checkpoint* generado durante los entrenamientos.

Por último, se discuten los resultados obtenidos basados en la métrica *Distance Ranks*, comparándolos con un modelo *baseline* y con el trabajo previo de Ham2Pose; y se presentan las conclusiones del capítulo.

6.1. Modelo Mistral-7B-Instruct-v0.2

Mistral AI⁷ es una *startup* francesa fundada en el año 2023 por ex miembros de empresas tecnológicas, especializada en inteligencia artificial.

En septiembre de 2023, Mistral lanzó el modelo conocido como *Mistral*

⁷<https://mistral.ai/>

7B (Jiang et al. 2023) con aproximadamente 7,3 billones (mil millones) de parámetros. Este desarrollo representó un intento por alcanzar un equilibrio entre eficiencia y rendimiento, logrando superar en diferentes *benchmarks* a los mejores modelos de 13 billones de parámetros, como el modelo Llama 2 13B (Touvron, Martin et al. 2023).

Mistral 7B incorporó técnicas avanzadas de atención como *sliding window attention* (Beltagy et al. 2020) y *grouped-query attention* (Ainslie et al. 2023) que permiten una inferencia más rápida y un manejo eficiente de secuencias largas. Junto con *Mistral 7B*, se presentó también *Mistral-7B-Instruct-v0.1* una versión del modelo original ajustada a tareas de instrucción, entrenada con *datasets* de instrucción públicos disponibles en la plataforma Hugging Face⁸.

Unos meses después, se implementa *Mistral-7B-v0.2*, una versión mejorada de *Mistral 7B* que incorpora varias modificaciones, como la eliminación del mecanismo *sliding window attention* y el incremento de la capacidad de la ventana de contexto de 8k a 32k tokens. *Mistral-7B-Instruct-v0.2*, el modelo utilizado en este proyecto, es una versión *Mistral-7B-v0.2* ajustada para tareas basadas en instrucciones.

6.1.1. Formato de las instrucciones

Para realizar inferencias o generar el conjunto de datos para ajustar el modelo *Mistral-7B-Instruct-v0.2*, se deben seguir las siguientes especificaciones:

- Las instrucciones deben estar delimitadas por los *tokens* [INST] y [/INST] para indicar el comienzo y el fin de cada instrucción, respectivamente.
- La primera instrucción debe comenzar con el identificador de inicio de oración <s>, mientras que las siguientes instrucciones no deben incluirlo.
- La generación de la respuesta del asistente debe terminar con el identificador de fin de oración </s>.

6.2. Instrucciones definidas

Dado un LLM F , una instrucción I , la respuesta del modelo es

$$\hat{P} = \mathcal{F}(I),$$

⁸<https://huggingface.co/>

siendo

$$\hat{P} = \{\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n\}$$

una secuencia de poses humanas. Cada $\hat{p}_i$ es un código del *codebook* de la red VQVAE, el cual se decodifica mediante el *decoder* para obtener la pose correspondiente.

La plantilla utilizada para generar el *dataset* de entrenamiento del LLM, respetando el formato de *Mistral-7B-Instruct-v0.2*, es la siguiente:

Instrucción (*I*)

```
<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>
```

{HamNoSys description} es la descripción de una secuencia de símbolos HamNoSys utilizando alguna de las modalidades descritas en el Capítulo 5.

{Sequence of human poses} es la secuencia obtenida luego de codificar una secuencia de poses humanas utilizando el *encoder* de una red VQ-VAE.

6.3. Dataset

Previo a la generación de las instrucciones se dividieron los 5749 ejemplos del *dataset* original de Ham2Pose (Shalev-Arkushin et al. 2023) en particiones de entrenamiento (80 %), validación (10 %) y testeo (10 %). La distribución por lengua dentro cada partición mantiene la distribución del conjunto total (ver Tabla 6.1).

Tabla 6.1: Composición por lengua para cada partición del *dataset* utilizado para ajustar el LLM.

	composición
pjm	44,7 %
dgs	32,2 %
gsl	15,5 %
lsf	6,6 %

6.4. Entrenamiento

Se utilizó **Mistral-7B-Instruct-v0.2** como modelo de lenguaje base, el cual fue ajustado mediante la técnica LoRA.

Los siguientes hiperparámetros fueron utilizados para todos los entrenamientos ejecutados:

- Número de pasos: 22000 (almacenando *checkpoints* cada 2000 pasos).
- Tamaño de *batch*: 4.
- Optimizador: `paged Adamw 32bit`.
- *Learning rate*: 0.0002.
- *Weight decay*: 0.001.
- *Alpha* de LoRA: 16.
- *Dropout* de LoRA: 0.1.

Con el fin de optimizar aún más el uso de la memoria durante el entrenamiento, se aplicó la técnica LoRA en conjunto con cuantización empleando la biblioteca *bitsandbytes*⁹. Los parámetros se almacenan con una precisión de 4 bits (mediante la opción `load_in_4bit=True`) en lugar de 32, utilizando la variante NF4 (4-bit Normal Float). Al momento de realizar las operaciones, estos parámetros se convierten temporalmente a 16 bits (`bnb_4bit_compute_dtype=torch.bfloat16`).

6.5. Posprocesamiento de las salidas generadas

Tras analizar las respuestas generadas por los LLMs ajustados, se observó que, en ocasiones, la salida no se ajustaba estrictamente al formato especificado. Por ejemplo, incluía texto adicional antes o después de la lista de códigos. En consecuencia, se implementó un método de posprocesamiento tolerante, capaz de identificar y extraer una secuencia válida de códigos incluso en casos donde el formato no sea perfectamente válido. Este procedimiento consiste en localizar en el texto crudo una secuencia de números enteros que cumpla con los siguientes criterios:

- Los números deben estar separados por comas, pudiendo existir espacios opcionales antes o después de cada coma.

⁹<https://github.com/bitsandbytes-foundation/bitsandbytes>

- No se requiere que la secuencia esté delimitada explícitamente por corchetes.
- Cada número debe corresponder a un código válido del diccionario de la red VQ-VAE.
- La secuencia debe contener al menos 5 códigos.
- La secuencia debe contener al menos dos códigos diferentes.

Si no se encuentra ninguna secuencia que cumpla con estos requisitos, se vuelve a ejecutar la inferencia. Cuando se obtiene una secuencia válida, podría ocurrir que esté truncada debido a una interrupción durante la generación de los dígitos del último código. Esto puede suceder si se alcanza el límite de longitud de salida del modelo de lenguaje. En particular, si el último código de la secuencia parece ser una versión truncada de los dígitos del penúltimo código, se considera una repetición incompleta de este último. En estos casos, para evitar inconsistencias en la secuencia generada, se elimina el último código como medida de precaución.

6.6. Experimentos

6.6.1. Elección de temperatura para inferencia

Al analizar los resultados de las inferencias generadas por los modelos ajustados, se observó que las secuencias presentaban repeticiones continuas de códigos de forma exagerada, lo que genera movimientos relativamente estáticos y menos dinámicos en comparación con las secuencias del *ground truth*.

Esto podría estar relacionado con el proceso de cuantización utilizado en la red VQ-VAE, donde las poses originales son mapeadas a un conjunto discreto de códigos, lo que puede provocar la pérdida de variaciones sutiles entre poses similares, comportamiento que el LLM ajustado podría estar amplificando durante la creación de nuevas secuencias.

Para abordar este problema, en un primer intento se decidió experimentar con diferentes valores de temperatura al realizar inferencias con los LLMs entrenados, bajo la hipótesis de que valores menos próximos a 0 podrían generar secuencias más variadas.

La Figura 5.1 presenta la evolución de la métrica $nDTW-MJE$ a lo largo del proceso de entrenamiento, mostrando el desempeño obtenido al utilizar cuatro

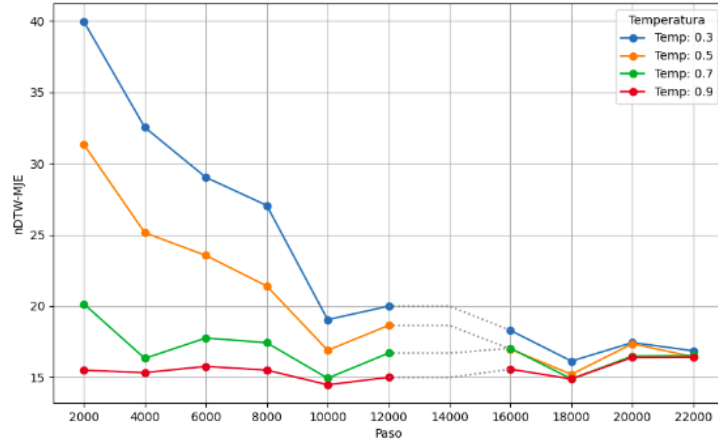


Figura 6.1: Evolución de $nDTW-MJE$ para diferentes valores de temperatura

valores diferentes de temperatura (0.3, 0.5, 0.7 y 0.9) durante la generación de secuencias. En el paso 14000 no fue posible obtener secuencias válidas para ninguno de los valores de temperatura evaluados, por lo que se ha omitido en la representación gráfica.

En las primeras etapas, los valores de temperatura menores (más cercanos a 0) generan secuencias con mayor error. Este resultado concuerda con la hipótesis inicial: cuando la temperatura es muy baja, el modelo es más conservador y tiende a repetir poses generando secuencias menos dinámicas.

A medida que avanza el entrenamiento, todas las configuraciones de temperatura convergen a un nivel de error similar, en el entorno de 15, indicando que, si bien la temperatura influye significativamente en las primeras etapas, su impacto puede reducirse a mayor entrenamiento. A pesar de esta convergencia, los valores reportados para la temperatura 0.9 son siempre los mínimos, independientemente del paso en cuestión.

Se decide utilizar para todos los próximos experimentos un único valor, 0.9, de temperatura al momento de realizar las inferencias.

6.6.2. Reducción de repeticiones consecutivas

Para mitigar el problema de las repeticiones excesivas de códigos en las secuencias generadas, se exploró también el impacto de aplicar distintas técnicas que permitan, en una etapa de preprocesamiento, disminuir la cantidad de repeticiones consecutivas de códigos en el conjunto de datos de entrenamiento. A continuación, se detallan las cuatro técnicas implementadas:

- **Eliminación de repeticiones:** consiste en eliminar todas las repeticiones consecutivas de un mismo código, manteniendo únicamente el primer código de cada secuencia repetida.
- **Reducción logarítmica:** la cantidad de repeticiones consecutivas se ajusta a:

$$\min(1, \log(\#repeticiones)),$$

manteniendo un mínimo de una repetición.

