UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Thiago Malta Coutinho

Learning a Discrete Intermediate Representation for Continuous Sign Language Production

Belo Horizonte
2023

Thiago Malta Coutinho

Learning a Discrete Intermediate Representation for Continuous Sign Language Production

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Erickson Rangel do Nascimento
Coorientador: Thiago Luange Gomes

Belo Horizonte
2023

Coutinho, Thiago Malta.

C871l Learning a discrete intermediate representation for continuous sign language production [recurso eletrônico] / Thiago Malta Coutinho – 2023.
1 recurso online (68 f. il., color): pdf

Orientador: Erickson Rangel do Nascimento.
Coorientador: Thiago Luange Gomes.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 60 - 68.

1. Computação – Teses. 2. Visão por computador – Teses. 3. Processamento de Linguagem Natural – Teses. 4.Linguagem e línguas - Teses. 5. Linguagem de sinais – Teses.
I. Nascimento, Erickson Rangel do. II. Gomes, Thiago Luange. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. IV. Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Irenquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais – ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

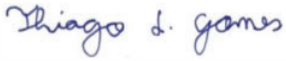
FOLHA DE APROVAÇÃO

Learning a Discrete Intermediate Representation for Continuous Sign Language Production

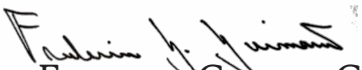
THIAGO MALTA COUTINHO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ERICKSON RANGEL DO NASCIMENTO - Orientador
Departamento de Ciência da Computação - UFMG



PROF. THIAGO LUANGE GOMES - Coorientador
Departamento de Informática - Universidade Federal de Viçosa



PROF. FREDERICO GADELHA GUIMARÃES
Departamento de Ciência da Computação - UFMG

Prof. MICHEL MELO DA SILVA
Departamento de Informática - Universidade Federal de Viçosa

Belo Horizonte, 22 de dezembro de 2023.

This work is not mine alone. I dedicate it to the community of friends, family, and mentors who built the foundation upon which this work stands.

Agradecimentos

A finalização desta dissertação marca a conclusão de uma importante etapa para mim. Este percurso, que envolveu desafios e aprendizados, felizmente, não foi uma caminhada solitária. Por essa razão, gostaria de expressar minha gratidão a todos que contribuíram para a realização deste trabalho.

Agradeço à minha família, em especial aos meus pais, Tácilo e Karla, meus avós, Vicente e Sônia, e aos meus irmãos, Victor e Daniel, pelo apoio e incentivo incondicional. Vocês são a base de tudo. Sem vocês, nada disso seria possível.

Ao meu orientador, Professor Erickson Rangel do Nascimento, agradeço pelos valiosos ensinamentos, por me receber no Verlab, um lugar de pessoas excepcionais, e também por sempre incentivar o desenvolvimento e a melhora contínua por meio de grandes desafios. Agradeço também pela compreensão e paciência em momentos difíceis.

Ao meu coorientador, Professor Thiago Gomes Luange, agradeço pela disponibilidade e colaboração. Suas sugestões e conselhos sempre vieram em momentos muito oportunos. Suas ideias contribuíram significativamente para esse trabalho e para meu aprendizado.

A Larissa Reis, minha companheira, agradeço por todas as conversas e apoio durante os tempos mais difíceis e também por tornar mais alegres os momentos felizes compartilhados.

Aos amigos e colegas de programa, agradeço pelo companheirismo e pela troca de experiências que tornaram a jornada muito mais rica. Em especial, gostaria de agradecer ao meu amigo Rafael Vieira, que compartilhou essa jornada desde a graduação e tornou tudo muito mais leve.

Expresso também minha gratidão ao corpo técnico-administrativo da Universidade, sempre muito solícitos e profissionais de extrema qualidade. Esta pesquisa não seria possível sem o fomento concedido pelas agências CNPq, CAPES, FAPEMIG e FINEP, às quais também registro meu agradecimento.

Por fim, agradeço à Universidade Federal de Minas Gerais pela oportunidade e pela qualidade do ensino oferecido. A UFMG é feita por pessoas excepcionais, e ter passado esses anos nesse ambiente me tornou uma pessoa e um profissional muito melhor.

A todos os mencionados, e a outros que de alguma forma contribuíram para este percurso, meu sincero agradecimento.

"*Science is more than a body of knowledge, it's a way of thinking, a skeptical way of interrogating the universe, with a full awareness of human fallibility.*"

(Carl Sagan)

Resumo

Centenas de milhões de pessoas sofrem de algum tipo de perda auditiva mundialmente. A Organização Mundial da Saúde (OMS) estima que esse grupo seja de aproximadamente 5% da população mundial. A Linguagem de Sinais é o principal meio de comunicação desses indivíduos. Atualmente, existe uma escassez de intérpretes profissionais de língua de sinais em todo o mundo, o que leva a uma má integração dos usuários de língua de sinais na sociedade em geral. A Produção de Linguagem de Sinais (PLS) é uma tarefa que pode ajudar nesse problema através da síntese automática de linguas de sinais. As áreas de Visão Computacional e Processamento de Linguagem Natural (PLN) fizeram avanços significativos na síntese de gestos e linguagem recentemente, proporcionando novas possibilidades para a PLS. No entanto, os modelos existentes ainda têm dificuldade em representar com precisão e compreensibilidade os movimentos da língua de sinais. Esta dissertação apresenta uma nova abordagem que utiliza uma representação intermediária discreta-contínua para gerar frases de língua de sinais de alta qualidade exclusivamente a partir de entradas de texto. O método utiliza a arquitetura *Transformers*, amplamente utilizada em PLN, para extrair representações textuais de Modelos de Linguagem de Larga-escala (MLL). Ao contrário das abordagens anteriores, que se concentram principalmente no uso de representações latentes contínuas, nosso método explora a natureza discreta do texto e dos sinais para capturar melhor as nuances da língua de sinais. O estudo investiga os benefícios do uso da Quantização Vetorial Residual em um esquema de aprendizado não-supervisionado para otimizar um modelo que sintetiza sinais contínuos a partir de *tokens* discretos. Além disso, uma arquitetura *Transformer Decoder* é empregada para mapear representações textuais para o espaço discreto-contínuo. Nossa abordagem é avaliada em dois conjuntos de dados (em lingua alemã e inglês americano). Os experimentos demonstram a eficácia da abordagem, superando métodos estado da arte em métricas de linguagem (BLEU e ROUGE) e métricas de movimentos (FGD e MAEJ). Esses resultados indicam que o nosso modelo sintetiza sinais mais próximos do esperado tanto espacialmente quanto semanticamente. As contribuições desta dissertação incluem a introdução de um novo modelo discreto-contínuo para gerar frases de língua de sinais de alta qualidade, um método que aproveita o poder dos MLLs para produção de texto para sinais, uma arquitetura *Transformer* que combina representações contínuas e discretas para aprimorar a geração de frases de língua de sinais, e um procedimento experimental extenso e estudo de ablação para validar a eficácia do método proposto.

Palavras-chave: Produção de língua de sinais; Quantização Vetorial Residual; *Auto-Encoder* Variacional; Modelos de Linguagem de Larga escala; modelos generativos.

Abstract

Hundreds of millions of people suffers from some form of hearing loss worldwide. The World Health Organization (WHO) estimates that this group comprises approximately 5% of the global population. Sign language is the primary means of communication for these individuals. Currently, there is a shortage of professional sign language interpreters worldwide, leading to poor integration of sign language users into society at large. Sign Language Production (SLP) is a task that can help address this issue through automatic sign language synthesis. The fields of Computer Vision and Natural Language Processing (NLP) have made significant advancements in gesture and language synthesis recently, offering new possibilities for SLP. However, existing models still struggle to accurately and comprehensibly represent sign language movements. This dissertation presents a new approach that uses an intermediate discrete-continuous representation to generate high-quality sign language sentences exclusively from text inputs. The method employs Transformers architecture, widely used in NLP, to extract textual representations from Large Language Models (LLMs). Unlike previous approaches that mainly focus on the use of continuous latent representations, our method explores the discrete nature of text and signs to better capture sign language nuances. The study investigates the benefits of using Residual Vector Quantization in an unsupervised learning scheme to optimize a model that synthesizes continuous signs from discrete tokens. Additionally, a Transformer Decoder architecture is employed to map textual representations to the discrete-continuous space. Our approach is evaluated on two datasets (in German and American English). The experiments demonstrate the effectiveness of the approach, surpassing state-of-the-art methods in language metrics (BLEU and ROUGE) and movement metrics (FGD and MAEJ). These results indicate that our model synthesizes signs closer to what is expected both spatially and semantically. The contributions of this dissertation include the introduction of a new discrete-continuous model for generating high-quality sign language sentences, a method that leverages the power of LLMs for text-to-sign production, a Transformer architecture that combines continuous and discrete representations to enhance sign language sentence generation, and an extensive experimental procedure and ablation study to validate the effectiveness of the proposed method.

Keywords: Sign language production; Residual Vector Quantization; Variational Auto-Encoder; Large Language Models; generative models.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

According to the World Health Organization (WHO), nearly 5% of the world's population requires rehabilitation to address their disabling hearing loss [75]. This includes individuals who are deaf, hard of hearing, or have some form of hearing loss. Sign language is a valuable tool that enables deaf and hard-hearing individuals to communicate. These languages are expressed through hand signals, gestures, facial expressions, and body language [76], but they do not have a standard written form. Despite being the primary means of communication for millions of people worldwide, there is a severe shortage of professional sign language interpreters globally and few to none technologies are designed to support sign langauge. This leads users of sign language to poor integration into society [7] and severely implicates the assistance the deaf community receives from public and private sector service providers, especially in health services [1].

Sign Language Production aims to translate written text into sign language. The primary objective of this process is to generate sign language sentences that are clear, accurate, and easily interpretable by members of the deaf community. This is a challenging objective. Sign language have their own grammar and lexicon, are not universal, not mutually intelligible with each other and are not a sign form of spoken languages [61], *e.g.*, LIBRAS is not a sign form of Portuguese. Figure 1.1 illustrates a signer performing an American Sign Language (ASL) sentence and its correspondent written text. Note that signs are composed of complex manual and non-manual components. The manual components involve hand gestures that correspond to words or concepts in oral languages, while the non-manual components—such as facial expressions, head movements, and body posture—play a crucial role in conveying grammatical structures, emphasizing certain signs, marking sentence boundaries, and indicating pronouns, negation, or other syntactic and semantic features essential for accurate communication.

Figure 1.1: **American Sign Language Example**. Frames extracted from a video of a person performing American Sign Language. The corresponding written text is shown in above the frames. Please note how a sign sentence can have complex movements involving torso, hands and facial expressions.

"You can build snowballs just fine but they're not really waterproof
and you probably would want to be able to add something waterproof
that you could keep your hands dry and therefore warm."

Source: Adapted from Duarte *et al*. [18]

Figure 1.2: **Example of gloss in American Sign Language.** An english sentence and its correspondence in American Sign Language gloss. Note that in gloss there is no "is" and symbols are always in all caps. The line above words marks a facial expression, in the current phrase it indicates that the signers performs an eyebrows up expression. The "fs" element indicates fingerspelling, followed by each letter with dashes in between.

English: My name is Barbara.

poss-eyebrows up
GLOSS: ME NAME fs-B-A-R-B-A-R-A

Source: Author.

1.1 Contextualization

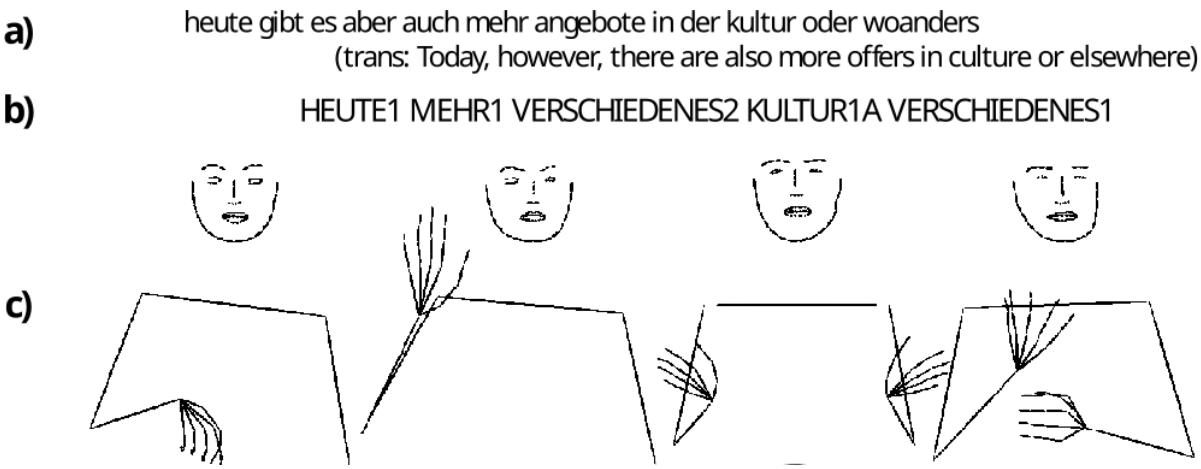
The fields of Computer Vision and Natural Language Processing (NLP) have made significant strides forward, revolutionizing understanding and generation of gesture [54, 79] and language [33, 40]. These advancements bring new possibilities for tackling the challenges in sign language communication, in particular, the Sign Language Production (SLP) [6, 80].

Currently, there are two main types of data used to synthesize signs, text, and glosses[1] (Figure 1.2). The first one being more abundant and easier to obtain whilst the second requires a person with knowledge in both the sign language and written text, making more time consuming and financially difficult to obtain this type of data. Although text data is more accessible, it does not has a direct or linear relationship with signs, *e.g.*, we cannot assume that a text phrase translated into signs will have a one-to-one correspondence, keep the same order of elements or the same local meaning. On the other hand, glosses have a linear relationship with signs. Due to this characteristics of the data, the proposed methods in the area generally uses both text and glosses in their solutions or only glosses.

Early models often produced choppy movements, which were difficult for members of the deaf community to interpret accurately [25, 20, 68, 86]. With the introduction of new techniques such as the attention mechanism and adversarial training, the latest models have achieved much smoother and more natural movements [64, 63]. Although there were remarkable advances in the Sign Language Production field, the proposed methods still under-represent movements, which decreases the accuracy and comprehensibility of the generated sign language sentences. Recent methods propose using intermediate representations to capture the nuances of sign language. A common approach is to model the representation of signs and text as continuous data [62, 32], *e.g.*, Gaussian distributions. However, texts or glosses [66, 67] are discrete, and this strategy can lead to limitations in the accuracy and naturalness of the resulting sign language sentences.