- **Reducción por raíz cuadrada:** la cantidad de repeticiones consecutivas se ajusta a:

$$\min(1, \sqrt{\#repeticiones}),$$

- **Truncamiento de repeticiones:** este método limita el número máximo de repeticiones consecutivas permitidas para cualquier código a un umbral fijo (6 repeticiones).

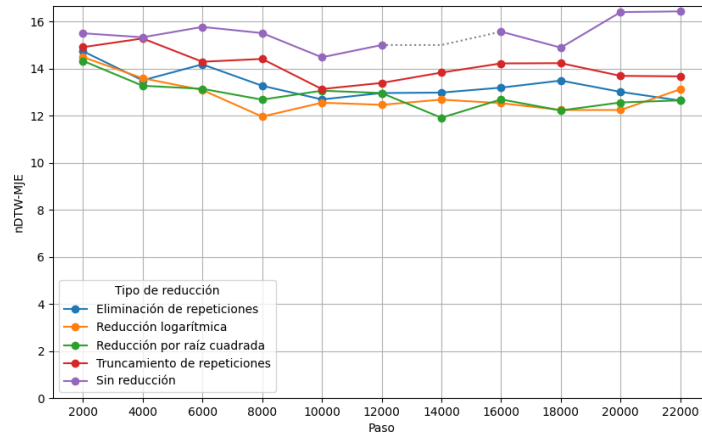


Figura 6.2: Evolución de $nDTW-MJE$ para los diferentes técnicas de reducción de repeticiones

En la Figura 6.2 se presenta la evolución de la métrica $nDTW-MJE$ en función del número de pasos, con temperatura 0.9, para cada técnica empleada. La reducción es aplicada únicamente en el conjunto de entrenamiento por lo que al momento de evaluar se comparara con las secuencias originales del conjunto de validación. Todas las técnicas de reducción de repeticiones demuestran un impacto positivo en el error $nDTW-MJE$. Entre ellas, las reducciones logarítmica y por raíz cuadrada destacan por ofrecer un mejor desempeño. Para

futuros experimentos, se optará por aplicar la reducción por raíz cuadrada, ya que mostró el valor mínimo de $nDTW-MJE$ (11.91) en el paso 14000.

6.6.3. Modalidad de descripción

La gráfica de la figura 6.3 muestra la evolución de la métrica $nDTW-MJE$ en función de los pasos, con temperatura 0.9 y reducción de repeticiones por raíz cuadrada, para las distintas modalidades de descripción de la secuencia HamNoSys definidas en el Capítulo 5.

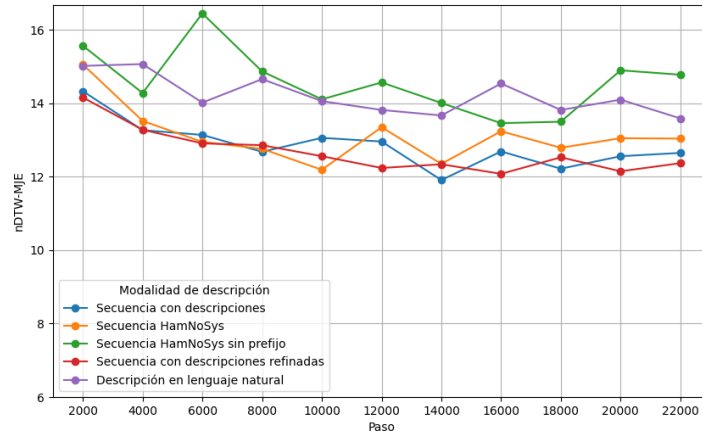


Figura 6.3: Evolución de $nDTW-MJE$ para las diferentes modalidades de descripción

Las dos modalidades basadas en descripciones en lenguaje natural de cada símbolo HamNoSys (*secuencia con descripciones* y *secuencia con descripciones refinadas*) alcanzan los valores más bajos del error $nDTW-MJE$. La *secuencia con descripciones refinadas* muestran un desempeño más estable a lo largo de los pasos, mientras que la *secuencia con descripciones* logra el valor mínimo de $nDTW-MJE$. En contraposición, las descripciones en lenguaje natural generadas a partir de un LLM, producen los peores resultados. Se observa un mejor desempeño para la modalidad *secuencias HamNoSys* en comparación con modalidad *secuencia HamNoSys sin prefijo*, lo que sugiere que el LLM podría poseer un cierto “conocimiento” previo acerca de la notación HamNoSys, en base a los datos utilizados para su pre-entrenamiento.

Se observa que agregar información en lenguaje natural al momento de generar descripciones de HamNoSys resulta beneficioso, siempre y cuando los

símbolos se reemplacen por sus explicaciones predefinidas en lenguaje natural y no se genere la descripción completa de la secuencia en dicho lenguaje. Se concluye que es necesario que el LLM cuente con una noción clara del orden y del contexto de los símbolos, de modo que pueda identificar patrones y captar las sutilezas propias de la notación.

Como se indicó en la Sección 5.4.5, es posible mejorar las descripciones en lenguaje natural utilizando *prompts* con explicaciones más precisas y contextualizadas de los símbolos. Estas descripciones de mayor calidad podrían ayudar a optimizar el entrenamiento del LLM.

6.6.4. Rango de LoRA

Se llevaron a cabo experimentos empleando diferentes valores del parámetro r (32, 64 y 128) de LoRA, con temperatura 0.9, reducción de repeticiones por raíz cuadrada y secuencias con descripciones.

El parámetro r corresponde al rango de la descomposición de LoRA. Un r más pequeño reduce el número de parámetros entrenables, lo que puede llevar a un entrenamiento más rápido y menos uso de memoria, pero también puede limitar la capacidad del modelo para aprender patrones complejos.

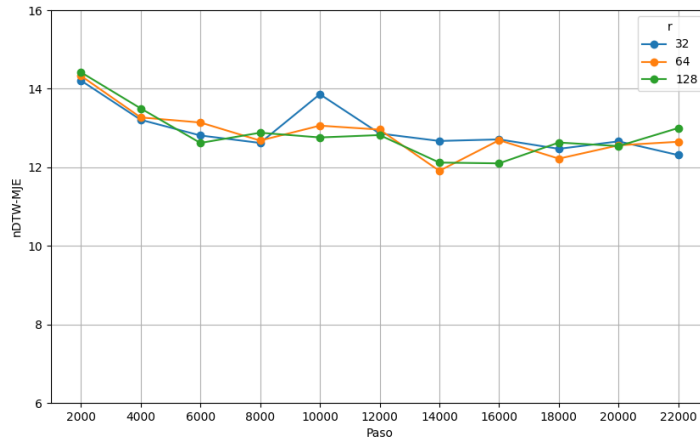


Figura 6.4: Evolución de $nDTW-MJE$ para los diferentes valores del rango de LoRA.

Como se observa en la Figura 6.4, no se observan diferencias considerables entre los distintos valores utilizados, en particular para los valores 64 y 128. Se opta por utilizar de ahora en más el valor 64 porque es valor para el que

se reporta el valor mínimo de $nDTW-MJE$ (11,91), y a su vez porque implica trabajar con una menor cantidad de parámetros.

6.6.5. Tamaño del *codebook*

El último experimento realizado tuvo como objetivo evaluar el impacto del tamaño del *codebook* de la red VQ-VAE utilizada para codificar y decodificar las poses de lengua de señas. Para ello, se utilizó una red VQ-VAE de 1024 códigos, y se la comparó con la red de 512 códigos utilizada en los experimentos anteriores. Para estas pruebas se utilizó el *dataset* sin reducir el número de repeticiones consecutivas.

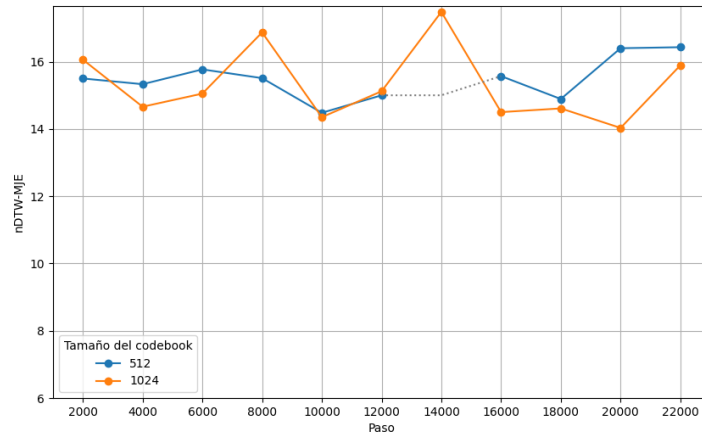


Figura 6.5: Evolución de $nDTW-MJE$ para diferentes tamaños del *codebook* de la red VQVAE.

En la Figura 6.5 se presentan los resultados obtenidos, los cuales no muestran diferencias significativas entre ambos tamaños de *codebook*, teniendo en cuenta los mayores costos de entrenamiento e inferencia que implican un incremento en la cantidad de códigos. Sin embargo, se destaca que el valor mínimo alcanzado de $nDTW-MJE$ con 1024 códigos fue de 14.03, inferior al valor óptimo alcanzado con 512 códigos, que fue de 14.48.

A pesar de esta mejora, la ganancia lograda con el incremento en el tamaño del *codebook* es menor en comparación con la obtenida mediante la reducción de repeticiones en el conjunto de datos.

Estos resultados sugieren que una prueba futura podría consistir en combinar este incremento en el número de códigos con las técnicas de reducción

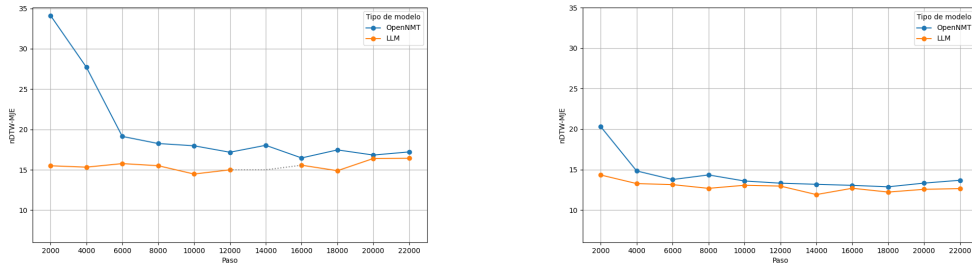
de repeticiones.