[1] Glosses are defined as written or typed approximation of signs.

Figure 1.3: **Sign Language Production pipeline.** Figure a) shows a example of a input text in german alongside with the english translation; b) its corresponding gloss[1] translation; c) illustrates the signs produced by a model.



Source: Saunders *et al*. [67]

1.2 Problem Definition

This thesis aims to improve Sign Language Production (see Figure 1.3) exclusively from text by proposing a new method that takes into account the discrete and continuous nature of text and signs, respectively.

Recently, the Natural Language Processing (NLP) community has seen great advances based on the Transformers [73] architecture. The attention mechanism present in the method enables the model to learn correspondences between two sequences with non-monotonic relationship. An example is the field of Neural Machine Translation (NMT), where Large Language Models (LLMs) [17, 44, 14, 8, 51] based on Transformers has achieved state-of-the-art for text-to-text translation. A common approach is to use LLMs as backbones to new sub-tasks since these models are trained on a large corpus that is difficult to obtain. As best to our knowledge, these models were not leveraged in Sign Language Production until now.

Since Sign Language Production aims to synthesize a different type of data, only the encoder features from LLMs can be leveraged. Although some recent works in SLP proposed a decoder architecture to synthesize signs [66, 67], these architectures create a continuous latent space, disregarding the discrete nature of text. Lately, models that use Vector Quantization (VQ) have shown a great capability of modeling, creating powerful generative models [72, 59, 21, 82]. These models are able to model a discrete input into a discrete space and synthesize high quality outputs from it. Due to the discrete nature of text, has a natural fit with the SLP decoding task.

In this thesis, we propose a new method that learns a discrete-to-continuous intermediate representation in a unsupervised manner and relies only on text as input data. The learning process is divided into two steps: first we optimize a model that synthesize continuous signs

from discrete tokens; then learn how to use this latent space to generate signs based on text inputs.

1.3 Thesis Statement

We argue that although human movements are continuous, text and signs are composed of a discrete set of symbols, *e.g.*, despite the sign "home" is represented by a continuous human motion, the word "home" and its overall meaning are discrete (the existence of a gloss "HOME" corroborates it). Hence, we propose a model that uses an intermediate representation capable of producing continuous human motion from a discrete representation. This approach naturally fits this discrete-continuous setting, enabling better sign language production.

1.4 Contributions

This work focus on the problem of Text-to-Sign translation (an alluring example is depicted in Figure 1.4). We aim to improve SLP by first learning a powerful discrete-to-continuous representation of Signs without supervision, then leveraging Large Language Models (LLMs) to extract text representations and use a Transformer Decoder architecture [73] to learn a mapping function of text representations to the discrete-to-continuous space. Our training methodology uses a learning strategy that explores the benefit of two types of representations, quantized vectors (continuous) and their corresponding codes (discrete). To evaluate our approach, we conduct extensive experiments and an ablation study on two datasets of varying languages, *i.e.*, Phoenix14T [9] (in Germany) and How2Sign [18] (in American English). Our approach outperforms state-of-the-art methods [64, 32] in terms of language metrics (BLEU and ROUGE) and movements metrics (FGD and MAEJ).

This thesis presents the following contributions to the field of Sign Language Production:

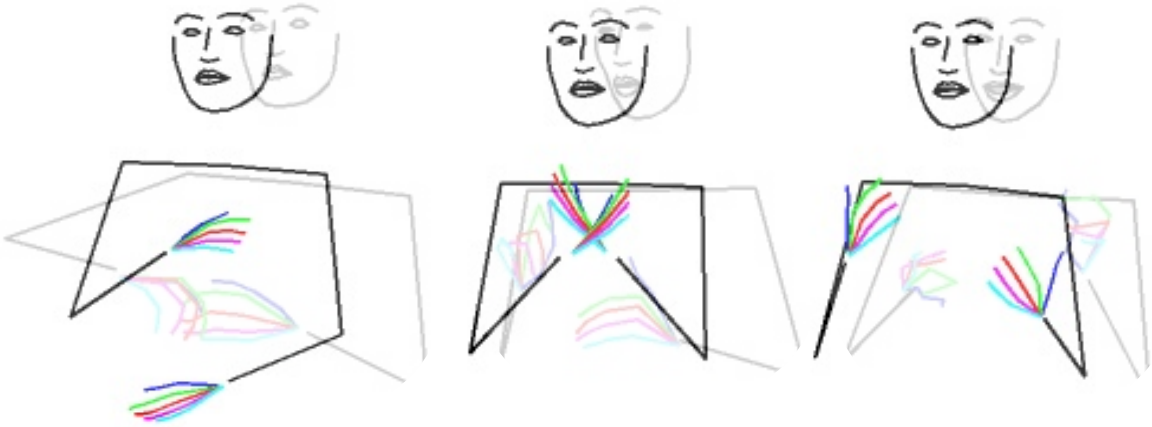
1. We introduce a novel discrete-to-continuous model that generates high-quality sign language sentences;

2. We propose a novel method that leverages the power of LLMs to extract meaningful text representations for direct text-to-sign production;

Figure 1.4: **Synthesis of sign language sentences.** Given a sentence of written text, our method synthesizes representative gestures. Skeletons shown in bold were automatically generated by our method and the ground truth is depicted as shaded skeletons.

Im süden und südwesten gebietsweise regen sonst recht freundlich
(In the south and south-west there will be some rain otherwise quite friendly)

Massey is putting such an extreme amount of English on the cue ball that it's going to slide around what's in your way to hit the ball you're going for now



Source: Author.

3. A new Transformer architecture combining continuous and discrete representations that improves the quality of sign language sentence generation;

4. An extensive experiment procedure and ablation study demonstrating the effectiveness of our method.

We make our code and our trained models weights publicly available to make our experiments reproducible and also contribute to the advance of research in the field.

1.5 Dissertation Outline

This dissertation is organized as follows: ***i)*** Chapter 2 gives theoretical background on Transformers models and Vector Quantization. ***ii)*** Chapter 3 discusses the related works in the field presenting their approach and contributions; In the ***iii)*** Chapter 4, we present our methodology in detail; ***iv)*** Chapter 5 describes our experiments procedure with the metrics, quantitative and qualitative results with further discussions and analysis; ***v)*** Chapter 6 shows our conclusions and outline future works that could potentially improve our work.

Chapter 2

Theoretical Background

In this chapter, we present the theoretical background to comprehend the following chapters of this dissertation. We overview key concepts related to Neural Machine Translation, Human Motion Synthesis, and Vector Quantization.

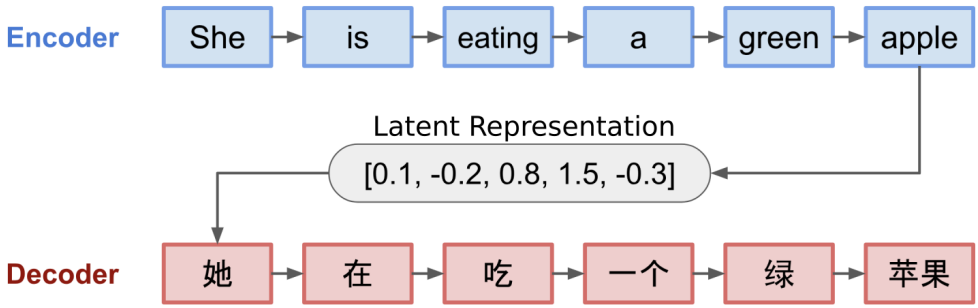
2.1 Neural Machine Translation

Neural Machine Translation (NMT) is a task that aims to translate one language to another by using a Deep Neural Network (DNN). The dominant architecture solution for this problem is the sequence-to-sequence [70] model, which predominantly uses the encoder-decoder [13] scheme (see Figure 2.1). The Encoder-Decoder has two main stages: first the input is encoded into a latent intermediate representation which is decoded into the output sequence by the Decoder. This process of encoding latent representations to decode them in a later moment is a fundamental concept in deep learning that enables models to learn meaningful abstractions from data. Intermediate representations comprises high-level abstract concepts of the underlying structure of the raw data, leading to a better generalization and robustness to noise, reducing the model overfitting. Moreover, latent representations can sometimes be more interpretable than the raw data, making it easier to understand what features the model is learning and how it is making decisions.

In the process of text-to-text translation, the input and output are discrete. However, deep neural networks (DNNs) require continuous operations for learning. To enable DNNs to learn from this type of data, the text undergoes a transformation through an Embedding layer. This layer maps the indexes of words to continuous and trainable vectors, establishing correspondences between them. The final layer of the model matches the dimension of the vocabulary in the translated language. In order to generate a word as the output, the Softmax function is applied to the model's output. The word corresponding to the index with the highest value is selected as the output.

In addition, it is common practice to include tokens at the beginning and end of the input

Figure 2.1: **Encoder-decoder architecture in NMT.** The input text is encoded to a latent representation by the Encoder, this representation is decoded by the Decoder into the desired translated language.



Source: Adapted from Lilian Weng [74].

sentence. These tokens, known as Beginning-of-Sequence (BOS) and End-of-Sequence (EOS), serve as markers. They help condition the Decoder to initiate the decoding process and also indicate where the translation ends in the output phrase.

2.1.1 Transformers

The Transformer model [73] has made a breakthrough in the NMT community, being the majority of state-of-the-art models based on them. The model has a encode-decoder architecture that uses the multi-head attention (MHA) mechanism as its principal component, its architecture is presented in Figure 4.3. Each attention module has three entries: query, key, and value. The main idea is that the model produces a value conditioned on the key and query inputs. This mechanism is often compared to a Database, where a Query will select a Key which will return a correspondent Value. Equation 2.1 presents how to compute the attention:

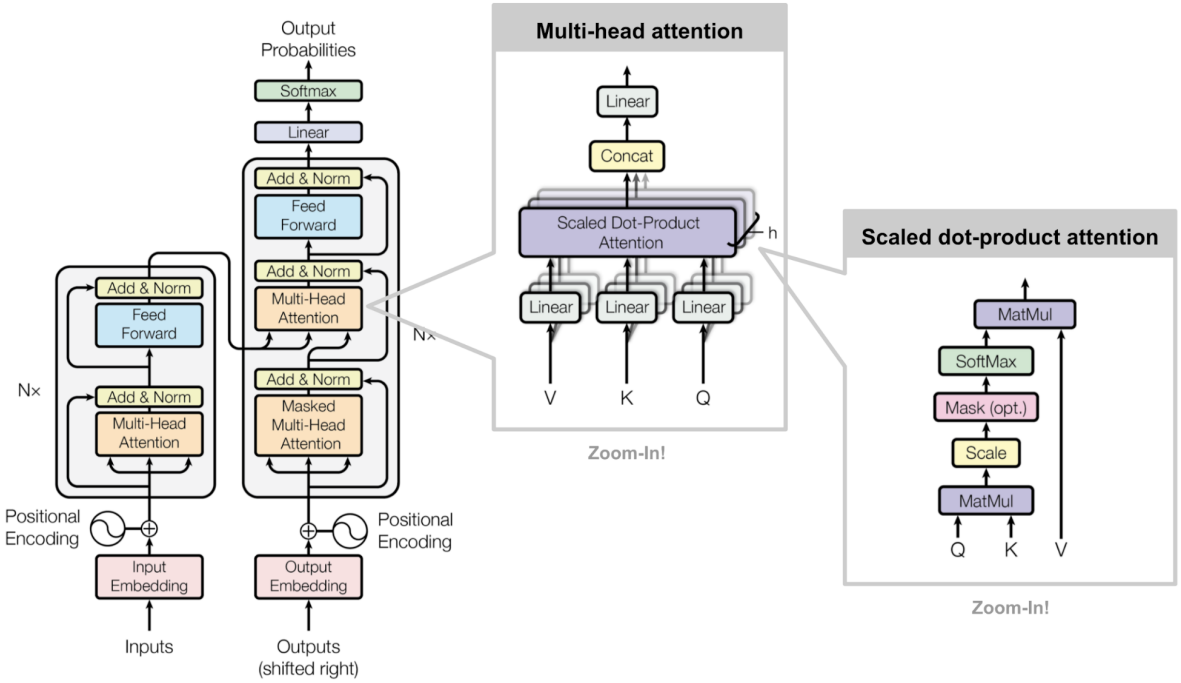
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.1}$$

where d_k is the dimension of the key vector.

The Encoder is composed of multi-head *self-attention* blocks that models the relationship between the inputs and project them into a latent space. The Decoder uses a multi-head self-attention layer to represent the inputs and then a cross-attention layer to combine its inputs representations and the latent representation coming from the Encoder. All inputs are passed through a Positional Encoder that adds a temporal encoding to the data, which helps the attention mechanism to model context based on the appearing order of words in a sentence.

The first Transformer model [73] predicts each output step at a time, concatenated it to

Figure 2.2: **Transformers architecture.** The leftmost block illustrates the encoder-decoder blocks composed of Multi-head attention layers. The MHA block is detailed in the middle figure. The Scaled dot-product attention (Equation 2.1) used in the MHA block is presented at the rightmost block.



Source: Lilian Weng [74]

the previous outputs and feed it to the decoder, in a approach called auto-regression. During training, all inputs and outputs are fed into the network, and attention masks are used to ensure the decoder only attends to past and current tokens, preventing it from accessing future outputs. In this setup, the decoder receives ground truth samples from previous steps and predicts the next token, a process known as Teacher Forcing. At test time, however, the decoder generates outputs sequentially, using its own previous predictions as inputs.

Another possible approach is to use a non-autoregressive inference, where all the decoder inputs are provided first hand and no masks are used in the attention module. In this scheme, all outputs are produce at once. Both methods has its upsides and downsides. When doing auto-regression in a model trained with teacher enforcing the small errors produced at each output can propagate [4], resulting in a problem named *exposure bias* [57] that degrades the model performance. Although a non-autoregressive decoder does not suffer from these problems, it can be very complex to predict all outputs at once.

Previous to Transformers, the state-of-the-art in NLP were composed of RNNs [27] and LSTMs [28]. These models processed the inputs in a sequential manner, making parallel training impossible and also adding a architectural bias toward processing sequences in one specific direction (left-to-right or right-to-left). The main advantage of the Transformers comes from the use of the attention mechanism with the positional embeddings, which enables parallel computation and removes the direction bias. Also, attention scores in Transformers can be visualized

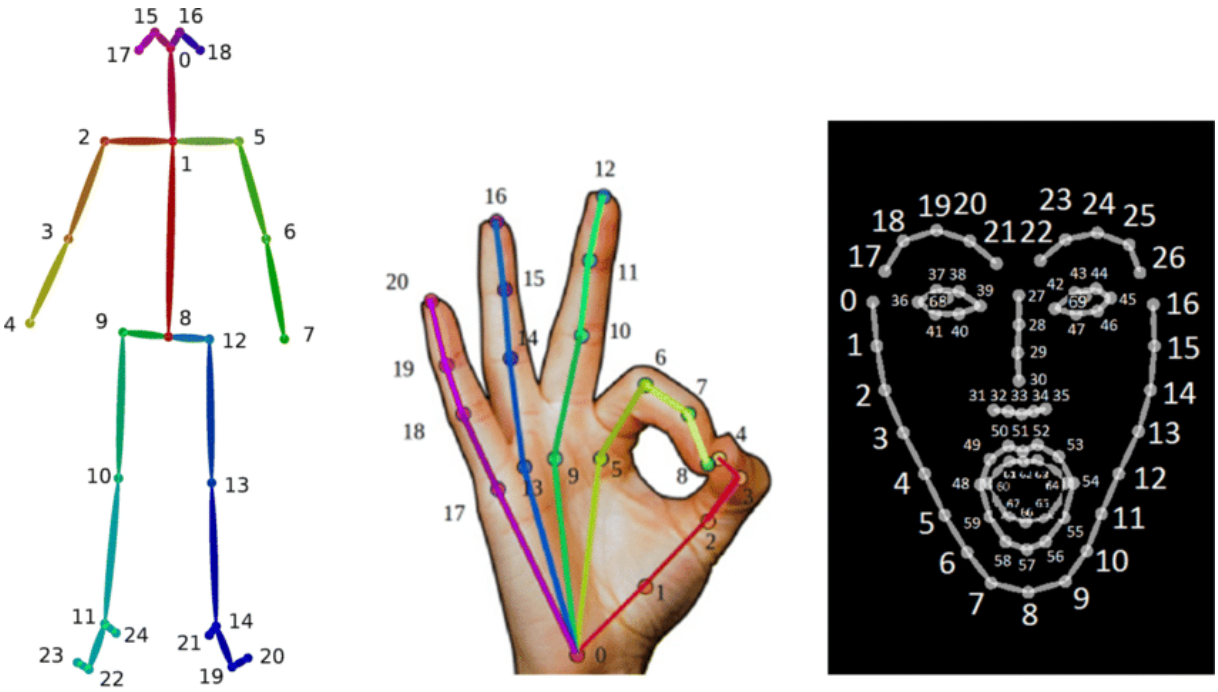
to gain insights into which parts of the input the model is focusing on when making predictions. This interpretability is crucial for understanding the model's behavior and debugging.

Despite being originally designed for text data, Transformers have been adapted for multimodal tasks, such as image captioning [2, 48, 29] and video summarization [49], by incorporating multiple data modalities into the model's input and extending the self-attention mechanism to consider cross-modal relationships.

2.2 Human Motion Synthesis (HMS)

Human motion synthesis (HMS) involves creating realistic and natural-looking movements of human-like characters, often used in animation, robotics, video games, and virtual reality. Producing human movements is a challenging task. A realistic movement has to take into account the human anatomy, considering the body joint angles, its positions in space and also the velocities applied. This implies a temporal coherence of the movement, which involves a motion planing strategy. In addition, human motion is influenced by several factors such as a persons age, cultural background or by the information being conveyed, like in sign languages.

Figure 2.3: **OpenPose skeleton keypoints**. Given an RGB image (left), the OpenPose [12] model can estimate the position of up to 135 keypoints of the human body (right).



a) Body b) Hand c) Face

Source: Adapted from Zabala *et al.* [84]

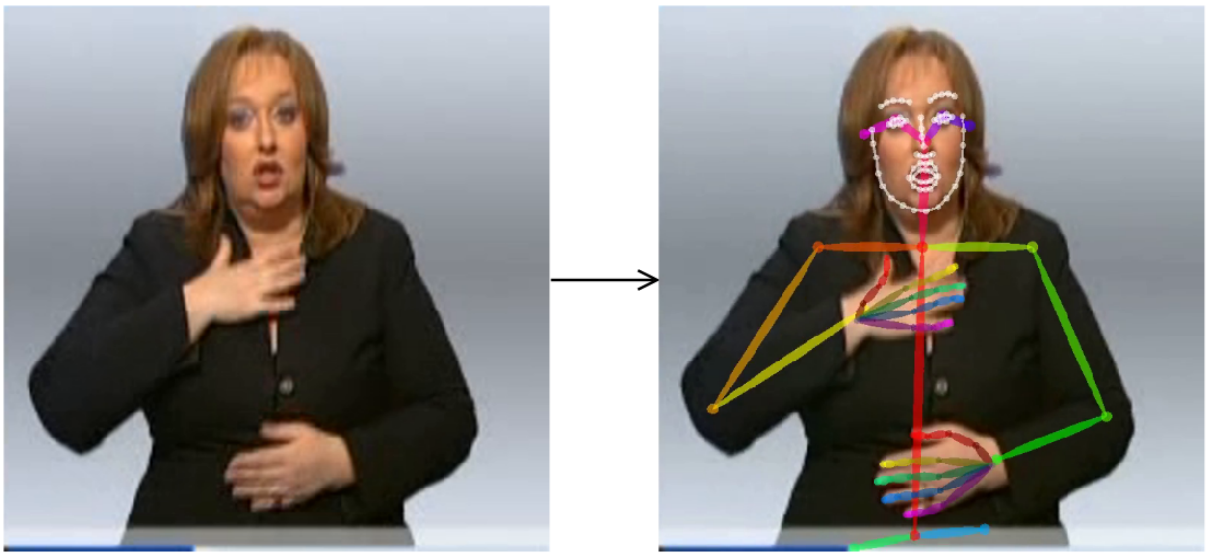
Synthesizing realistic human motions can be a expensive and time consuming task which involves specialists in the area. Several factors could be considered such as skeleton kinematics and dynamics, motion planning and a character characteristics and individuality. In the recent years, deep learning techniques has become a promising approach due to its increasing capability to synthesize realistic movements at a much lower cost.

2.2.1 Data Collection

The data collection step is crucial to enable the usage of learning-based methods. These algorithms needs human movements to learn how to synthesize new ones. There are several models to represent the human body, being the SMPL [45] and a skeleton composed of body joints [12, 3, 43] the most commonly used in the literature.

Using deep learning models to synthesize human motion requires large datasets of annotated human poses. Manually annotating these datasets is a time consuming task. The advances in Human Pose Estimation models made possible create large datasets by using models such as OpenPose [12], a high accurate model capable of estimating up to 135 human body keypoints from a single RGB image. The model uses 25 body keypoints, 42 hands keypoints (21 for each hand) and 68 for the face. Figure 2.3 describes the OpenPose detection keypoints of the body (a), hand(b) and face(c). In the context of SLP, the lower body keypoints (from 8 to 24) are not used since they do not convey information; totaling 120 keypoints. Figure 2.4 illustrates the process of obtaining a sign pose from an image.

Figure 2.4: **Human Pose Estimation**. Given an RGB image (left), the OpenPose [12] model can estimate the position of up to 135 keypoints of the human body (right).



Source: Saunders *et al*. [64]

2.2.2 Deep learning in Human Motion Synthesis

We provide in this section a concise literature review on the Deep Learning techniques in the field of Human Motion Synthesis. Our goal is to present some of the recent models and illustrate how the Transformers architecture is being used alongside with learned latent representations.

Yan *et al*. [77] synthesized dance movements using a Spatial-Temporal Graph Convolutional Network (ST-GCN) [78] and Gaussian Processes [58]. While these movements captured the human motion structure, they needed a seed movement to be realistic and when generated unconditionally, the movements had low temporal correlation. To introduce more information about the movement to be generated, Ferreira *et al*. [23] use the style of a music to condition a Generative Adversarial Network (GAN) [26] and generate realistic dance movements.

More recent approaches are leveraging the capacity of the Transformers to model temporal dependency and using it to synthesize human movements. Li *et al*. [41] use a seed movement and extracted music features to a Transformers network which predicts the next movements of the dancer. In a different proposition, Petrovich *et al*. [55] use a Transformer as backbone to a Variational Auto-Encoder (VAE) [36] that learns a gaussian latent representation of movements conditoned on classes of actions. At inference time this latent space can be conditioned with the class action and sampled from to generate realistic movements. Neither of these approaches can be directly applied to sign language production, as the former requires a predefined seed movement at inference time, while the latter depends on predefined action classes, both of which are not readily available in a sign language generation setting.

A more close method to ours is the work of Ghosh *et al*. [24], where it is proposed a encoder-decoder architecture to predict human motion from a textual description. The authors preprocess the text using a pretrained language model named BERT [17] and also split the skeleton modeling into a hierarchical structure composed of upper and lower body parts.

2.3 Vector Quantization

When learning a latent representation, different approaches can be used. The choice of the architecture is mostly guided by the problem characteristics, aiming to generate more meaningful latent representations. The most simple Encoder-Decoder architecture is the Auto-Encoder [38] model. The model learns a latent representation using an Encoder and a Decoder without any regularization in the latent space, leading to less generalization, interpretability and

robustness. To tackle these problems, Kingma and Welling proposed a probabilistic regularized model, the Variational Auto-Encoder (VAE) [36]. Oord *et al*. later proposed its discrete variation, the Vector-Quantized Variational Auto-Encoder (VQ-VAE) [72], a model more adapted to work with discrete data.

In this section we provide a theoretical background to understand how Vector Quantization works and how to use it to learn a discrete representation of data in a unsupervised way. We first present the Variational Auto-Encoder framework in Section 2.3.1 and then the Vector Quantize (discrete) version of it in Section 2.3.2.

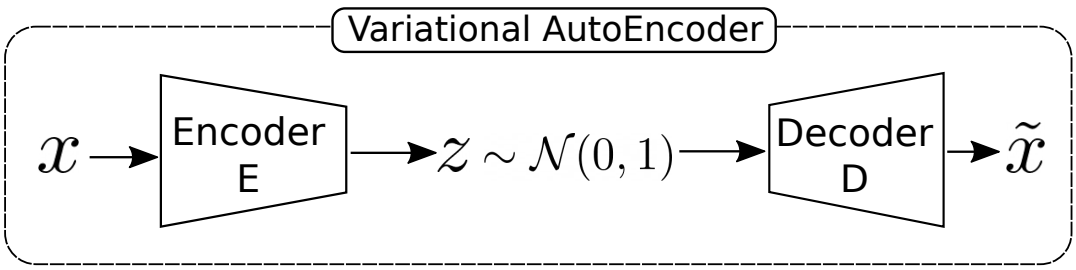
2.3.1 Variational Auto-Encoder (VAE)

Given an input data X, the Variational Auto-Encoder (VAE) [36] model aims to encode this data into a learned higher-dimensional space Z and then reconstruct it to the output $\tilde{X}$. This process occurs by using a encoder-decoder architecture composed of neural networks as mapping functions from the input to the latent space and from the latent space to the reconstruction. The encoder E learns the distribution probability $p_\theta(z|x)$ of the latent space Z given the input X and the decoder D learns the distribution $p_\phi(x|z)$ of the data given the latent space. This configuration is illustrated in Figure 2.5.

Besides reconstructing the input, the model also regularizes the latent space to try to ensure that close samples in the input space also lie close in the learned representation. This is done by imposing a condition that the distribution of the latent space $p(z)$ is a isotropic Gaussian $\sim \mathcal{N}(0, 1)$. The model loss function is presented in Equation 2.2:

$$\mathcal{L}_{VAE} = -\mathbb{E}_{z\sim p(z|x)} \left[\log p_\phi(x|z)\right] + D_{KL} \left(p_\theta(z|x)||p(z)\right), \tag{2.2}$$

Figure 2.5: **VAE model.** The input data x is encoded into a latent space z by the encoder E that models the distribution $p_\theta(z|x)$. This latent space is then decoded into the reconstruction of x, $\tilde{x}$, by the Decoder D which models the distribution $p_\phi(x|z)$.



Source: Author.

where the first term optimizes the expected log likelihood of the data given the latent space and the second term is the KL-divergence between the learned distribution and the desired distribution $p(z)$, which acts as a regularizer term that forces the latent space to be approximately Gaussian. Since the decoder is a deterministic function, we can write the function $p_\phi(x|z)$ as $p_\phi(x|\tilde{x})$; when the input data is continuous, a common assumption made about this distribution is that its Gaussian. From this observation we can derive that the log likelihood can be rewritten as the Mean Squared Error (MSE) between the input and the reconstruction, as shown in Equation 2.3:

$$\begin{aligned} \log p_\phi(x|z) = \log p_\phi(x|\tilde{x}) &\sim \log \exp\left(-(x - D(z))^2\right) \\ &\sim -(x - D(z))^2 \\ &= -(x - \tilde{x})^2. \end{aligned} \tag{2.3}$$

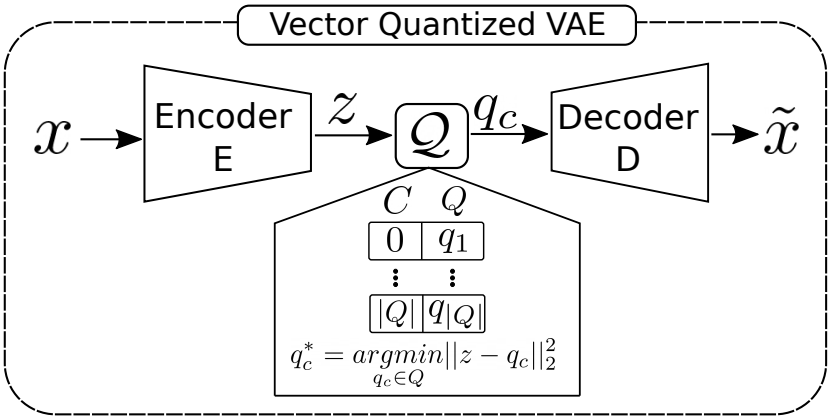
The Variational Auto-Encoder model proposes to regularize the latent space by minimizing the KL-divergence between the latent distribution and a Gaussian distribution. This approach produces a more smooth representation, which leads to better generalization, robustness and also makes it more interpretable since similar samples tends to be more close in its latent space. Despite its advantages, a continuous latent representation is not ideal for discrete data, as it can introduce semantic mismatches, reduce interpretability, complicate reconstruction, suffer from posterior collapse, and require complex relaxations for optimization. In the following section we present a model that is designed to work with discrete data inputs.