6.7. Comparación con *baseline*

OpenNMT (Klein et al. 2017) una plataforma de código abierto diseñado para la traducción automática basada en redes neuronales. En este trabajo se utiliza *OpenNMT-py*¹⁰, la implementación de OpenNMT basada en PyTorch (Paszke et al. 2019), para llevar a cabo la traducción de secuencias HamNoSys a secuencias de códigos del diccionario de la red VQ-VAE.

En particular, se utiliza como *baseline* el modelo predeterminado de OpenNMT-py, que cuenta con una arquitectura de redes neuronales recurrentes. Tanto el *encoder* como el *decoder* están implementados con un red LSTM (Hochreiter y Schmidhuber, 1997) de 2 capas con 500 unidades ocultas.

Se ejecutó un algoritmo de tokenización simple mediante el cual cada elemento del vocabulario de entrada representa un símbolo HamNoSys, y cada elemento del vocabulario de salida representa un código del diccionario de la red VQ-VAE.



(a) Sin reducción de repeticiones

(b) Reducción por raíz cuadrada

Figura 6.6: Evolución de $nDTW-MJE$: LLM vs OpenNMT

En la Figura 6.6 se presenta una comparativa del desempeño entre OpenNMT y el LLM (con la configuración óptima) en base al conjunto de validación, utilizando tanto el *dataset* original como el *dataset* con reducción de repeticiones por raíz cuadrada. Los resultados muestran un rendimiento superior por parte del LLM en ambos casos. Sin embargo, el rendimiento de OpenNMT no es significativamente inferior; de hecho, supera a otras configuraciones de LLM evaluadas.

¹⁰<https://github.com/OpenNMT/OpenNMT-py>

Conjunto	Modelo	nDTW-MJE ↓	Predicción			Ground truth		
			Rank 1 ↑	Rank 5 ↑	Rank 10 ↑	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑
Validación	OpenNMT	12,87	0.10	0.38	0.58	0.24	0.56	0.73
	LLM	11,91	0.10	0.40	0.61	0.26	0.57	0.75
Testeo	OpenNMT	12.77	0.09	0.39	0.58	0.26	0.56	0.72
	LLM	12.38	0.09	0.41	0.60	0.25	0.58	0.74

Tabla 6.2: Métricas *nDTW-MJE* y *Distance Ranks* para los conjuntos de validación y testeo.

En la Tabla 6.2 se presentan las métricas *nDTW-MJE* y los valores de *Distance Ranks* correspondientes a los *checkpoints* de LLM y de OpenNMT (para las particiones de testeo como validación) en los que se obtuvieron los valores mínimos de *nDTW-MJE*.

Los resultados indican que tanto para LLM como para OpenNMT, el desempeño observado entre las particiones de validación y testeo es consistente. Al comparar ambos modelos, se observa en términos generales un mejor desempeño de LLM, con la excepción de la métrica *rank 1* (tomando como referencia el *ground truth*) donde OpenNMT supera a LLM por un punto porcentual.

Asimismo, debido al carácter no determinístico de las salidas del LLM, pueden producirse ligeras variaciones entre diferentes ejecuciones de inferencia, y en consecuencia se puede concluir que ambos tipos de modelos presentan desempeños bastante comparables.

Es interesante notar que la arquitectura LSTM de OpenNMT, a pesar de su simplicidad en comparación con la basada en Transformers del LLM, logra obtener resultados muy competitivos en este problema. Esto sugiere que podrían explorarse más experimentos con esta técnica en el futuro. Una posible razón de su buen desempeño es el tamaño relativamente reducido del conjunto de datos, lo que facilita el entrenamiento de un modelo *encoder-decoder* basado en LSTM en comparación con un enfoque basado en Transformers, que suele requerir grandes volúmenes de datos para entrenarse de manera efectiva. El Transformer empleado en este caso está preentrenado, lo que puede otorgar ventajas en términos representación y generalización. En definitiva, ambos enfoques presentan fortalezas y debilidades.

6.8. Comparación con Ham2Pose

En esta sección se busca hacer una comparación entre los resultados obtenidos en el presente trabajo y los reportados por Ham2Pose en Shalev-Arkushin

et al. 2023. Es importante señalar que el *dataset* proporcionado por Ham2Pose no incluía una división explícita en particiones. Aunque el código publicado por los autores incluye un *script* para realizar el particionamiento, este proceso se lleva a cabo de manera aleatoria, lo que imposibilita replicar exactamente los mismos subconjuntos de datos. Tal como se describió en este capítulo, tanto para el entrenamiento del modelo basado en LLM como el modelo de OpenNMT, se generaron tres particiones (entrenamiento, validación y testeo), manteniendo la misma proporción por lengua de señas que en el conjunto total.

Modelo	Predicción			Ground truth		
	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑
Ham2Pose	0.08	0.20	0.35	0.21	0.44	0.56
OpenNMT	0.09	0.39	0.58	0.26	0.56	0.72
LLM	0.09	0.41	0.60	0.25	0.58	0.74

Tabla 6.3: Comparación de *Distance Rnks* entre Ham2Pose, OpenNMT y LLM (considerando que los conjuntos de datos no son exactamente comparables).

Al comparar la métrica *Distance Rnks* entre los modelos entrenados en el contexto de este proyecto y el modelo de Ham2Pose (ver Tabla 6.3), se observa que tanto el LLM como OpenNMT exhiben un rendimiento superior para todos los *ranks*, ya sea tomando como referencia las predicciones o el *ground truth*. Las diferencias son especialmente significativas para *rank 5* y *rank 10*, mientras que para *rank 1* resultan mucho más comparables.

En la Figura 6.4 se presenta un análisis detallado del rendimiento por lengua de señas, comparando en particular el LLM con Ham2Pose. Se observa que el LLM obtiene mejores resultados para PJM y DGS, las dos lenguas mayoritarias del *dataset*. En contraste, Ham2Pose muestra un mejor desempeño en la lengua minoritaria, LSF. Para GSL, los resultados no indican un claro ganador.

6.9. Análisis cualitativo

A continuación se presenta el análisis de algunos ejemplos de resultados con el objetivo de mostrar características generales de las salidas generadas por el sistema, así como algunos comportamientos y errores comunes que se observan. En primer lugar, el análisis cualitativo de los ejemplos examinados, en base a los bloques y sub-bloques de la estructura HamNoSys, permite identificar

Ref	Lengua	Rank 1	Rank 5	Rank 10
Pred	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	0/0	0.1/ 0.18	0.25/ 0.38
	LSF	0.22 /0.08	0.6 /0.18	0.8 /0.21
GT	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	0.8 /0.76
	LSF	0.7 /0.13	0.95 /0.47	0.97 /0.63

Tabla 6.4: *Distance Ranks* por lengua de señas (Ham2Pose / LLM).

qué características de la poses se pueden producir más fielmente, y cuáles se obtienen con menor precisión:

1. La representación del ***tipo de seña*** mostró el desempeño más destacado, diferenciando adecuadamente entre señas de una o dos manos y los tipos de simetría entre ambas.
2. El ***movimiento de los brazos*** también exhibió una buena precisión, logrando identificar los tipos de trayectorias y las repeticiones.
3. Las configuraciones de la mano presentan los mayores desafíos, siendo el sub-bloque correspondiente a la ***localización de la mano***, el de mayor precisión.
4. La ***forma y orientación de las manos*** arrojaron rendimientos heterogéneos, con algunos casos interpretados de mejor manera que otros.
5. La ***inclinación de los dedos*** es sin duda el sub-bloque de mayor imprecisión y que requiere más revisión para su mejora.

Estos hallazgos son coherentes con la complejidad del problema, ya que el número de combinaciones posibles en las configuraciones de las manos es significativamente mayor que en los otros sub-bloques.

En la Figura 6.7 se muestra una seña donde se logró representar de forma bastante precisa. En cambio, en la Figura 6.8 se muestra una seña con los movimientos de los brazos correctos pero las configuraciones de las manos incorrectas. En la Figura 6.9 se ve un ejemplo donde hay error de localización de la mano.

Al analizar en detalle errores en las configuraciones de las manos, se observa que parte del error ya se arrastra en la etapa de codificación de las poses a través de la red VQ-VAE, donde se detectan imprecisiones al reconstruir los

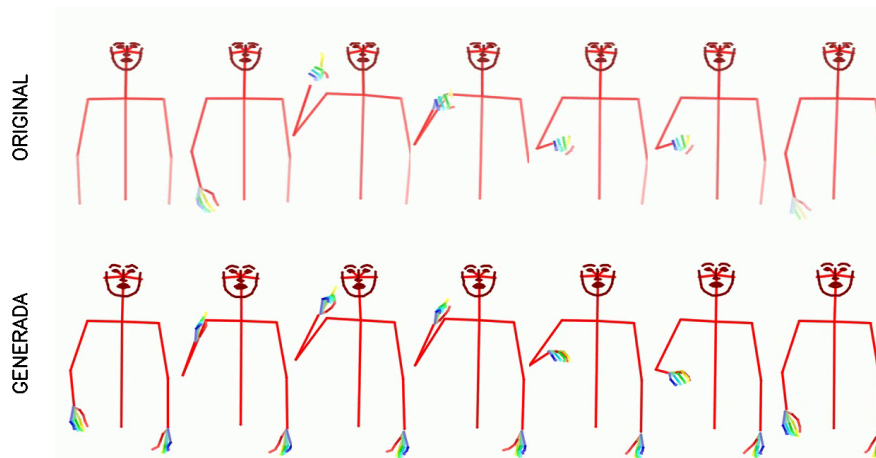


Figura 6.7: Ejemplo de una secuencia de poses generada de forma precisa.