2.3.2 VQ-VAE

The Vector Quantized VAE (VQ-VAE) was proposed by Oord *et al.* [72] under the premise that some types of data, *e.g.*, audio and images, are naturally represented by discrete values, thus it makes more sense to use a discrete latent representation rather an continuous one. The model follows the VAE framework but uses Vector Quantization (VQ) to transform the continuous latent space regressed by the decoder into a discrete one.

A forward computation of the model is devised as follows: the encoder network E encodes the input data a latent vector z; this vector is quantized into the vector q_c; which is fed to the decoder network D that outputs the reconstruction $\tilde{X}$. These operations are illustrated in Figure 2.6. The process of Vector Quantization is composed of a set Q of L-dimensional learnable discrete vectors, a codebook C and a Quantizer function $\mathcal{Q}(z, Q)$. Each vector $q \in Q$ has a code number c in the codebook C. The quantizer function $\mathcal{Q}$ outputs q_c, an embedding

Figure 2.6: **VQVAE architecture.** The model architecture follows a encoder-decoder schema. The encoder creates latent vectors z from the input x that are quantized by the process of Vector Quantization. These quantized vectors q_c are decoded into the reconstruction of the input $\tilde{x}$.



Source: Author.

vector with code number c. The embedding vector q_c minimizes the reconstruction error of the vector z in the set Q. Specifically,

$$q_c^* = \arg\min_{q_c \in Q} ||z - q_c||_2^2. \tag{2.4}$$

Since the latent space is quantized, we redefine the latent distribution $p_\theta(z|x)$ in Equation 2.5 to reflect the change:

$$p_\theta(z = c|x) = \begin{cases} 1, & \text{if } c = c^* \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

In the VQ-VAE, the desired latent distribution $p(z)$ is Uniform as opposed to the Gaussian distribution of VAE. Following this modification, we derive the regularization term in Equation 2.6 for $p_\theta(z|x)$ and $p(z) \sim \mathcal{U}(0, 1)$:

$$D_{KL}\left(p_\theta(z|x)||p(z)\right) = \sum_{z \in Z} p_\theta(z|x) \log \frac{p_\theta(z|x)}{p(z)} = 1. \log \frac{1}{1/|C|} = \log |C|. \tag{2.6}$$

Where "$|\cdot|$" is the *modulo* mathematical operator. Finally, we present the loss function for the VQ-VAE model in Equation 2.7:

$$\mathcal{L}_{VQ} = \text{MSE}(x, \tilde{x}) + \|\text{sg}[z] - q_c\|_2^2 + \beta\|\text{sg}[q_c] - z\|_2^2 + \log |C|, \tag{2.7}$$

where *sg* stands for stopgradient, an operator that stops the gradient computation in the computational graph of differentiable libraries such as PyTorch [53]. Please note that the second term ($\|\text{sg}[z] - q_c\|_2^2$) updates the codebook vectors q_c by pulling them toward the encoder output z, while the third term ($\beta\|\text{sg}(q_c) - z\|_2^2$) ensures the encoder output z commits to the quantized

representation. The stop-gradient (sg) prevents gradients from flowing through the respective variables, ensuring that these terms influence different parts of the model without interfering with each other. The first term of the equation is the reconstruction term derived before in Equation 2.2, the second term updates the discrete latent vectors q_c to be closer to z while the last term forces the latent z vectors to be closer to q_c and β weights this relationship.

2.4 Summary & Closing Remarks

Section 2.1 presented the concept of Neural Machine Translation and its main framework: the Encoder-Decoder. We follow discussing a model that achieved state-of-the-art in numerous areas of Deep Learning and intuitions how its design choices contributed to its performance.

In Section 2.2, we introduced the task of Human Motion Synthesis and discuss how its complex. We went through how the area can benefit from using deep learning models and how the data necessary to train these models is collected and organized. A overview of the recent deep learning techniques is given in sub-section 2.2.2.

At last, in Section 2.3 focus on models that learns latent representations. We detail the VAE architecture and explain why use such framework benefit the construction of a latent space. In the sequence, we present the VQ-VAE model, which is a method more suitable to work with discrete input data, *e.g.*, sign language.

Chapter 3

Related Work

In this chapter, we present a comprehensive review of the existing research in the field. First we focus on the Sign Language Production works, contextualizing the current state of the literature. Then, we show how recent works are using a discrete intermediate representation to produce high quality results in different fields.

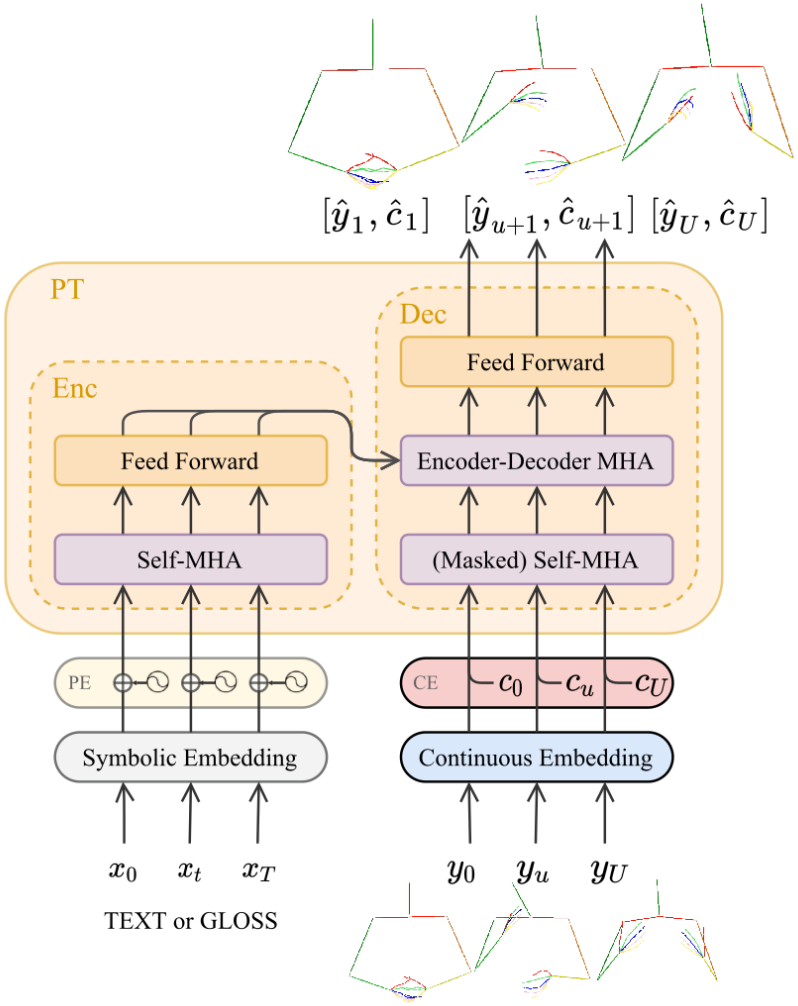
3.1 Sign Language Production

Early sign language production methods used animated avatars [25, 34, 47, 19, 20] to generate the movements. Although the results were realistic, these models rely on pre-defined correspondences of written text and sign movements, *e.g.*, dictionaries, which demands extensive human annotation and restricts the model's scalability. Other techniques such as [35, 37] combine the ideas of rule-based models and statistical machine translation to diminish the annotation burden. However, the complexity of these methods rapidly increases as the translated phrases become more complex.

Stoll *et al*. [68] propose to learn the mapping between text and glosses and create a dictionary of mean poses for each gloss. A pose is generated from a dictionary for every word input and then fed to an image generator. The work of Zelinka and Kanis [86] builds an end-to-end model that produces a sign of seven frames for each word. Both works generate blocks of signs with length linearly dependent on the input size, neglecting that the alignment between sign and spoken language sequences is usually unknown and nonmonotonic [10]. Stoll *et al*. [69] were one of the first authors to use deep learning techniques to model the problem and produce sign videos as outputs. Each written text sentence is first translated into glosses and then a Markov Process produces a pose output for each gloss. These pose outputs are fed to a Generative Adversarial Network (GAN) [26] to synthesize an image. Despite generating good frames from glosses, the model produced a discontinuous output as a concatenation of signs.

Saunders *et al*. [64] were the first to synthesize continuous signs outputs and is currently the reference baseline in the literature. The authors modified the original Transformer archi-

Figure 3.1: **ProgressiveTransformers architecture.** The model is adapted from the original Transformers model to generate continuous data. The authors remove the softmax layer and substitute discrete embeddings layers with linear projections. They also introduces an auxiliary variable named *Counter Decoder* C_u.



Source: Saunders *et al.* [64]

tecture [73] (See Figure 3.1) to produce continuous outputs instead of discrete ones, *e.g.*, text, by exchanging the Embedding layer dictionary for a Continuous Embedding layer, *i.e.* a linear layer. To help the process of pose decoding, the Positional Embedding is substituted by a Counter Decoder, which includes a value representing the percentage of the output completion at the autoregressive process. The final softmax layer was removed and the model directly regresses poses. In their following work, Saunders *et al.* [63] added facial landmarks and trained the model in an adversarial regime, which mitigated the problem of under-expressed sign production (regression to the mean).

Most recently, several methods are adopting an intermediate representation [62, 66, 30] to synthesize more expressive signs. Saunders *et al.* [66] presented a Mixture-of-Expert architecture that uses gloss annotation to learn low-level sign representations. Huang *et al.* [30]

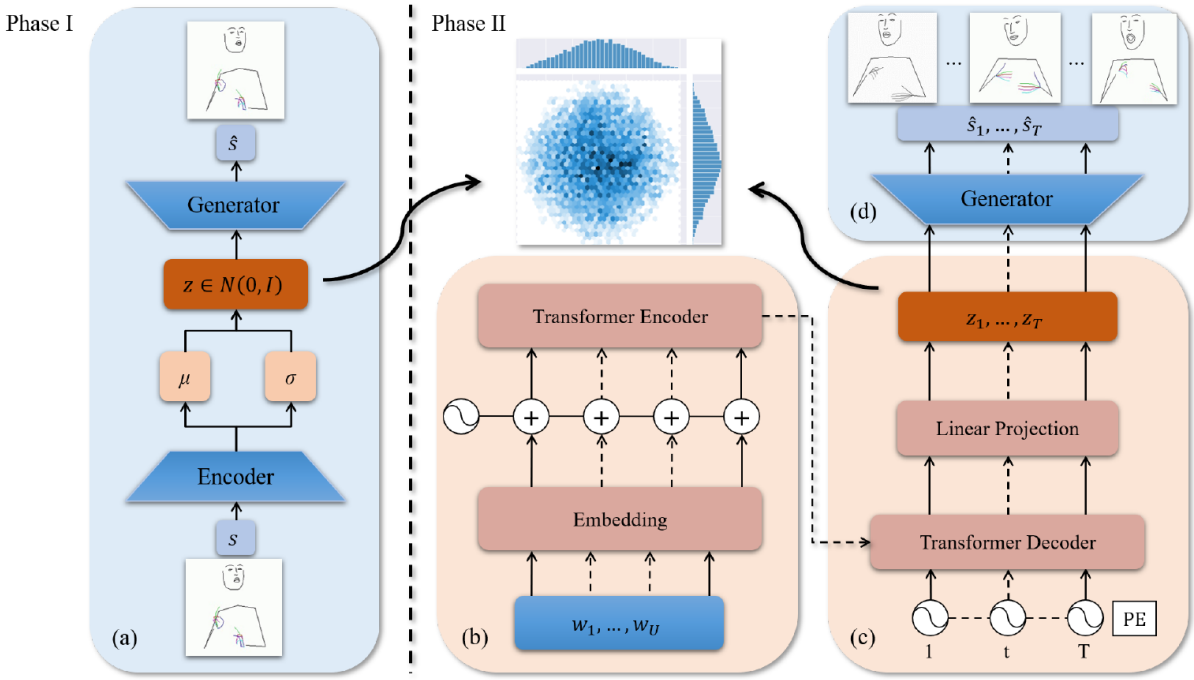
proposed a non-autoregressive model to create latent features from glosses and then use an external module to predict the duration of the output. Saunders *et al*. [67] tackled the SLP problem in a five-step process: text-to-gloss translation; creation of a dictionary of glosses-to-poses; a module that selects which dictionary values will be maintained; conversion of these dictionaries into poses and finally the translation from pose to image. Some of the latest approaches [71, 31] are trying perform translations in both ways, written text to signs and signs to written text, and leverage the correspondence between the two types of data in the training phase to synthesize better outputs. Despite the introduction of a new paradigm, these models still performs worst than the ones that use intermediate representations and also do not include non-manual components of signs (only synthesize the torso of the interpreter), a crucial characteristics for Sign Language Production.

However, all of these proposed methods rely on gloss annotation, which is challenging to acquire on a large scale [16] since it requires a person who knows sign language and the spoken language. Moreover, gloss annotation lacks standardization. Saunders *et al*. [62] presented one of the first sign language production models based on a learnable intermediate representation without gloss supervision. The authors applied the Progressive Transformer to regress the parameters of a Mixture Density Network and generates signs by sampling from a learned distribution.

The work of Hwang *et al*. [32] is the work more close to ours and currently holds the state-of-the-art results for text-to-sign translation model with code publicly available (See Figure 3.2). The authors propose a two-phase learning scheme: first they learn how to synthesize high-quality poses by training a Variational AutoEncoder model (Figure 3.2 -a); then a Transformer model is used in a non-autoregressive manner to regress the latent values to be decoded into poses by the VAE Decoder. By learning a intermediate representation, they can train more expert models in each task, *i.e.* translate text into latent sign representations (Transformers) and synthesize poses from a latent representation (VAE). Moreover, this latent representation is learned by a self-supervised method which does not require more data labeling. Another contribution of this work is the non-autoregressive decoding. Instead of generating one frame at a time, a sequence of integers corresponding to the indexes of the output frames is passed as input do the Transformers Decoder. This approach mitigates the problem of Error Propagation [4] in autoregressive models.

While our model also incorporates a learnable intermediate representation, it differs from the last two works in that it adopts a discrete representation. This approach helps circumvent the issue of posterior collapse, which occurs when the intermediate representation is ignored when pairing with a powerful decoder [72].

Figure 3.2: **NSLP-G architecture.** Figure a) illustrates model Phase I, where the VAE model learns continuous sign representations. These representations are used in Phase II (Figure b)) as the output of the Transformers model. The VAE decoder is used to generate signs from the regressed latent space.

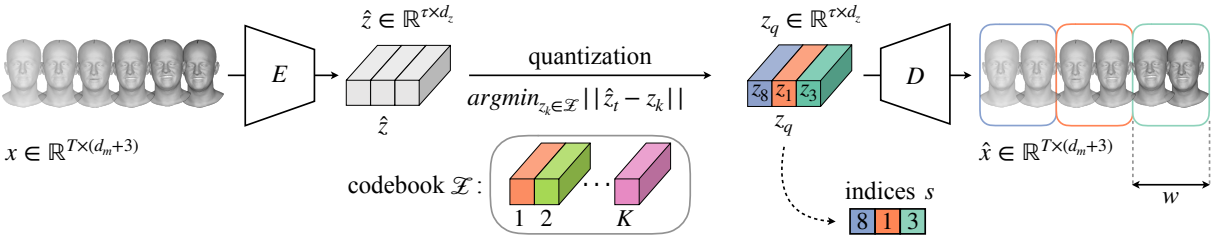


Source: Hwang *et al.* [32]

3.2 Intermediate Representation

A common approach to learning an intermediate representation, *i.e.*, latent space, is to use self-supervised learning techniques. Vector Quantized VAE (VQVAE) [72] was one of the first successful solutions for quantizing latent vectors to create a discrete space. The model works in an encoder-decoder manner, where the encoder projects the input data into a higher dimension which can be decoder in the input data again. Following this approach, several works were devised in recent years demonstrating the power of discrete latent representations. The works of Azavi *et al.* [59] and Fauw *et al.* [22] use a hierarchical structure to construct its latent space incrementally in a top-down approach. Each layer output is fed into the next one, serving as conditioning. Ramesh *et al.* [56] concatenated preprocessed text tokens with discrete image latent tokens to synthesize images. Esser *et al.* [21] added into the VQVAE a patch-based discriminator in an adversarial regime and a perceptual loss based on the reconstruction of the latent space to high-resolution conditional image synthesis. Yu *et al.* [82] improve the results by normalizing the latent codes and projecting them into a lower dimension before quantization. In the following research, Yu *et al.* [83] extended the model to receive text prompts and generate realistic images. Zeghidour *et al.* [85] use Residual Quantization (RQ) to approximate the latent space in a coarse-to-fine manner and compress audio signals with a high Signal-to-Noise ratio.

Figure 3.3: **Motion VQ-VAE architecture.** At the first stage the codebook is learned using a Residual Quantizer. The latent vectors are reshaped and fed to a Transformers architecture that first regresses the spatial features and then regresses the depth residuals.



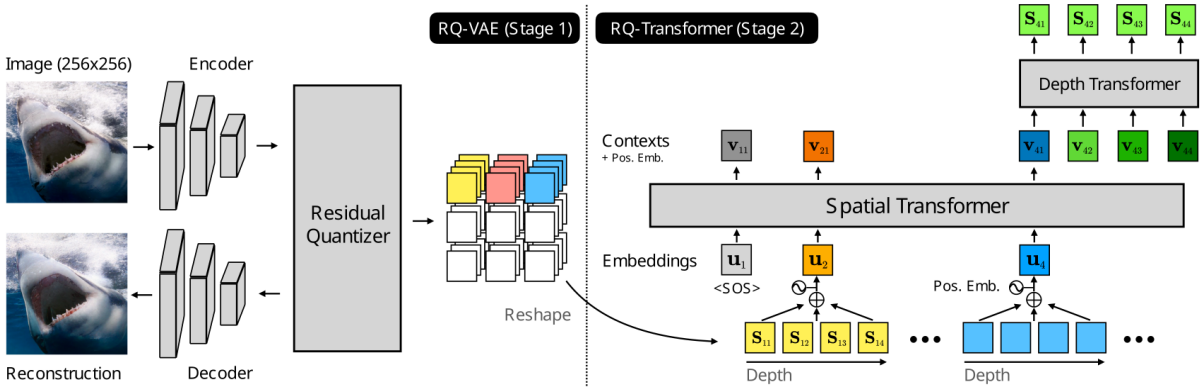
Source: Ng *et al.* [50]

Ng *et al.* [50] were the first to use a VQ-VAE in the context of motion synthesis (see Figure 3.3). The authors encoded a temporal sequence of 3D facial parameters from a speaker into a latent space and quantized it. Then, they used these vectors to sequentially decode batches of the listener facial parameters responses.

A work more close to ours was proposed by Lee *et al.* [39], the RQ-Transformer (presentend in Figure 3.4). The model uses a Residual Quantization module that incrementally quantizes a latent image representation. This quantized representation is then used by the decoder to reconstruct the image. After training the RQ-VAE module, the authors use it alongside with a Transformers to learn how to synthesize these latent tokens from text inputs.

Our work differs from the previous methods by two major contributions: our model uses codes and quantized vector information on the training step, and the sign poses are represented in a discrete dictionary that is learned self-supervised without requiring glosses. As shown in our experiments, this approach brings a substantial improvement and, to the extent of our knowledge, has not been used before in Sign Language Production.

Figure 3.4: **RQVAE and RQTransformers architecture.** At the first stage the codebook is learned using a Residual Quantizer. The latent vectors are reshaped and fed to a Transformers architecture that first regresses the spatial features and then regresses the depth residuals.



Source: Lee *et al.* [39]

3.3 Summary & Closing Remarks

In this chapter, we reviewed the literature of Sign Language Production and generative models that uses Intermediate Representations. We start Section 3.1 by presenting first rule-based and statistical SLP models. These models might not produce realistic results either because of lack of data that requires human annotation or due to the models inability to handle complex phrases. With the availability of more data, deep learning techniques started to arise, and the Transformer architecture became the most used model in the area. Several models were devised upon the Transformers using a intermediate representation to produce more realistic signs. Despite the improvements shown from these models, the majority of them uses glosses annotations as a form of supervision in the training phase, making unfeasible train them in most languages due to lack of data. The section also explores the NSLP-G model, which uses a VAE model as a intermediate representation to be learned by a Transformers. This model is the most similar to our work.

The literature of models that uses Intermediate Representations is reviewed in Section 3.2. We focus on models with discrete latent representations. This section starts with the VQ-VAE model, first model to propose a discrete representation using de VAE framework. Then we present models that propose a hierarchical structure as well models with multimodal capability such as text-to-image generation. At last, we outline the first model to use a powerful Residual Vector Quantization procedure with a Transformers to synthesize realistic images from text.

Chapter 4

Methodology

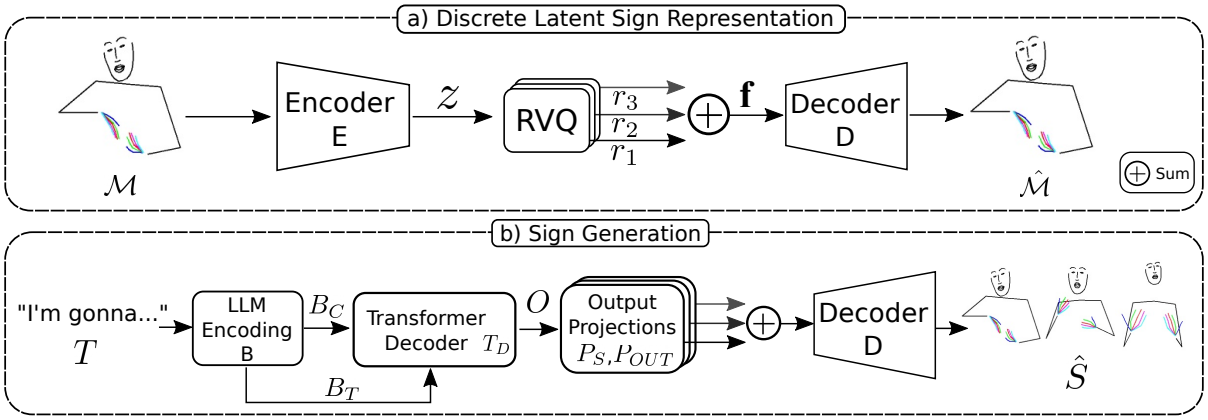
Our methodology is designed to produce a sequence of signs S compounded from manual and non-manual components of sign language, *i.e.*, body pose, hand shape and motion, and facial expressions. Given an input text T, we synthesize a sequence S of N signs defined as follows:

$$S = \{M_1, \cdots, M_N\} \in \mathbb{R}^{N \times 120 \times 2}, \tag{4.1}$$

where M_i is the skeleton of the interpreter in the i-th frame, *i.e.*, $M_i \in \mathbb{R}^2$ contains the skeleton landmarks in the 2D image coordinates.

Our model comprises two main steps, outlined in Figure 4.1. First, it learns a Discrete Latent Sign Representation in a residual manner. This representation maps a discrete sequence of tokens into a continuous sequence of signs without supervision (Section 4.1). Second, we use a transformer decoder T_D to learn sign representations from text (Section 4.2). This two-stage training schema enables us to dismember the learning process into specific tasks.

Figure 4.1: **Overview of our sign language production method.** Our methodology is composed of two main steps: Our model consists of two key steps: **a)** First, it employs a residual learning approach to learn a Discrete Latent Sign Representation. This representation maps a discrete sequence of tokens to a continuous sequence of signs, all without supervision; **b)** Second, we use a transformer decoder, denoted as T_D, to learn sign representations from text and which are decoded into a sign sequence by components of the first step.



Source: Author.

4.1 Discrete Latent Sign Representation

We present a discrete latent sign representation that aims to model the correspondences between a discrete learnable space into a continuous one (signs sequence S). To create a representation with natural fit for the sign production context, we exploit the capability of Residual Vector Quantized VAEs (RVQVAE) to produce a rich discrete representation.

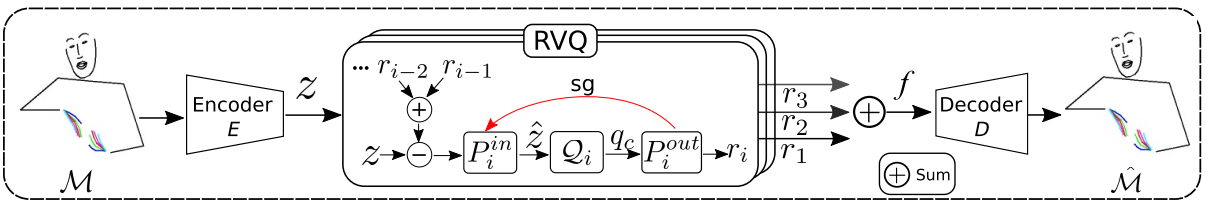
Our architecture is based on an Encoder-Decoder network with a Residual Vector Quantization module, Figure 4.2 shows a detailed schematic representation of the process. The Encoder E produces a latent representation $z \in \mathbb{R}^L$ of the input sign, where L is the dimensionality of the latent vector space. This representation feeds the Residual Vector Quantization (RVQ) module, which incrementally generates a quantized representation $f \in \mathbb{R}^L$. Finally, the decoder D outputs the skeleton $\hat{M}$.

4.1.1 Residual Vector Quantization

Our architecture incrementally learns a discrete representation using three residual layers. Each residual layer is composed of a quantizer $\mathcal{Q}$ function and two linear projections.

Let Q_i be the set of L-dimensional learnable vectors q and C_i the corresponding codebook of the i-th residual layer. Each vector $q \in Q_i$ has a code number c in the codebook C_i. The quantizer function $\mathcal{Q}(\hat{z}, Q_i)$ outputs q_c, an embedding vector with code number c. The embedding vector q_c minimizes the reconstruction error of the vector $\hat{z}$ in the set Q_i. Specifically,

Figure 4.2: **Residual Vector Quantization Module.** The input sign pose is encoded into the latent representation z by the Encoder E and progressively refined by the RVQ module. At each layer, the residual r_{i-1} from the previous quantization step undergoes another quantization (q_c) and is projected into a new residual r_i by P_i^{out}, recursively structuring the latent space. Before quantization in $\mathcal{Q}_i$, each signal is first transformed by P_i^{in}. The input to this projection is computed as the difference between z and the cumulative sum of the previous residuals. The final residuals are aggregated into the output f, which is then decoded to reconstruct the original sign pose.



Source: Author

$$q_c^* = \arg\min_{q_c \in Q_i} ||\hat{z} - q_c||_2^2. \tag{4.2}$$

To produce the quantized representation of the input vector z, first, we feed z into the first residual layer

$$r_1 = P_1^{out}(\mathcal{Q}(P_1^{in}(z), Q_1)), \tag{4.3}$$

where P_{in} and P_{out} are linear projections. Then, we iteratively feed the output of a residual layer to the next one (Figure 4.1-a). Formally,

$$r_i = P_i^{out}\left(\mathcal{Q}\left(P_i^{in}\left(z - \sum_{j=1}^{i-1} r_j\right), Q_i\right)\right). \tag{4.4}$$

At last, we sum the output of all residual layers to compose the final representation, *i.e.*,

$$f = \sum_{i=1}^{3} r_i. \tag{4.5}$$

We formalize this iterative process in the Algorithm 1. Please note that we copy the gradient of $\hat{z}$ into q_c in line 6 of Algorithm 1. This operation is done by using the stop-gradient (sg) operator to add the difference between q_c and $\hat{z}$ into $\hat{z}$. The arithmetic result is q_c, which does not modify the value of the variable q_c but copy the gradients from $\hat{z}$ to q_c in the computational graph. By copying the gradient in the computational graph, we are able to backpropagate the errors through the discontinuous quantizer function $\mathcal{Q}$ and train the model.

Our Encoder and Decoder architecture are simple and lightweight, reducing the computational costs and time of training. Each one is composed of a sequence of blocks of Linear layers with Tanh activations. We detail the full architecture of our Discrete Latent Sign Representation module in Table 4.1.