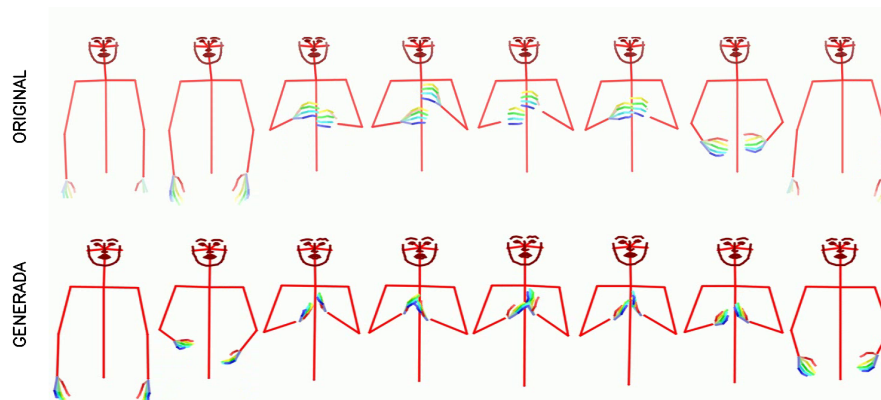


Figura 6.8: Ejemplo de una secuencia de poses con el movimiento de brazos correcto pero con errores en las configuraciones de las manos.

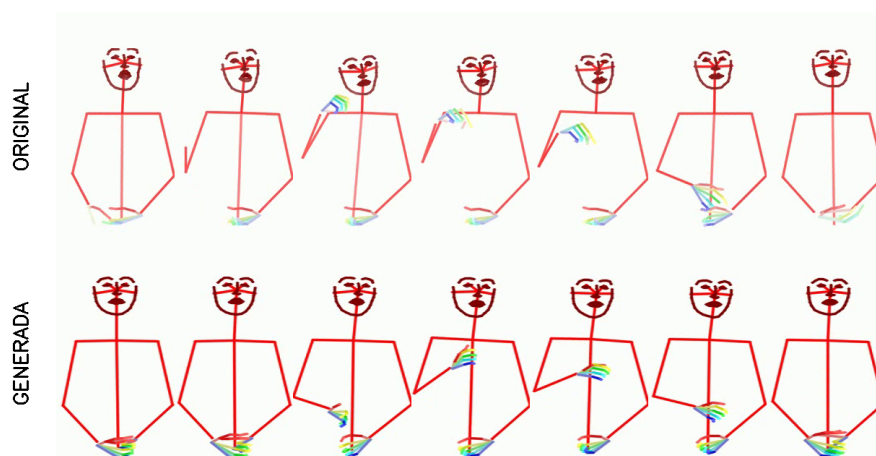


Figura 6.9: Ejemplo de una secuencia de poses con la localización de la mano imprecisa. La mano derecha se lleva al pecho en lugar del hombro derecho.

movimientos y formas de las manos, y en particular la inclinación de los dedos. Son mejor representadas aquellas poses en las cuales hay formas de manos relativamente homogéneas con escasa separación e inclinación de los dedos.

En la Figura 6.10 se visualiza un caso donde se arrastra el error en la etapa de discretización, en cambio en la Figura 6.11 no hay error de discretización sino exclusivamente al momento de la generación por parte del LLM.

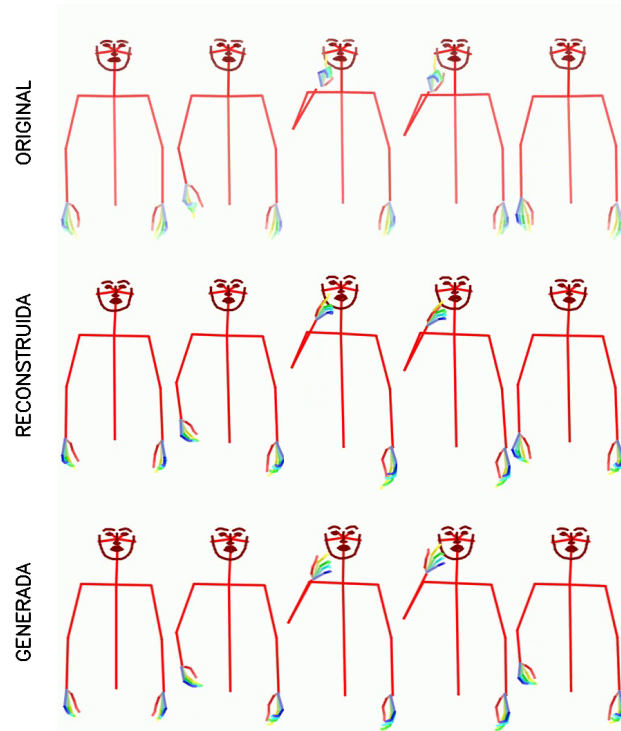


Figura 6.10: Ejemplo donde las imprecisiones en la configuración de las manos de la secuencia generada, ya están presentes en la codificación de las poses

En algunas secuencias se observan errores al representar la mano izquierda (ver Figura 6.12), usualmente en casos de poses de una sola mano donde la misma se encuentra en estado de reposo. Esto probablemente se debe al hecho que en el *dataset* hay muchos *keypoints* faltantes en las manos en reposo, y nuestra implementación termina mapeando esos *keypoints* a (0,0). En futuros trabajos se podría tomar la idea utilizada en Ham2Pose de modificar las funciones de loss utilizando determinadas ponderaciones de forma que el modelo aprenda a rellenar esos *keypoints* faltantes y genere poses completas.

El LLM entrenado también tiende a generar secuencias de menor duración que las originales, probablemente debido al efecto de eliminar repeticiones.



Figura 6.11: Ejemplo donde la secuencia generada muestra errores en la configuración de la mano, a pesar de que las poses son codificadas correctamente

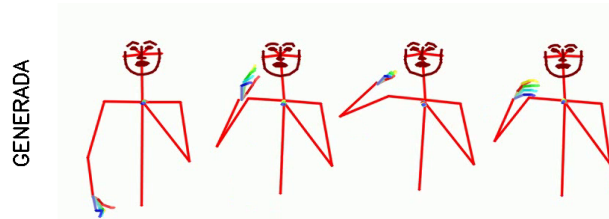


Figura 6.12: Ejemplo donde la secuencia generada muestra errores al representar la mano izquierda en reposo.

6.10. Conclusiones

En este capítulo se presentaron los resultados de los experimentos de ajuste fino de un modelo de lenguaje para la predicción de poses de lengua de señas. A continuación, se discuten las principales observaciones y recomendaciones derivadas del estudio.

Uno de los principales hallazgos es que la estrategia de cuantización de la red VQ-VAE, utilizada para codificar las poses de lengua de señas, puede generar secuencias con repeticiones consecutivas amplificadas después de entrenar el LLM. Para mitigar este problema, se recomienda:

- Utilizar valores altos de temperatura.
- Entrenar el modelo durante un número significativo de pasos.
- Aplicar técnicas de preprocesamiento en el conjunto de datos para reducir las repeticiones consecutivas.

- Incrementar el tamaño del diccionario de códigos en la red VQ-VAE.

Se observaron diferencias relacionadas con la forma de representar las secuencias de símbolos HamNoSys, concluyéndose que la inclusión de representaciones semánticas de los símbolos y de información sobre su orden resulta beneficiosa para el entrenamiento del LLM. Asimismo, se determinó que un valor de rango 64 en LoRA constituye un buen balance entre eficiencia y rendimiento.

Por otra parte, aunque el modelo LLM mostró resultados superiores en comparación con una red LSTM implementada mediante OpenNMT, las diferencias no resultaron significativamente notables. En términos de resultados cuantitativos, se obtienen métricas competitivas en comparación con el proyecto Ham2Pose (con la consideración que los conjuntos de datos no son idénticos).

Asimismo, a pesar de que HamNoSys es un sistema de notación de carácter universal, los resultados de este estudio reportan un mejor desempeño en ejemplos correspondientes a las lenguas predominantes dentro del conjunto de datos utilizado.

El análisis cualitativo de los resultados permite identificar que algunos bloques de la estructura HamNoSys son representados con mayor precisión que otros, siendo la configuración de la mano la más compleja de representar.

Para mejorar estos aspectos en el futuro, podría ser beneficioso aumentar el tamaño del *dataset*, así como incrementar el número de códigos del diccionario, y en particular explorar el uso de *codebooks* diferenciados: uno para cada mano y otro para el resto del cuerpo. Esto último permitiría aumentar la cantidad de poses disponibles para representar con mayor precisión las configuraciones específicas de la mano, lo que también podría mitigar el problema de las repeticiones consecutivas.

Además, se podría ajustar el posprocesamiento utilizado para extraer la secuencia de códigos de poses a partir de la inferencia del LLM, con el objetivo de hacerlo menos permisivo, lo que podría mejorar la calidad de las salidas generadas. También sería útil experimentar con la definición de *prompts* más restrictivas o con el uso de modelos optimizados para generar salidas estructuradas, por ejemplo, en formato JSON¹¹.

En este trabajo se optó por realizar todos los experimentos con una una

¹¹<https://www.json.org/>

temperatura de 0.9 por su efecto positivo en la reducción de repeticiones consecutivas. Sin embargo, para investigaciones futuras, se sugiere explorar valores de temperatura más bajos. Dado que ya se están implementando métodos adicionales para mitigar las repeticiones, reducir la temperatura podría contribuir a generar secuencias más consistentes y con menor aleatoriedad entre diferentes inferencias.

En general, los resultados presentados en este capítulo muestran un desempeño competitivo en la generación de secuencias de poses de lengua de señas. Se evidencia el potencial de la combinación de técnicas de generación con discretización aplicada en este trabajo, aunque también se identifican limitaciones que pueden abordarse en el futuro.

Capítulo 7

Conclusión

La presente tesis abordó el desafío de la producción de poses de lengua de señas a partir de representaciones en lenguaje natural de símbolos HamNoSys, con un enfoque innovador basado en modelos generativos.

7.1. Estado del arte y metodología

Desde una perspectiva teórica, se realizó un análisis exhaustivo del estado del arte en producción de lengua de señas. Este análisis evidenció que la generación de avatares de calidad condicionados a poses todavía no alcanza un nivel de madurez que permita obtener resultados robustos. Por este motivo, la investigación se concentró en la generación de poses de lengua de señas a partir de una notación intermedia, etapa en la que sí se han logrado avances medibles y comparables.

Se revisaron las principales notaciones utilizadas en su representación, identificando que HamNoSys es una opción óptima para este proyecto debido a su estructura secuencial y universalidad. Se evaluaron métodos previos, como Ham2Pose, identificando sus limitaciones y oportunidades de mejora mediante el uso de modelos generativos recientes.