Algorithm 1: Residual Vector Quantization

Input: $z = E(x)$ the output of the encoder
Output: the quantized representation f

1 $f \leftarrow \vec{0}$
2 $residual \leftarrow z$
3 **for** $i = 1$ *to* 3 **do**
4 $\quad \hat{z} = P_i^{IN}(residual)$
5 $\quad q_c = \mathcal{Q}_i(\hat{z})$
6 $\quad q_c = \hat{z} + sg[q_c - \hat{z}]$ `// gradient copy`
7 $\quad r_i = P_i^{OUT}(q_c)$
8 $\quad residual \mathrel{-}= r_i$
9 $\quad f \mathrel{+}= r_i$
10 **return** f

Table 4.1: **Discrete Latent Sign Representation architecture.** B is the batch size input dimension. The $\circ$ operator defines sequence of layers in our architecture.

Block	Operations	Module	Input Size	Output Size
1	$Linear \circ Tanh$	Encoder	(B, 240)	(B, 512)
2	$Linear \circ Tanh$	Encoder	(B, 512)	(B, 512)
3	$Linear \circ Tanh$	Encoder	(B, 512)	(B, 768)
4	$Linear \circ Tanh$	Encoder	(B, 768)	(B, 768)
5	$Linear \circ Tanh$	Encoder	(B, 768)	(B, 1024)
6	$Linear$	P_1^{IN}	(B, 1024)	(B, 64)
7	$Quantizer$	$\mathcal{Q}_1$	(B, 64)	(B, 64)
8	$Linear$	P_1^{OUT}	(B, 64)	(B, 1024)
9	$Linear$	P_2^{IN}	(B, 1024)	(B, 64)
10	$Quantizer$	$\mathcal{Q}_2$	(B, 64)	(B, 64)
11	$Linear$	P_2^{OUT}	(B, 64)	(B, 1024)
12	$Linear$	P_3^{IN}	(B, 1024)	(B, 64)
13	$Quantizer$	$\mathcal{Q}_3$	(B, 64)	(B, 64)
14	$Linear$	P_3^{OUT}	(B, 64)	(B, 1024)
15	$Linear \circ Tanh$	Decoder	(B, 1024)	(B, 768)
16	$Linear \circ Tanh$	Decoder	(B, 768)	(B, 768)
17	$Linear \circ Tanh$	Decoder	(B, 768)	(B, 512)
18	$Linear \circ Tanh$	Decoder	(B, 512)	(B, 512)
19	$Linear \circ Tanh$	Decoder	(B, 512)	(B, 240)

4.1.2 Optimization

To train our model, we minimize the reconstruction error and approximate each latent vector to its quantized version. The reconstruction loss $\mathcal{L}_{rec}$ is given by the sum of the Mean Squared Error (MSE) individually applied into four body parts: the head, the torso, and the left and right hands. This approach avoids parts with small magnitude errors being over-smoothed by the denominator of the equation of MSE. The latent vector approximation loss $\mathcal{L}_{commit}$ is defined as:

$$\mathcal{L}_{commit} = \sum_{i=1}^{3} ||z_{r_i} - sg\left[\mathcal{Q}(z_{r_i}, Q_i)\right]||_2^2, \tag{4.6}$$

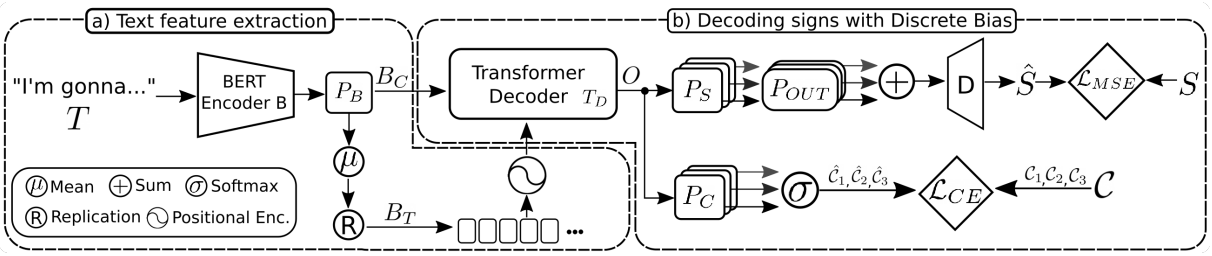
where z_{r_i} is the input to i-th residual layer and $sg[\cdot]$ is the stop-gradient operator.

The final loss is given by

$$\mathcal{L}_M = \mathcal{L}_{rec} + \beta\mathcal{L}_{commit}, \tag{4.7}$$

where β weights the commitment loss. Considering the quantization operation is not differentiable, we copy the gradients after the quantization step to the step before. To update our dictionary, we utilize the exponential moving average method described in Oord *et al.* [72].

Figure 4.3: **Sign Generation.** Our sign generation procedure consists of two main steps: **a)** we extract text features from a Bert Encoder B and project them into a new space P_B; **b)** a Transformer Decoder T_D decodes the text features into the output O which will be forwarded through the upper branch to synthesize signs by using components of the RVQVAE module. We induce a discrete bias in the training phase in the lower branch of the model by learning how to synthesize the correspondent codes of signs (quantized vectors that have correspondent codes in the VQVAE codebook).



Source: Author.

4.2 Sign Generation

To synthesize signs $\hat{S}$ based on an input text T, we devise an architecture (illustrated in Figure 4.3) that leverages the representative features produced by Large Language Models (LLMs) and correlates them with the discrete latent space of signs learned. This sign synthesis comprises two steps: a) we extract a sequence of text features from an LLM model B and project them into a new learnable space P_B; b) the sequence of features are used as inputs to a Transformer Decoder [73] T_D to produce discrete latent sign vectors that will be decoded into the final output signs $\hat{S}$ by the decoder D.

Table 4.2: **Architecture of our language projector** P_B. We specify its layers with each correspondent input and output sizes.

Block	Operations	Input Size	Output Size
1	$Linear$	(B, 768)	(B, 768)
2	$GELU$	(B, 768)	(B, 768)
3	$Dropout(p = 0.1)$	(B, 768)	(B, 768)
4	$Linear$	(B, 768)	(B, 768)

4.2.1 Text feature extraction

We create text representations by leveraging pretrained Large Language Models (LLMs). Given an input text T, we encode this sentence using a BERT Encoder [17] B. Since these encoded features were optimized for a different task, we project them into a new space with a projection P_B (detailed in Table 4.2). The projected features are used as inputs for the Transformer Decoder T_D. We use these features as key and value inputs (B_C) to condition T_D and query the outputs by taking the mean vector of the projections and repeating them N times (B_T) (the desired output length). We used the mean representation of the key and values as queries to keep the same projected text representation space as the key and value inputs. This mean vector serves as a comprehensive representation of the phrase's overall meaning, capturing the central semantic information embedded in the key and value features. Finally, Positional Encoding is applied to the query input for temporal context. Figure 4.4 illustrates this procedure.

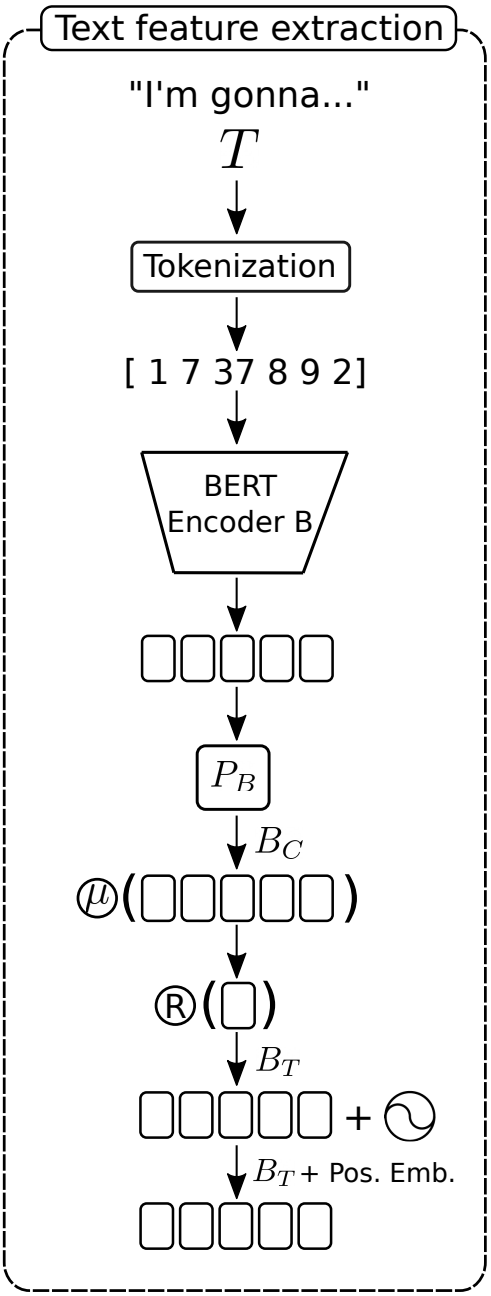
4.2.2 Decoding signs with discrete bias

We use a Transformer Decoder architecture T_D to transform text representations into our Discrete Latent Representation. The decoding process is based on a non-autoregressive strategy that uses the projected features as inputs, a detailed schema of the forward process is presented in Figure 4.5. Thus, given a sequence of inputs encoded by the LLM, the decoder T_D outputs a sequence of feature vectors O that correlates the input text T and an intermediate sign representation.

Since the Discrete Latent space has two types of representations, *i.e.*, quantized vectors and their corresponding codes, we use these two pieces of information to condition the model and produce better signs. We project O into the discrete latent space of the decoder D using a projection P_S to explore the quantized vector information. This projection comprises three parallel residual layers. Then, these features are decoded into a sequence of signs $\hat{S}$ using the projection P_{out} followed by the sum of all residual layers ($f = \sum_{i=1}^{3} r_i.$) and the decoder D. Finally, we use the Mean Squared Error function $\mathcal{L}_{MSE}$ to computer the error between the generated signs and the ground truth signs sequence. This process is illustrated in the lower branch of Figure 4.3-b).

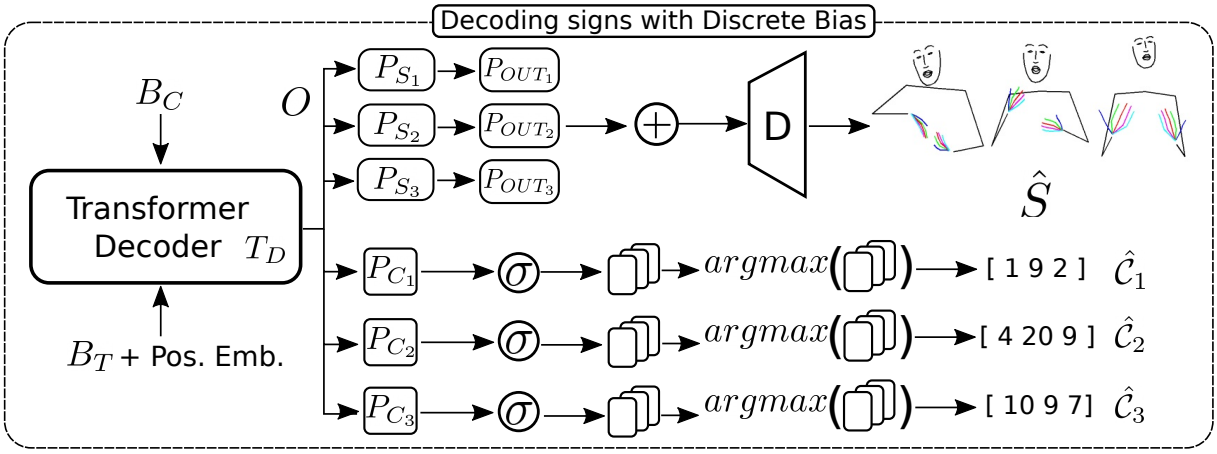
To explore the code information (discrete bias), we add another projection module P_C that transforms each vector output O into a vector of length $|C|$. The softmax function is applied to produce a probability distribution of the codes. Therefore, given a sequence of latent vectors O, we transform them into three sequences of probability distributions $\hat{\mathcal{C}}$ which are optimized

Figure 4.4: **Text feature extraction.** Given an input text, we transform each word into its correspondent token by tokenization. Then, we feed this tokens into the pretrained BERT Encoder B, extracting its features. These features are fed to our language projector P_B which outputs the features B_C that will serve as key and value inputs of the Transformers decoder T_D. We take the mean vector of the sequence B_C and replicate them into the sequence of length L, B_T. This sequence passes through the Positional Embedding and acts as the Query input in the Transformers decoder T_D.



Source: Author.

Figure 4.5: **Decoding signs with discrete bias.** This phase receives as inputs the sequences B_C and B_T + Positional Embedding coming from the text feature extraction module. The Transformer Decoder T_D decodes these text features into the output O which is fed into the projectors P_C and P_S sub-modules. On the upper branch, the RVQVAE components P_{OUT} and D are used to decode the output sign sequence $\hat{S}$. On the lower branch the outputs from P_C are passed through a *softmax* function and a *argmax* function to raise the sequences of codes $\hat{\mathcal{C}}_1$, $\hat{\mathcal{C}}_2$ and $\hat{\mathcal{C}}_3$.



Source: Author.

by using the Cross-Entropy loss $\mathcal{L}_{CE}$ as outlined in the upper branch of Figure 4.3-b). The final loss function for the Transformer Decoder model is as follows

$$\mathcal{L}_{SC} = \mathcal{L}_{MSE}(S, \hat{S}) + \sum_{i=1}^{K} \mathcal{L}_{CE}(\mathcal{C}_i, \hat{\mathcal{C}}_i), \tag{4.8}$$

where C_i represents the i-th sequence of discrete codes, each corresponding to an entry selected from the i-th codebook in the residual quantization process. These codes are not directly used during inference but serve to introduce a discrete bias in the learning process. This bias is enforced by computing the Binary Cross-Entropy (BCE) loss between the predicted probabilities $\hat{C}_i$ and the corresponding ground truth codes. By incorporating this constraint, the model learns to align its outputs with discrete representations during training while still generating continuous outputs during inference. We detail our projection functions architectures P_S and P_C in Table 4.3.

By jointly optimizing codes and signs, we add a bias to our model, guiding the projection P_s to produce better results regarding sign semantics and more faithful movements. This behavior is verified in our ablation study.

In line with the practices in the field and for the sake of a fair comparison, we follow the same strategy used in [32, 64, 65, 66, 67], where the sequence is decoded in fixed size and subsequently crop it.