7.2. Método propuesto

Se propuso un método novedoso para la transformación de secuencias HamNoSys en secuencias de poses, tomando inspiración del proyecto MotionGPT, basado en la combinación de:

- Un generador de descripciones HamNoSys, capaz de convertir los símbolos en explicaciones detalladas en lenguaje natural.
- Un LLM ajustado mediante *Instruction Tuning*, que recibe las descripciones anteriores y predice *tokens* discretos de pose.
- Una red VQ-VAE, encargada de codificar y decodificar las secuencias de poses hacia y desde esos *tokens* discretos.

Al “traducir” primero los símbolos HamNoSys a oraciones en lenguaje natural y luego emplear un LLM, se aprovecha el conocimiento encapsulado en LLMs preentrenados, transformando el problema de generación de poses a partir de HamNoSys en una tarea de generación condicionada por texto.

7.3. Resultados obtenidos

La implementación de VQ-VAE demostró ser efectiva para la representación discreta de poses, aunque se identificaron desafíos asociados a la cuantización, como la amplificación de repeticiones consecutivas. Para mitigar este efecto, se propusieron estrategias como el ajuste de la temperatura en la inferencia del LLM y el preprocesamiento del conjunto de datos para reducir el número de repeticiones consecutivas.

El ajuste del LLM evidenció que la inclusión de información semántica, pero manteniendo el orden estructural de los símbolos HamNoSys, mejora el rendimiento en la generación de poses.

Los resultados experimentales indican que el enfoque propuesto logra un rendimiento comparable al proyecto Ham2Pose (con la consideración de que los conjuntos de datos utilizados no son idénticos), demostrando el potencial de la combinación de técnicas de generación con discretización aplicada en este trabajo.

El desarrollo de diccionarios con explicaciones en lenguaje natural de los símbolos HamNoSys no solo ha permitido el entrenamiento efectivo del LLM, sino que también podría contribuir a la accesibilidad y difusión de la lengua de señas.

7.4. Limitaciones identificadas

A pesar de los resultados obtenidos, el trabajo presentado también pone en evidencia algunas limitaciones:

1. La necesidad de un *corpus* más amplio:

- Un conjunto de datos más extenso permitiría aprovechar mejor las capacidades de generalización y el conocimiento previo del LLM, facilitando la generación de secuencias de poses de mayor calidad.
- Con respecto al entrenamiento de la VQ-VAE, en este caso se optó por extraer las poses correspondientes al *dataset* con descripciones HamNoSys. Podría extenderse utilizando un *corpus* más amplio de videos de lengua de señas, independientemente de la notación intermedia, ya que solo es necesario extraer las poses para capturar el espacio de movimientos.

2. Calidad del *dataset*:

- Se podría realizar un preprocesamiento más exhaustivo del *dataset* para detectar inconsistencias o errores en el uso de los símbolos HamNoSys entre los diferentes conjuntos de datos utilizados para generar el *corpus*.
- La conversión de representaciones 2D de OpenPose a 3D podría permitir una representación más precisa de las poses.
- Se podrían explorar alternativas para la detección de poses, como MediaPipe¹², que podría reducir la cantidad de *keypoints* faltantes o incorrectamente detectados.

3. Problema de repeticiones consecutivas:

Se ha detectado que la estrategia de cuantización puede generar secuencias con repeticiones consecutivas amplificadas después del entrenamiento del LLM, lo que plantea la necesidad de pruebas adicionales en futuros trabajos:

- La exploración de múltiples *codebooks*, diferenciando entre la mano derecha, la mano izquierda y otros *keypoints*, podría mejorar la precisión de la reconstrucción y reducir las inconsistencias en la generación de poses, especialmente en la representación de las configuraciones de la mano.

¹²<https://github.com/google-ai-edge/mediapipe>

- Combinar las técnicas de disminución de repeticiones con *codebooks* más grandes.
4. **Exploración limitada de hiperparámetros:** Aunque se han realizado experimentos con diferentes configuraciones, una exploración más exhaustiva de todos los posibles hiperparámetros tanto de la VQ-VAE como del proceso de *fine tuning* del LLM queda para futuros trabajos.
 5. **Complejidades de la notación HamNoSys:** Se podrían implementar mejoras en las distintas modalidades de generación de las descripciones HamNoSys, en base a los desafíos identificados, de forma de contar con instrucciones más precisas al momento de entrenar el LLM.
 6. **Limitaciones de las métricas utilizadas:** Al realizar el análisis cualitativos, se puede observar inconsistencias en ciertos resultados que son cuantitativamente similares en base a la distancia $nDTW-MJE$, pero visualmente diferentes en cuanto a las configuraciones de las manos. Quizá la implementación de esta distancia, así como de las funciones de loss utilizadas, podrían modificarse para dar más peso a los *keypoints* de la mano de forma de forzar la optimización de representación de esa parte del cuerpo en particular.

7.5. Agenda para investigaciones futuras

Se propone la siguiente agenda para futuras investigaciones:

1. **Exploración de arquitecturas alternativas:** Como el método propuesto incluye diferentes componentes, se pueden probar alternativas a los mismos.
 - Se puede investigar el potencial de modelos de difusión en sustitución del LLM. En particular, se podría aprovechar el modelo CLIP utilizado en Stable Diffusion, ya que su naturaleza bimodal permitiría establecer correlaciones entre descripciones textuales de movimientos de lengua de señas y representaciones visuales de poses.
 - Se podría reemplazar el LLM utilizado, ya sea por otro modelo de Mistral con más parámetros, o por modelos más recientes como LLaMA 3 (Dubey et al. 2024).
 - En el proceso de *fine tuning* del LLM se utilizó el *tokenizer* original del modelo provisto por Mistral. Se podría probar extender el vo-

cabulario del LLM agregando *tokens* específicos para la poses (en base a los códigos del diccionario de la red VQ-VAE).

2. **Potencial para aplicaciones multimodales:** Las técnicas desarrolladas podrían extenderse a otras aplicaciones que requieran la representación de movimientos detallados de las manos, más allá del ámbito específico de la lengua de señas.
3. **Integración con la generación de avatares:** La integración con futuras soluciones para la generación de avatares representaría el siguiente paso lógico hacia un sistema completo de producción de lengua de señas.
4. **Evaluación cualitativa con usuarios reales:** La validación mediante pruebas con personas sordas e intérpretes de lengua de señas permitiría obtener retroalimentación crítica sobre la naturalidad, precisión y comprensibilidad de las poses generadas

7.6. Conclusión final

En resumen, esta tesis evidencia el potencial de los modelos generativos, en particular los LLMs ajustados en combinación con técnicas de cuantización vectorial, para la generación de lengua de señas a partir de la notación HamNoSys. Aunque se han identificado algunas limitaciones, los resultados obtenidos son prometedores y abren múltiples vías para futuras investigaciones. El desarrollo de estas tecnologías no solo impulsa el avance del estado del arte en inteligencia artificial, sino que también tiene el potencial de mejorar significativamente la accesibilidad y la inclusión de millones de personas sordas en todo el mundo.

Referencias bibliográficas

- Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebr'on, F., y Sanghai, S. K. (2023). GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *ArXiv*, *abs/2305.13245*. <https://api.semanticscholar.org/CorpusID:258833177>
- Bangham, J., Cox, S., Elliott, R., Glauert, J., Marshall, I., Rankov, S., y Wells, M. (2000). Virtual signing: capture, animation, storage and transmission-an overview of the ViSiCAST project. *IEE Seminar on Speech and Language Processing for Disabled and Elderly People (Ref. No. 2000/025)*, 6/1-6/7. <https://doi.org/10.1049/ic:20000136>
- Beltagy, I., Peters, M. E., y Cohan, A. (2020). Longformer: The Long-Document Transformer. *ArXiv*, *abs/2004.05150*. <https://api.semanticscholar.org/CorpusID:215737171>
- Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., y Kreis, K. (2023). Align Your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 22563-22575. <https://api.semanticscholar.org/CorpusID:258187553>
- Cabeza, C., y García-Miguel, J. M. (2019). iSignos: Interfaz de datos de Lengua de Signos Española (versión 1.0) [[Consulted: March 18, 2025]]. <http://isignos.uvigo.es/>
- Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., y Sheikh, Y. (2018). OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43, 172-186. <https://api.semanticscholar.org/CorpusID:198169848>
- Cripps, J. H. (2008). A case study on reading processes of signing deaf children. <https://api.semanticscholar.org/CorpusID:60032508>
- Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A. S.,

- Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., ... Zhao, Z. (2024). The Llama 3 Herd of Models. *ArXiv, abs/2407.21783*. <https://api.semanticscholar.org/CorpusID:271571434>
- Elliott, R., Bueno, J., Kennaway, R., y Glauert, J. R. W. (2010). Towards the Integration of Synthetic SL Animation with Avatars into Corpus Annotation Tools. <https://api.semanticscholar.org/CorpusID:59694093>
- Gill, B. (2005). Glossing and Reading Comprehension.
- Goyal, L. (2015). Review and Comparison of Writing Notations of Sign Language. *Research Cell : An International Journal of Engineering*. <https://arxiv.org/abs/2404.11499>
- Guo, C., Zou, S., Zuo, X., Wang, S., Ji, W., Li, X., y Cheng, L. (2022). Generating Diverse and Natural 3D Human Motions from Text. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5142-5151. <https://api.semanticscholar.org/CorpusID:250602257>
- Hanke, T. (2004). HamNoSys – Representing Sign Language Data in Language Resources and Language Processing Contexts. <https://api.semanticscholar.org/CorpusID:15434469>
- Hanke, T. (2009). DGS corpus project - Development of a corpus based electronic dictionary German Sign Language / German. <https://api.semanticscholar.org/CorpusID:268107827>
- He, K., Zhang, X., Ren, S., y Sun, J. (2015). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. <https://api.semanticscholar.org/CorpusID:206594692>
- Ho, J., Jain, A., y Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *ArXiv, abs/2006.11239*. <https://api.semanticscholar.org/CorpusID:219955663>
- Hochreiter, S., y Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9, 1735-1780. <https://api.semanticscholar.org/CorpusID:1915014>
- Hu, J. E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., y Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv, abs/2106.09685*. <https://api.semanticscholar.org/CorpusID:235458009>