Table 4.3: **Architecture of projection functions P_S and P_C.** For each function we present its sequences of block operations and its correspondent input and output sizes. The operator $\circ$ indicates a composition of two operations. Note that we use the same architecture for each sub-projector, hence the sub-index i. Also note that the projectors from P_C and P_S are similar, only differing in the last layer when the output size differs.

P_{S_i}

Block	Operations	Input Size	Output Size
1	$Linear \circ ReLU$	(B, 768)	(B, 384)
2	$Linear \circ ReLU$	(B, 384)	(B, 192)
3	$Linear \circ ReLU$	(B, 192)	(B, 96)
4	$Linear \circ ReLU$	(B, 96)	(B, 64)

P_{C_i}

Block	Operations	Input Size	Output Size
1	$Linear \circ ReLU$	(B, 768)	(B, 384)
2	$Linear \circ ReLU$	(B, 384)	(B, 192)
3	$Linear \circ ReLU$	(B, 192)	(B, 96)
4	$Linear \circ ReLU$	(B, 96)	(B, 129)

4.3 Summary & Closing Remarks

In this Chapter we presented our methodology in detail. We first started with a formal presentation of our problem. Then we followed to a outline of our proposed architecture, presenting its two main steps: Discrete Latent Sign Representation and Sign Generation.

Section 4.1 introduces our Discrete Sign Representation module. It uses the Residual Vector Quantization model to incrementally learn a discrete representation of human skeletons in the 2D plane. We obtain two types of data from this representation: discrete codes and its correspondent discrete latent vector.

Our Sign Generation step is presented in Section 4.2. In this phase, we leverage the pre-trained weights of a BERT Encoder [17] to extract powerful representations of the written text. We adapt the Transformer Decoder architecture to use these representations and decode them into the space of our Discrete Latent Representation. Our previously trained human skeleton decoder D uses the Transformer Decoder output to produce a sequence of sign $\hat{S}$. This process is optimized by the $\mathcal{L}_{MSE}$ and the $\mathcal{L}_{CE}$ losses, which reconstructs the signs in the output and adds a discrete bias in the learning process, respectively.

Chapter 5

Experiments and Results

This chapter presents the experiments conducted to evaluate our method, and the corresponding results. The main objective of this chapter is to offer a thorough analysis of our approach and its effectiveness in generating sign language sequences. In the following sections we provide details about the datasets used and how we pre-processed them, the implementation and training details of our method and the evaluation metrics used in the experiments. Then we conduce ablation studies to verify the importance of our components and compare our model to the state-of-the art in both quantitative and qualitative ways.

5.1 Dataset and preprocessing

We evaluate our method on two datasets from two different languages, Phoenix14T [9] (Figure 5.1), and How2sign [18] (Figure 5.2). Phoenix14T is a common benchmark for Sign Language Production; it contains 8,257 weather forecasting samples of German Sign Language (GSL) with 2,887 unique words. How2Sign is a multi-modal and multiview American Sign Language (ASL) dataset with 35,129 sentence-level annotations. Since the How2Sign dataset has enough samples per speaker and the signs have nuances per speaker, we evaluate our method per speaker. More specifically, we select the two signers with the largest amount of data in the dataset, namely signers 5 and 8. Both datasets are annotated on the sentence level (one written sentence for each sign video), but only Phoenix14T made publicly available the glosses annotations to this date. To extract skeletal information from the videos, we utilize OpenPose [12] to detect landmarks, which are then preprocessed[1] using the same method as Zelinka *et al*. [86].

[1] Head data is filtered and interpolated with a Smoothing Spline algorithm [15].

Figure 5.1: **Phoenix14T clip examples.** Please note the variety and complexity of signs.

Source: Adapted from Camgoz *et al*. [9]

5.2 Implementation and training

Our RVQ-VAE model uses a codebook of size $|C| = 128$, latent dimensionality $L = 64$, and three residual layers. We update our codebooks with the EMA decay $\alpha = 0.99$, and we empirically set the parameter $\beta = 0.1$ in the loss function. Our model is trained for 100 epochs with a batch size of 64 and learning rate $\eta = 1 \times 10^{-4}$. In the process of training, we reduce the learning rate when a plateau is found by a factor of 0.8 with patience factor of 5 until a minimum learning rate of 1×10^{-6}.

We train the Sign Generation module for 600 epochs with a batch size of 64, a learning rate of 1×10^{-4}, we reduce the learning rate by a factor of 0.5 with patience factor of 10 and do not use dropout. Our Transformers Decoder T_D module is formed by 6 hidden layers with 12 attention heads with an intermediate size of 1024. To optimize our model, we use the AdamW [46] algorithm with AmsGrad [60] parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay of 1×10^{-2}. We trained both of our models with an Intel Xeon CPU E5-2620, an NVIDIA

Figure 5.2: **How2Sign clip examples.** Different scenes from the datase How2Sign including all the signers.

Source: Adapted from Duarte *et al*. [18]

GeForce GTX Titan X 12GB, and 128GB RAM. The inference time of a unitary batch is 0.02 seconds. Each model training time was 4.5 hours and 28 hours for the our Discrete Latent Sign Representation and Sign Generation models, respectively.

For the How2Sign dataset, we train two separate models, one for each signer, and evaluate each model on the corresponding signer's test data. In contrast, for the Phoenix14T dataset, we train on data from all signers and evaluate on a test set that includes all signers.

5.3 Evaluation metrics

We evaluate the quality of a synthesized sign from two perspectives: language and movement. From the language perspective, *i.e.* the translation from written to sign language, we employ the BLEU [52] (1 to 4 n-grams) and ROUGE [42] metrics, which are widely used in the literature on Sign Language Production and Neural Machine Translation. To use these metrics, we train a state-of-the-art back-translation model [11] that translates the synthesized expressions to oral language sentences. Since the back-translation model requires gloss annotations, we only evaluate our results with this metric on the Phoenix14T dataset.

The BLEU metric measures the similarity between two sentences by generating n-grams of them and comparing it. To every pair of sentences, we compute the harmonic mean for every n-gram clipped precision and weight it by a brevity penalty. This procedure is defined by Equation 5.1:

$$BLEU(K) = BP \times \prod_{k=1}^{K} cp_k^{(\frac{1}{k})}, \tag{5.1}$$

where BP is the brevity penalty, cp is the clipped precision and K is the n-gram number. The clipped precision count the occurencies of the current generated n-gram into the reference sentence but limits it by the number of its ocurrrencies in the reference sentence. The brevity penalty (BP) penalizes a sentence that is shorter than the reference sentence. We define it in Equation 5.2:

$$BP = \begin{cases} 1, & \text{if } |\hat{T}| > |T| \\ e^{(1-|T|/|\hat{T}|)}, & \text{otherwise,} \end{cases} \tag{5.2}$$

where $|T|$ and $|\hat{T}|$ are the lengths of the ground truth input text and the back-translated text, respectively.

The ROUGE score is based on the Longest Common Subsequence (LCS) of two sentences. Given a pair of sentences, we compute the LCS and calculate the precision and recall. For every score computed, we divide it by the generated sentence length. We aggregate all scores and multiply then by 100, as described in Equation 5.3:

$$ROUGE = \left(\sum \frac{Score(T, \hat{T})}{|\hat{T}|} \right) * 100. \tag{5.3}$$

From the movement perspective, we use the original Fréchet Gesture Distance (FGD) [81] proposed by Hwang *et al.* [32] and Mean Average Error of Joint (MAEJ). The FGD measures the the distance between two distributions of latent vectors produced by a powerful model. The ideia is to use this model to create new representations of sign expressions and then compare

how similar the distribution of generated sing and the real signs are. We present the FGD formula in Equation 5.4:

$$FGD(U, \hat{U}) = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}), \tag{5.4}$$

where U is the set that contains all the sign sequences S, μ_r and μ_g are the mean of the real and generated distributions, respectively, and Σ_r and Σ_g are the covariance matrices of the real and generated distributions, respectively.

For the MAEJ metric, it is worth noting that, since a slight misalignment between the synthesized and ground-truth signs can occur, we follow the steps of Huang *et al.* [30] and apply Dynamic Time Warping [5] to align signs temporally. We define the metric in Equation 5.5:

$$MAEJ(M, \hat{M}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{120}|M_{i,j} - \hat{M}_{i,j}|}{120 \times N}, \tag{5.5}$$

where N, M and $\hat{M}$ are the sign length, the ground truth and the predicted skeleton, respectively.

5.4 Ablation study

To verify that the components of our method contribute to the overall sign synthesis performance, we investigated the benefits of the Discrete Latent Space, the Large Language Models (LLMs) representation, and the discrete bias in the sign generation:

- In the **Wo/RVQVAE** experiment, we replace our RVQVAE module with a continuous VAE [36] model using a latent space of the same dimension size;
- In the **Wo/LLMs** experiment, we remove the stage presented in Section 4.2 and replace it with the original Transformers Encoder architecture;
- At last, in **Wo/DB**, we remove the discrete bias induction from the training procedure and compare the synthesized results with our complete architecture.

The Language Metrics results reported in Table 5.1 indicate that the complete configuration performs better. Removing the LLM pre-trained features significantly impacted our model, suggesting that the learned features from large datasets improve text representations for sign synthesis more effectively than classical Transformers Encoder training. The results also demonstrate the importance of the Discrete Latent Space and the Discrete Bias since their removal decreased performance. In terms of Movement metrics, our complete model generally

Table 5.1: **Ablation study.** Models variants evaluated in language and movements metrics. Best in bold.

Language Metrics ↑

	Back Translation (Phoenix)				
Models	BLEU-4	BLEU-3	BLEU-2	BLEU-1	ROUGE
W\o RVQVAE	9.38	12.25	17.41	29.45	30.03
W\o LLMs	6.60	8.72	12.88	24.84	24.50
W\o DB	11.18	14.30	19.91	32.32	32.44
Ours	**11.27**	**14.71**	**20.78**	**33.86**	**33.38**

Movements Metrics ↓

	Phoenix		How2Sign (Singer 8)		How2Sign (Singer 5)	
Models	FGD	MAEJ[1]	FGD	MAEJ[1]	FGD	MAEJ[1]
W\o RVQVAE	3.04	2.60	10.29	2.50	101.88	2.13
W\o LLMs	8.15	2.72	10.64	2.54	126.22	**2.08**
W\o DB	3.44	2.60	8.21	2.50	86.09	2.16
Ours	**2.82**	**2.59**	**7.13**	**2.49**	**79.80**	2.14

[1] *Multiplied by 100*

produced better results, with only a small margin of difference observed in the MAEJ metric on the TEST set of How2Sign (Signer 5). As removing LLM features resulted in poorer metrics in all other cases, the lower MAEJ value could be due to an under-articulated model with movements closer to the mean. Overall, we can see that learning to generate codes corresponding to signs helps induce a better model, which produces outputs with more correspondence with the ground truth.

We also present a sensitivity analysis in Table 5.2 where we vary the number of VQ layers to investigate how the number of layers affects our model performance. We trained our model using two, four and none (one VQ layer) residual layers. The model without residual layers presented the worst performance in all metrics. When varying the number of the residual layers, we see that the model produces results close to the three layer model. Since the language metrics are the most commonly used metrics in the literature, we consider the three layer model the best one since it achieved the best results in all language metrics.

Table 5.2: **VQ layer sensitivity study.** We study how our model behaves when varying the number of VQ layers from one to four. The model with one VQ layer corresponds to a model without residual layers. All of the remaining models has residual layers, being the one with three VQ layers our model.

	Phoenix dataset						
# VQ Layers (# Residual)	BLEU-4 ↑	BLEU-3 ↑	BLEU-2 ↑	BLEU-1 ↑	ROUGE ↑	FGD ↓	MAEJ ↓
1 (0)	9.35	12.45	17.99	30.37	30.73	2.85	2.61
2 (1)	10.51	13.75	19.70	32.29	32.18	2.70	**2.58**
3 (2)	**11.27**	**14.71**	**20.78**	**33.86**	**33.38**	2.82	2.59
4 (3)	10.32	13.62	19.67	32.68	32.71	**2.60**	2.64

5.5 Quantitative Evaluation

We compare our method against the state-of-the-art NSLP-G [32] and Progressive Transformers [64], a commonly used baseline. We selected these models as they have publicly available code and have been used in the Sign Language Production task. Furthermore, to ensure a fair comparison, we trained both models on the same data setup (person-specific and all signers tasks) as our method for the How2Sign [18] and Phoenix14T [9].

Table 5.3 demonstrates that our approach consistently outperforms competing methods across all language metrics, indicating that our model generates more precise and contextually accurate signs. This superior performance in language metrics highlights our model's effectiveness in producing coherent and semantically rich sign language translations.

In terms of movement metrics, our model achieves the best results in all FGD measurements, demonstrating that the synthesized gestures are more closely aligned with the ground-truth distribution compared to other methods. This suggests that our model effectively captures the nuanced dynamics and naturalness of sign language movements.

Although our method did not achieve the lowest MAE results, we argue that this is because our model prioritizes generating realistic, semantically accurate signs rather than merely minimizing numerical error. This approach leads to outputs that are more representative of human sign language, which is crucial for maintaining high-level semantic integrity and communicative effectiveness. Consequently, our model excels in producing both linguistically meaningful and visually realistic sign language outputs.

For completeness, we also compare our method against models using gloss as supervision and 3D data. The comparison is presented in Table 5.4. We can observe that our method still outperforms NSLPG (T2PG, short for Text-to-Pose with Gloss) [32] even with gloss supervision. When comparing with methods that output 3D predictions and only use text, Adversarial [63] and MDN [62], our process has better performance in all metrics besides BLEU-4 by a small margin. Although our method reaches values that are not as competitive as those achieved by using gloss supervision (MoMP [66] and FS-NET [67]), it is important to highlight our choice to avoid utilizing glosses as a form of supervision. We trade off performance for a model that requires less data, thus making it more viable for training in more sign languages. This enables more integration of the deaf community into society. An example of data unavailability is the How2Sign dataset, which currently does not have publicly available glosses.

Table 5.3: **State-of-the-art comparison.** Models were evaluated in both language and movements metrics. Our model outperforms other methods in generating precise signs across different language metrics. It also excels in the Movements metric, which measures how well the synthesized gestures match the ground-truth distribution. These results show that our model can produce more realistic and accurate signs than our competitors (Best in bold).