- Huang, W., Pan, W., Zhao, Z., y Tian, Q. (2021). Towards Fast and High-Quality Sign Language Production. *Proceedings of the 29th ACM International Conference on Multimedia*. <https://api.semanticscholar.org/CorpusID:239011890>
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., y Sayed, W. E. (2023). Mistral 7B. *ArXiv, abs/2310.06825*. <https://api.semanticscholar.org/CorpusID:263830494>
- Khachatryan, L., Movsisyan, A., Tadevosyan, V., Henschel, R., Wang, Z., Navasardyan, S., y Shi, H. (2023). Text2Video-Zero: Text-to-Image Diffusion Models are Zero-Shot Video Generators. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 15908-15918. <https://api.semanticscholar.org/CorpusID:257687280>
- Kingma, D. P., y Welling, M. (2013). Auto-Encoding Variational Bayes. *CoRR, abs/1312.6114*. <https://api.semanticscholar.org/CorpusID:216078090>
- Klein, G., Kim, Y., Deng, Y., Senellart, J., y Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *Annual Meeting of the Association for Computational Linguistics*. <https://api.semanticscholar.org/CorpusID:16538528>
- Kruskal, J. B. (1983). An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules. *Siam Review*, 25, 201-237. <https://api.semanticscholar.org/CorpusID:121839291>
- Li, K., He, Y., Wang, Y., Li, Y., Wang, W., Luo, P., Wang, Y., Wang, L., y Qiao, Y. (2024). VideoChat: Chat-Centric Video Understanding. <https://arxiv.org/abs/2305.06355>
- Ma, Y., He, Y., Cun, X., Wang, X., Chen, S., Shan, Y., Li, X., y Chen, Q. (2024). Follow Your Pose: Pose-Guided Text-to-Video Generation using Pose-Free Videos. <https://arxiv.org/abs/2304.01186>
- Matthes, S., Hanke, T., Regen, A., Storz, J., Worsack, S., Efthimiou, E., Dimou, A.-L., Braffort, A., Glauert, J. R. W., y Sáfár, É. (2012). Dicta-Sign – Building a Multilingual Sign Language Corpus. <https://api.semanticscholar.org/CorpusID:58690434>
- Moryossef, A., Müller, M., y Fahrni, R. (2023). pose-format: Library for Viewing, Augmenting, and Handling .pose Files. <https://arxiv.org/abs/2310.09066>

- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L. E., Simens, M., Aspell, A., Welinder, P., Christiano, P. F., Leike, J., y Lowe, R. J. (2022). Training language models to follow instructions with human feedback. *ArXiv, abs/2203.02155*. <https://api.semanticscholar.org/CorpusID:246426909>
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *ArXiv, abs/1912.01703*. <https://api.semanticscholar.org/CorpusID:202786778>
- Petrovich, M., Black, M. J., y Varol, G. (2022). TEMOS: Generating diverse human motions from textual descriptions. *ArXiv, abs/2204.14109*. <https://api.semanticscholar.org/CorpusID:248476220>
- Plappert, M., Mandery, C., y Asfour, T. (2016). The KIT Motion-Language Dataset. *Big data, 4*, 236-252. <https://api.semanticscholar.org/CorpusID:161488>
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Aspell, A., Mishkin, P., Clark, J., Krueger, G., y Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. *International Conference on Machine Learning*. <https://api.semanticscholar.org/CorpusID:231591445>
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., y Ommer, B. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10674-10685. <https://api.semanticscholar.org/CorpusID:245335280>
- Rowles, C. (2023). TemporalKit. <https://github.com/CiaraStrawberry/TemporalKit>
- Salvador, S., y Chan, P. K. (2004). FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. <https://api.semanticscholar.org/CorpusID:6226669>
- Saunders, B., Camgöz, N. C., y Bowden, R. (2020a). Everybody Sign Now: Translating Spoken Language to Photo Realistic Sign Language Vi-

- deo. *ArXiv, abs/2011.09846*. <https://api.semanticscholar.org/CorpusID:227054383>
- Saunders, B., Camgöz, N. C., y Bowden, R. (2020b). Progressive Transformers for End-to-End Sign Language Production. *ArXiv, abs/2004.14874*. <https://api.semanticscholar.org/CorpusID:216869341>
- Schulder, M., y Hanke, T. (2022). The hamnosys package. <https://ctan.math.utah.edu/ctan/tex-archive/fonts/hamnosys/hamnosys.pdf>
- Shalev-Arkushin, R., Moryossef, A., y Fried, O. (2023). Ham2Pose: Animating Sign Language Notation into Pose Sequences. <https://arxiv.org/abs/2211.13613>
- Skobov, V., y Lepage, Y. (2020). Video-to-HamNoSys Automated Annotation System. *SIGNLANG*. <https://api.semanticscholar.org/CorpusID:219303000>
- Smith, R. (2013). HamNoSys 4.0 User guide. <https://robertsmithresearch.wordpress.com/wp-content/uploads/2012/10/hamnosys-user-guide-rs-draft-v3-0.pdf>
- Stokoe, W. C. (1961). Sign language structure: an outline of the visual communication systems of the American deaf. 1960. *Journal of deaf studies and deaf education*, 10 1, 3-37. <https://api.semanticscholar.org/CorpusID:5948293>
- Stoll, S., Camgoz, N. C., Hadfield, S., y Bowden, R. (2020). Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks. *International Journal of Computer Vision*, 128, 891-908. <https://api.semanticscholar.org/CorpusID:209538936>
- Stoll, S., Camgöz, N. C., Hadfield, S., y Bowden, R. (2018). Sign Language Production using Neural Machine Translation and Generative Adversarial Networks. *British Machine Vision Conference*. <https://api.semanticscholar.org/CorpusID:52288950>
- Sutton, V., y Frost, A. (2009). SignWriting: Sign Languages Are Written Languages. <https://www.signwriting.org/archive/docs6/sw0523-US-SignWritingVAILConference2008.pdf>
- Tevet, G., Gordon, B., Hertz, A., Bermano, A. H., y Cohen-Or, D. (2022). MotionCLIP: Exposing Human Motion Generation to CLIP Space. *European Conference on Computer Vision*. <https://api.semanticscholar.org/CorpusID:247450907>

- Tevet, G., Raab, S., Gordon, B., Shafir, Y., Cohen-Or, D., y Bermano, A. H. (2022). Human Motion Diffusion Model. *ArXiv, abs/2209.14916*. <https://api.semanticscholar.org/CorpusID:252595883>
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., y Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *ArXiv, abs/2302.13971*. <https://api.semanticscholar.org/CorpusID:257219404>
- Touvron, H., Martin, L., Stone, K. R., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D. M., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., ... Scialom, T. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv, abs/2307.09288*. <https://api.semanticscholar.org/CorpusID:259950998>
- Tze, C. O., Filntisis, P. P., Dimou, A.-L., Roussos, A., y Maragos, P. (2023). Neural Sign Reenactor: Deep Photorealistic Sign Language Retargeting. <https://arxiv.org/abs/2209.01470>
- United Nations. (2023). International Day of Sign Languages. <https://www.un.org/en/observances/sign-languages-day>
- Universidad Politécnica de Madrid. (2010). Introducción a HamNoSys v 4.0. https://vhg.cmp.uea.ac.uk/tech/hamnosys/Introduccion%20a%20HamNosys_castellano.pdf
- van den Oord, A., Vinyals, O., y Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. *ArXiv, abs/1711.00937*. <https://api.semanticscholar.org/CorpusID:20282961>
- Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I. (2017). Attention is All you Need. *Neural Information Processing Systems*. <https://api.semanticscholar.org/CorpusID:13756489>
- Walsh, H., Ravanshad, A., Rahmani, M., y Bowden, R. (2024). A Data-Driven Representation for Sign Language Production. <https://arxiv.org/abs/2404.11499>
- Walsh, H., Saunders, B., y Bowden, R. (2022). Changing the Representation: Examining Language Representation for Neural Sign Language Production. *ArXiv, abs/2210.06312*. <https://api.semanticscholar.org/CorpusID:252624661>

- Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Liu, G., Tao, A., Kautz, J., y Catanzaro, B. (2018). Video-to-Video Synthesis. *Neural Information Processing Systems*. <https://api.semanticscholar.org/CorpusID:52049245>
- Williams, W., Ringer, S., Ash, T., Hughes, J., Macleod, D., y Dougherty, J. (2020). Hierarchical Quantized Autoencoders. *ArXiv, abs/2002.08111*. <https://api.semanticscholar.org/CorpusID:211171717>
- Xie, P., Zhang, Q., Li, Z., Tang, H., Du, Y., y Hu, X. (2022). G2P-DDM: Generating Sign Pose Sequence from Gloss Sequence with Discrete Diffusion Model. *AAAI Conference on Artificial Intelligence*. <https://api.semanticscholar.org/CorpusID:251710263>
- Zhang, J., Zhang, Y., Cun, X., Huang, S., Zhang, Y., Zhao, H., Lu, H., y Shen, X. (2023). T2M-GPT: Generating Human Motion from Textual Descriptions with Discrete Representations. <https://arxiv.org/abs/2301.06052>
- Zhang, L., Rao, A., y Agrawala, M. (2023). Adding Conditional Control to Text-to-Image Diffusion Models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3813-3824. <https://api.semanticscholar.org/CorpusID:256827727>
- Zhang, M., Cai, Z., Pan, L., Hong, F., Guo, X., Yang, L., y Liu, Z. (2022). MotionDiffuse: Text-Driven Human Motion Generation With Diffusion Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46, 4115-4128. <https://api.semanticscholar.org/CorpusID:251953565>
- Zhang, Y., Huang, D., Liu, B., Tang, S., Lu, Y., Chen, L., Bai, L., Chu, Q., Yu, N., y Ouyang, W. (2023). MotionGPT: Finetuned LLMs are General-Purpose Motion Generators. *ArXiv, abs/2306.10900*. <https://api.semanticscholar.org/CorpusID:259203911>
- Zhu, D., Chen, J., Shen, X., Li, X., y Elhoseiny, M. (2023). MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. *ArXiv, abs/2304.10592*. <https://api.semanticscholar.org/CorpusID:258291930>

APÉNDICES

Apéndice 1

Diccionarios Parte I

En la Tabla 1.1 se muestra el *nombre HamNoSys* y el *nombre HamNoSys sin prefijo* correspondientes a las descripciones de los símbolos HamNoSys para las modalidades *secuencia HamNoSys* y *secuencia HamNoSys sin prefijo* respectivamente.

Tabla 1.1: Diccionarios Parte I

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
○	hamfist	fist
□	hamflathand	flathand
⌋	hamfingertwo	fingertwo
⌋	hamfingertwothree	fingertwothree
⌋	hamfingertwothreespread	fingertwothreespread
⌋	hamfingertwothreefourfive	fingertwothreefourfive
◁	hampinchnetwo	pinchnetwo
◁	hampinchall	pinchall
▷	hampinchnetwoopen	pinchnetwoopen
▷	hamceenetwo	ceenetwo
≡	hamceeall	ceeall
▷	hamceeopen	ceeopen
↙	hamthumboutmod	thumboutmod

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
˘	hamthumbacrossmod	thumbacrossmod
˙	hamthumbopenmod	thumbopenmod
˚	hamfingerstraightmod	fingerstraightmod
˛	hamfingerbendmod	fingerbendmod
˜	hamfingerhookmod	fingerhookmod
˝	hamdoublebent	doublebent
˞	hamdoublehooked	doublehooked
˟	hamextfingeru	extfingeru
ˠ	hamextfingerur	extfingerur
ˡ	hamextfingerr	extfingerr
ˢ	hamextfingerdr	extfingerdr
ˣ	hamextfingerd	extfingerd
ˤ	hamextfingerdl	extfingerdl
˥	hamextfingerl	extfingerl
˦	hamextfingerul	extfingerul
˧	hamextfingerol	extfingerol
˨	hamextfingero	extfingero
˩	hamextfingeror	extfingeror
˪	hamextfingeril	extfingeril
˫	hamextfingeri	extfingeri
ˬ	hamextfingerir	extfingerir
˭	hamextfingerui	extfingerui
ˮ	hamextfingerdi	extfingerdi
˯	hamextfingerdo	extfingerdo
˰	hamextfingeruo	extfingeruo
◌	hampalmu	palmu
◌	hampalmur	palmur
◌	hampalmr	palmr
◌	hampalmdr	palmdr
◌	hampalmd	palmd
◌	hampalmdl	palmdl
◌	hampalml	palml

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
	hampalmul	palmul
	hamhead	head
	hamheadtop	headtop
	hamforehead	forehead
	hameyebrows	eyebrows
	hameyes	eyes
	hamnose	nose
	hamnostrils	nostrils
	hamear	ear
	hamearlobe	earlobe
	hamcheek	cheek
	hamlips	lips
	hamtongue	tongue
	hamteeth	teeth
	hamchin	chin
	hamunderchin	underchin
	hamneck	neck
	hamshouldertop	shouldertop
	hamshoulders	shoulders
	hamchest	chest
	hamstomach	stomach
	hambelowstomach	belowstomach
	hamneutralspace	neutralspace
	hamupperarm	upperarm
	hamelbow	elbow
	hamelbowinside	elbowinside
	hamlowerarm	lowerarm
	hamwristback	wristback
	hamwristpulse	wristpulse
	hamthumbball	thumbball
	hampalm	palm
	hamhandback	handback

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
1l	hamthumbside	thumbside
5l	hampinkyside	pinkyside
1	hamthumb	thumb
2	hamindexfinger	indexfinger
3	hammiddlefinger	middlefinger
4	hamringfinger	ringfinger
5	hampinky	pinky
ḷ	hamfingertip	fingertip
l̤	hamfingernail	fingernail
l̥	hamfingerpad	fingerpad
l̦	hamfingermidjoint	fingermidjoint
ļ	hamfingerbase	fingerbase
l̨	hamfingerside	fingerside
◻	hamlrbeside	lrbeside
▪	hamlrat	lrat
□	hamcoreftag	coreftag
○	hamcorefref	corefref
↑	hammoveu	moveu
↗	hammoveur	moveur
→	hammover	mover
↘	hammovedr	movedr
↓	hammoved	moved
↙	hammovedl	movedl
←	hammoveul	moveul
↖	hammoveul	moveul
↖̣	hammoveol	moveol
⊥	hammoveo	moveo
↗̣	hammoveor	moveor
↖̥	hammoveil	moveil
↓̣	hammovei	movei
↘̣	hammoveir	moveir
↖̦	hammoveui	moveui

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
↶	hammovedi	movedi
↷	hammovedo	movedo
↘	hammoveuo	moveuo
⊂	hamcircleo	circleo
⊃	hamcirclei	circlei
⊆	hamcircled	circled
⊇	hamcircleu	circleu
⊈	hamcirclel	circlel
⊉	hamcircler	circler
⊊	hamcircleul	circleul
⊋	hamcircledr	circledr
⊌	hamcircleur	circleur
⊍	hamcircledl	circledl
⊎	hamcircleol	circleol
⊏	hamcircleir	circleir
⊐	hamcircleor	circleor
⊑	hamcircleil	circleil
⊒	hamcircleui	circleui
⊓	hamcircledo	circledo
⊔	hamcircleuo	circleuo
⊕	hamcircledi	circledi
✋	hamfingerplay	fingerplay
↕	hamnodding	nodding
↔	hamswinging	swinging
↻	hamtwisting	twisting
↺	hamstircw	stircw
↻	hamstirccw	stirccw
↗	hamreplace	replace
⊗	hamnomotion	nomotion
⌚	hamclocku	clocku
⌚	hamclockul	clockul
⌚	hamclockl	clockl

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
⌚	hamclockdl	clockdl
🕒	hamclockd	clockd
🕒	hamclockdr	clockdr
🕒	hamclockr	clockr
🕒	hamclockur	clockur
⌚	hamclockfull	clockfull
⤵	hamarcl	arcl
⤵	hamarcu	arcu
⤵	hamarcr	arcr
⤵	hamarcd	arcd
〰	hamwavy	wavy
〰	hamzigzag	zigzag
🕐	hamellipseh	ellipseh
🕐	hamellipseur	ellipseur
🕐	hamellipsev	ellipsev
🕐	hamellipseul	ellipseul
↗	hamincreasing	increasing
↘	hamdecreasing	decreasing
*	hamfast	fast
—	hamslow	slow
⚡	hamtense	tense
🛑	hamrest	rest
⏹	hamhalt	halt
🔒	hamclose	close
🤏	hamtouch	touch
🔒	haminterlock	interlock
✕	hamcross	cross
👉	hamarmextended	armextended
👈	hambehind	behind
🪮	hambrushing	brushing
◦	hamsmallmod	smallmod
•	hamlargemod	largemod

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
+	hamrepeatfromstart	repeatfromstart
⦶	hamrepeatfromstartseveral	repeatfromstartseveral
⤵	hamrepeatcontinue	repeatcontinue
⦶⤵	hamrepeatcontinueseveral	repeatcontinueseveral
↶	hamrepeatreverse	repeatreverse
~	hamalternatingmotion	alternatingmotion
(	hamseqbegin	seqbegin
)	hamseqend	seqend
[	hamparbegin	parbegin
]	hamparend	parend
⟨	hamfusionbegin	fusionbegin
⟩	hamfusionend	fusionend
\	hambetween	between
+	hamplus	plus
:	hamsymmpar	symmpar
..	hamsymmlr	symmlr
∅	hamnondominant	nondominant
ⲡ	hamnonipsi	nonipsi
...	hametc	etc
~	hamorirelative	orirelative
⊔	hammime	mime
4.0	hamversionfourzero	versionfourzero
	hamspace	space
!	hamexclaim	exclaim
,	hamcomma	comma
.	hamfullstop	fullstop
?	hamquery	query
{	hamaltbegin	altbegin
	hammetaalt	metaalt
}	hamaltend	altend
↘	hamwristtopulse	wristtopulse
↗	hamwristtoback	wristtoback

Continúa en la siguiente página

Símbolo	Nombre HamNoSys	Nombre HamNoSys sin prefijo
↖	hamwristtothumb	wristtothumb
↗	hamwristtopinky	wristtopinky
↔	hammovecross	movecross
✕	hammoveX	moveX

Apéndice 2

Diccionarios Parte II

En la Tabla 2.1 se muestra la *descripción* y la *descripción refinada* de los símbolos HamNoSys para las modalidades *secuencia con descripciones* y *secuencia con descripciones refinadas* respectivamente.

Tabla 2.1: Diccionarios Parte II

Símbolo	Descripción	Descripción refinada
○	fist	fist
□	flat hand	flat hand
┐	one finger	one finger
┌	two fingers non spread	two fingers non spread
└	two fingers spread	two fingers spread
≡	four fingers spread	four fingers spread
◁	closed pinch between thumb and index - rest of the fingers in fist form	closed pinch between thumb and index and rest of the fingers in fist form
↖	closed pinch between thumb and other fingers	closed pinch between thumb and other fingers
▷	closed pinch between thumb and index - rest of the fingers separated	closed pinch between thumb and index and rest of the fingers separated

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
▷	thumb and index form letter C - rest of the fingers in fist form	thumb and index form letter C and rest of the fingers in fist form
⊖	thumb and other fingers form letter C	thumb and the other fingers form letter C
⊃	thumb and index form letter C - rest of the fingers separated	thumb and index form letter C and rest of the fingers separated
↗	thumb opened outward	thumb opened outward
┐	thumb across	thumb across
↖	thumb extended	thumb extended
└	flat bending	flat bending
⌋	round bending	round bending
┘	sharp bending	sharp bending
⋈	double-bent	double-bent
↘	hooked bending	hooked bending
⤴	extended finger up (parallel to body)	extended finger up - parallel to body
↗	extended finger up and to the right (parallel to body)	extended finger up and to the right - parallel to body
↘	extended to the right (parallel to body)	extended to the right - parallel to body
⤵	extended finger down and to the right (parallel to body)	extended finger down and to the right - parallel to body
↘	extended finger down (parallel to body)	extended finger down - parallel to body
⤵	extended finger down and to the left (parallel to body)	extended finger down and to the left - parallel to body
↖	extended finger to the left (parallel to body)	extended finger to the left - parallel to body
⤴	extended finger up and to the left (parallel to body)	extended finger up and to the left - parallel to body

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
└	extended finger away from the body and to the left	extended finger away from the body and to the left
⤴	extended finger away from the body	extended finger away from the body
┘	extended finger away from the body and to the right	extended finger away from the body and to the right
└	extended finger towards the body and to the left	extended finger towards the body and to the left
⤵	extended finger towards the body	extended finger towards the body
┘	extended finger towards the body and to the right	extended finger towards the body and to the right
┐	extended finger upward and toward the body	extended finger upward and toward the body
┑	extended finger downward and toward the body	extended finger downward and toward the body
┐	extended finger downward and away from the body	extended finger downward and away from the body
┑	extended finger upward and away from the body	extended finger upward and away from the body
◊	palm up	palm up
◊	palm up and right	palm up and right
◊	palm right	palm right
◊	palm down and right	palm down and right
◊	palm down	palm down
◊	palm down and left	palm down and left
◊	palm left	palm left
◊	palm up and left	palm up and left
○	head	head
◌	top of head	top of head
◌	forehead	forehead
~	eyebrows	eyebrows
∞	eyes	eyes

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
⏶	nose	nose
⏷	nostrils	nostrils
⏸	ear	ear
⏹	ear lobe	ear lobe
⏺	cheek	cheek
⏻	lips	lips
⏼	tongue	tongue
⏽	teeth	teeth
⏾	chin	chin
⏿	under the chin	under the chin
⏿	neck	neck
⏿	top of shoulder	top of shoulder
⏿	shoulder line	shoulder line
⏿	chest	chest
⏿	belly line	belly line
⏿	abdominal line	abdominal line
⏿	neutral sapce location (in front of the upper part of the body)	neutral sapce location in front of the upper part of the body
⏿	upper arm	upper arm
⏿	elbow	elbow
⏿	inside elbow	inside elbow
⏿	lower arm	lower arm
⏿	wrist joint (CMC)	wrist joint
⏿	wrist pulse	wrist pulse
⏿	ball of thumb	ball of thumb
⏿	palm	palm
⏿	back of hand	back of hand
⏿	side of thumb	side of thumb
⏿	side of little finger	side of little finger
⏿	thumb	thumb
⏿	index finger	index finger
⏿	middle finger	middle finger

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
4	ring finger	ring finger
5	little finger	little finger
†	fingertip	fingertip
‡	fingernail	fingernail
‡	fingerpad	fingerpad
‡	middle joint of finger (PIP)	middle joint of finger
‡	base joint of finger (MCP)	base joint of finger
‡	side of finger	side of finger
◻	left left side of right right side of	side
▪	right of to the right of left of to the left of	adjacent
□	co-reference tag	co-reference tag
○	co-reference reference	co-reference reference
↑	straight movement up (parallel to body)	straight movement up - parallel to body
↗	straight movement up and to the right (parallel to body)	straight movement up and to the right - parallel to body
→	straight movement to the right (parallel to body)	straight movement to the right - parallel to body
↘	straight movement down and to the right (parallel to body)	straight movement down and to the right - parallel to body
↓	straight movement down (parallel to body)	straight movement down - parallel to body
↙	straight movement down and to the left (parallel to body)	straight movement down and to the left - parallel to body
←	straight movement to the left (parallel to body)	straight movement to the left - parallel to body
↖	straight movement up and to the left (parallel to body)	straight movement up and to the left - parallel to body

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
	straight movement away from the body and to the left	straight movement away from the body and to the left
	straight movement away from the body	straight movement away from the body
	straight movement away from the body and to the right	straight movement away from the body and to the right
	straight movement towards the body and to the left	straight movement towards the body and to the left
	straight movement towards the body	straight movement towards the body
	straight movement towards the body and to the right	straight movement towards the body and to the right
	straight movement upward and toward the body	straight movement upward and toward the body
	straight movement downward and toward the body	straight movement downward and toward the body
	straight movement downward and away from the body	straight movement downward and away from the body
	straight movement upward and away from the body	straight movement upward and away from the body
	circular movement clockwise	circular movement clockwise
	circular movement counter-clockwise	circular movement counter-clockwise
	circular movement down (parallel to body)	circular movement down - parallel to body
	circular movement up (parallel to body)	circular movement up - parallel to body
	circular movement to the left (parallel to body)	circular movement to the left - parallel to body

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
	circular movement to the right (parallel to body)	circular movement to the right - parallel to body
	circular movement up and to the left (parallel to body)	circular movement up and to the left - parallel to body
	circular movement down and to the right (parallel to body)	circular movement down and to the right - parallel to body
	circular movement up and to the right (parallel to body)	circular movement up and to the right - parallel to body
	circular movement down and to the left (parallel to body)	circular movement down and to the left - parallel to body
	circular movement away from the body and to the left	circular movement away from the body and to the left
	circular movement towards the body and to the right	circular movement towards the body and to the right
	circular movement away from the body and to the right	circular movement away from the body and to the right
	circular movement towards the body and to the left	circular movement towards the body and to the left
	circular movement upward and toward the body	circular movement upward and toward the body
	circular movement downward and away from the body	circular movement downward and away from the body
	circular movement upward and away from the body	circular movement upward and away from the body
	circular movement downward and toward the body	circular movement downward and toward the body
	fingerplay	fingerplay

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
↕	nodding (repeatedly moving vertically - up and down)	nodding - repeatedly moving vertically up and down
↔	swinging (repeatedly moving horizontally - right and left)	swinging - repeatedly moving horizontally right and left
↻	lower arm rotation (indefinite number of repetitions)	lower arm rotation - indefinite number of repetitions
↷	single rotation in wrist joint to the right	single rotation in wrist joint to the right
↶	single rotation in wrist joint to the left	single rotation in wrist joint to the left
→	finger movements resulting from handshape change change of handshape and orientation	finger movements resulting from handshape change or a change of handshape and orientation
⊗	no motion	no motion
○	point of the circle up	point of the circle up
◐	point of the circle up and left	point of the circle up and left
◑	point of the circle left	point of the circle left
◒	point of the circle down and left	point of the circle down and left
◓	point of the circle down	point of the circle down
◔	point of the circle down and right	point of the circle down and right
◕	point of the circle right	point of the circle right
◖	point of the circle up and right	point of the circle up and right
⊕	full circumference	full circumference
↪	curved movement towards body curved movement to the left	curved movement towards body or curved movement to the left

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
ˆ	curved movement upwards curved movement away from body	curved movement upwards or curved movement away from body
ˆ	curved movement away from body curved movement to the right	curved movement away from body or curved movement to the right
ˆ	curved movement downwards curved movement towards body	curved movement downwards or curved movement towards body
˜	wavy movement movement	wavy movement movement
˜	zigzag movement	zigzag movement
⦶	elliptic shape horizontal	elliptic shape horizontal
⦶	elliptic shape up and right	elliptic shape up and right
⦶	elliptic shape vertical	elliptic shape vertical
⦶	elliptic shape up and left	elliptic shape up and left
∠	increasing	increasing
∠	decreasing	decreasing
*	fast	fast
—	slow	slow
✕	tense	tense
ʔ	stoppage at start	stoppage at start
	sudden stop	sudden stop
) (	close to	close to
χ	contact with	contact with
⌘	interlocking hands	interlocking hands
×	crossing hands (also for arms - wrists and fingers)	crossing
↗	outstretched arm	outstretched arm
↖	behind	behind
†	brushing movement	brushing movement
◦	small movement	small movement
•	large movement	large movement
+	repeated once	repeated once

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
⋈	repeated a couple of times	repeated a couple of times
⤵	repeated once (2nd iteration starting when first ended)	repeated once - second iteration starting when first ended
⤵⤵	repeated a couple of times (each iteration starting when first ended)	repeated a couple of times - each iteration starting when first ended
↩	repeated in reversed direction	repeated in reversed direction
↔	two-handed sign alternating (hands flip roles)	two-handed sign alternating - hands flip roles
(	sequential action begin	sequential action (
)	sequential action end	)
[	co-temporal action begin	co-temporal action [
]	co-temporal action end	]
⟨	fused sequential actions begin	fused sequential actions<
⟩	fused sequential actions end	>
\	between	between
↯	hand permutation	hand permutation
:	two-handed sign (paralleled but not mirrored)	two-handed sign - paralleled but not mirrored
⋈	two-handed simmetrical sign (hands position mirror one another)	two-handed simmetrical sign - hands position mirror one another
∅	nondominant-hand	nondominant-hand
↯	do not mirror to the non-dominant hand	do not mirror to the non-dominant hand
...	Un(der)specified	Unspecified
~	the orientation of the hand varies constantly during drag	the orientation of the hand varies constantly during drag

Continúa en la siguiente página

Símbolo	Descripción	Descripción refinada
□	place holder for sign-level facial expression	place holder for sign-level facial expression
٤٠	version four zero	version four zero
!	sentence boundary (with some facial expression)	sentence boundary - with some facial expression
,	sequential execution (sequentially performed gestures)	sequential execution - sequentially performed gestures
.	sentence boundary (neutral facial expression)	sentence boundary - neutral facial expression
?	sentence boundary (with some facial expression)	sentence boundary - with some facial expression
{	alternatives begin	alternatives{
	either/or	
}	alternatives end	}
↗	wirst movement bent towards pulse	wirst movement bent towards pulse
↖	wirst movement bent towards hand back	wirst movement bent towards hand back
↗	wirst movement bent towards little finger side	wirst movement bent towards little finger side
↖	wirst movement bent towards thumb side	wirst movement bent towards thumb side
✂	crossing movement	crossing movement
✂	X-movement	X-movement

Apéndice 3

Diccionarios Parte III

En la Tabla 3.1 se muestra el grupo asignado a los símbolos HamNoSys, utilizado para generar las descripciones en la modalidad *descripción en lenguaje natural*.

Tabla 3.1: Dicionarios Parte III

Símbolo	Grupo del símbolo
○	Handshape - Base from
○	Handshape - Base from
㇀	Handshape - Base from
㇁	Handshape - Base from
㇂	Handshape - Base from
㇃	Handshape - Base from
㇄	Handshape - Base from
㇅	Handshape - Base from
㇆	Handshape - Base from
㇇	Handshape - Base from
㇈	Handshape - Base from
㇉	Handshape - Base from
㇊	Handshape - Base from
㇋	Handshape - Thumb position
㇌	Handshape - Thumb position
㇍	Handshape - Thumb position

Continúa en la siguiente página

Símbolo	Grupo del símbolo
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement
	Movement

Continúa en la siguiente página