Language Metrics ↑

	Back Translation (Phoenix)				
Models	BLEU-4	BLEU-3	BLEU-2	BLEU-1	ROUGE
PT [64]*	7.60	9.87	14.20	25.27	27.35
NSLPG [32]	10.82	13.80	19.30	31.10	31.67
Ours	**11.27**	**14.71**	**20.78**	**33.86**	**33.38**

**Values extracted from the original paper*

Movements Metrics ↓

	Phoenix		How2Sign (Singer 8)		How2Sign (Singer 5)	
Models	FGD	MAEJ[1]	FGD	MAEJ[1]	FGD	MAEJ[1]
PT [64]	9.65	2.61	20.31	**2.42**	107.21	2.27
NSLPG [32]	3.41	2.62	13.18	2.47	94.17	**2.04**
Ours	**2.82**	**2.59**	**7.13**	2.49	**79.80**	2.14

[1] *Multiplied by 100*

Table 5.4: **Comparison against methods that are either supervised with gloss labels or utilize 3D data as outputs.** Results in Back Translation task on the Test set of the Phoenix dataset (Results reported in the original paper). Higher is better.

	Back Translation (Phoenix)				
Models	BLEU-4 ↑	BLEU-3 ↑	BLEU-2 ↑	BLEU-1 ↑	ROUGE ↑
NSLPG (T2PG) [32] *	10.88	13.79	18.94	30.06	31.32
Adversarial [63] †	10.81	13.72	18.99	30.93	32.74
MDN[62] †	11.68	14.55	19.70	31.56	33.19
MoMP [66] *†	13.30	16.86	23.27	35.89	36.77
FS-NET [67] *†	18.78	-	-	-	40.60
Ours	11.27	14.71	20.78	33.86	33.38

*: Utilizes gloss.
†: Utilizes 3D data as output.

5.6 Qualitative Evaluation

In Figure 5.3, we present a qualitative comparison between our model and NSLPG [32] that holds the state-of-the-art results with reproducible results. We did not compare with Progressive Transformers [64] since it yielded inferior results in the Text-to-Pose (T2P) settings.

In the upper portion of the figure, we compare the results of our model and NSLPG on the Phoenix dataset. Our model produces movements that are more faithful to the ground truth output. We highlight three frames where we draw the following observations: (a) the ground truth sign involves the hands above the shoulders and wide-open palms; (b) NSLPG, for its turn, produces a completely different sign with lower hands; (c) our model reproduces the movement in a very similar way, although the positioning of the left shoulder differs slightly. Overall, our model outperforms NSLPG in producing more accurate and faithful movements.

In the lower part of the figure, we compare the models in the How2Sign (signer 8) dataset. The same results from the Phoenix dataset can be seen in this evaluation, where our model generally makes signs more similar to the ground truth. We highlight the last frame (d), where the ground truth shows a movement with the stretched arms; (e) NSLPG does not follow it, whereas our model (f) does.

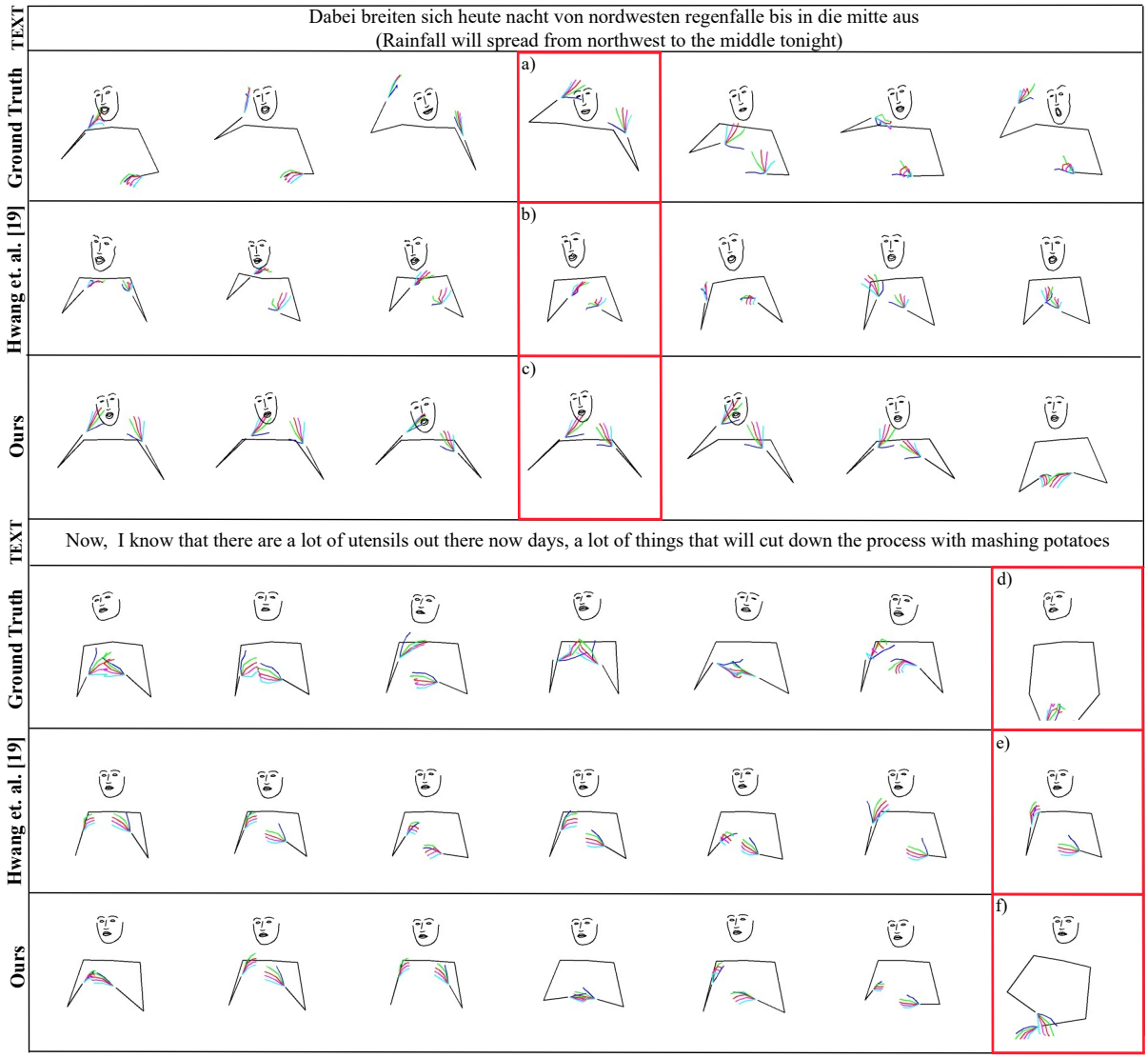
Figure 5.4 presents additional qualitative comparisons between our model and NSLPG [32]. In the upper portion of the figure, the comparison focuses on the Phoenix dataset. The sequences illustrate movements where our model demonstrates superior fidelity to the ground truth compared to NSLPG. In the first example, the ground truth exhibits intricate hand movements synchronized with facial expressions, conveying nuanced semantic information. NSLPG struggles to maintain this synchrony, resulting in gestures that lack the expressive depth observed in the ground truth. In contrast, our model effectively captures the interplay between hand movements and facial expressions, preserving the semantic integrity of the sign.

The middle section highlights a sequence from the How2Sign dataset for Signer 8, involving complex arm rotations. NSLPG simplifies the gesture, leading to a loss of contextual meaning, while our model more accurately follows the ground truth trajectory, producing a closer approximation of the sign language motion. This demonstrates our model's capacity to replicate complex gestures with greater fidelity.

The lower portion of the figure continues the evaluation on the How2Sign dataset, this time for Signer 5. In this case, our model generates movements that are more consistent with the ground truth. NSLPG's output appears to collapse into a narrow range of spatial movements, reducing the expressive quality of the signs. Although both models seem to be challenged by the data from Signer 5, our model produces more representative movements, indicating a greater robustness to data variations. This suggests that our model is better equipped to handle diverse signing styles while maintaining semantic accuracy.

These qualitative results underscore the robustness of our model in producing not only

Figure 5.3: **Qualitative evaluation.** Sign generation using text sequences from Phoenix dataset (upper) and How2Sign dataset (lower). In each sequence: First row: Input text; Second row: Ground truth; Third row: NSLP-G [32] result; Fourth row: Our method. Our model on the Phoenix dataset generates movements that closely match the ground truth output. In comparison, the movements produced by NSLPG diverged from the ground truth in the highlighted frames in red. On the How2Sign dataset, our model once again produces signs that are more accurate and similar to the ground truth, as evident in the last frame (d), where the ground truth shows a movement with stretched arms.



Source: Author.

accurate but also semantically faithful sign language movements across different datasets. The comparative analysis highlights the limitations of NSLPG in handling complex gestures, reinforcing our model's effectiveness in preserving the linguistic and expressive nuances of sign language.

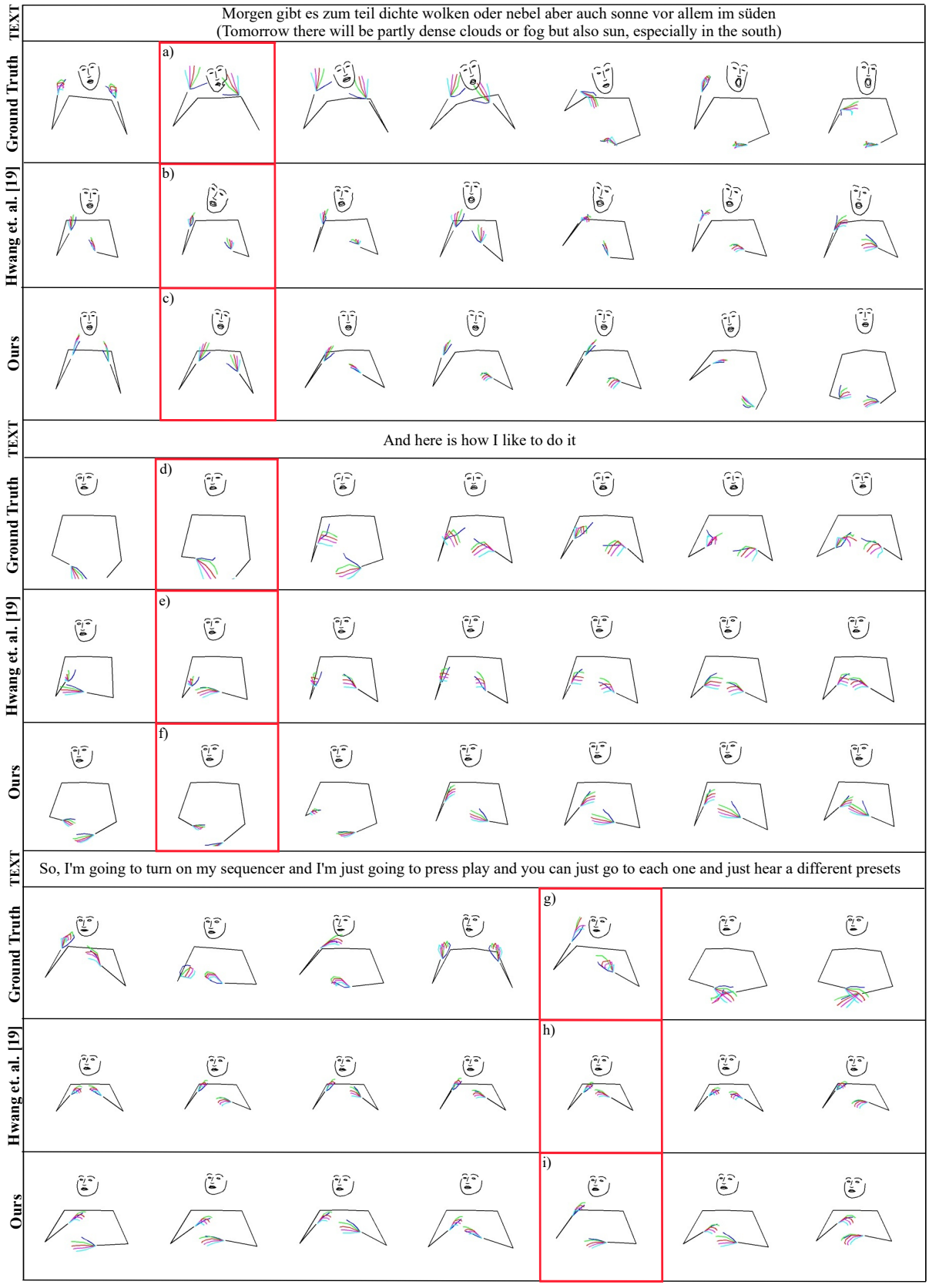
5.7 Summary & Closing Remarks

In this Chapter we presented our experiments details. First, we introduce the datasets utilized and how we processed its raw data. We used datasets from two different languages: German Sign Language (Phoenix14T) and American Sign Language (How2Sign). On the Phoenix14T we used all signers whilst on How2Sign we used the signers with more volume of data, namely signers 5 and 8. In the following section we detailed our architecture parameters as well our optimizers configurations. Section 5.3 presents our language and distance metrics, providing insights about how they work and its capacity of measuring model performance.

An ablation study of our model is shown in Section 5.4. We studied (Table 5.1) the impact of using a Continuous representation instead of a discrete one (W\o RVQVAE), not using LLMs (W\o LLMs) and not adding the Discrete Bias (W\o DB) in our architecture configuration. The results indicated that all of the components were effective in improve our model performance in the proposed metrics for our configuration. A sensitivity analysis (Table 5.2) regarding the number of Residual Layers was also conduced. We concluded that Residual Layers had a positive impact in model performance and it still retained performance when varying the number of layers from the optimal point.

In our quantitative evaluation (Section 5.5, we compared our model with the state-of-the-art in 2D text-to-sign synthesis (direct competitors) as well with models that uses more data in the training phase (indirect competitors). Our model presented overall better results in comparison with the direct competitors. When comparing with indirect competitors, our model presented better results in comparison with methods that did not use glosses. We stayed within a close range of most methods that used glosses with the benefit of using only text. The qualitative results (Section 5.6) corroborated with the quantitative results. Our method generated movements more close to the ground truth with better hand poses.

Figure 5.4: **Qualitative evaluation 2.** Sign language synthesis using text sequences from How2Sign dataset. First row: Input text; Second row: Ground truth; Third row: NSLP-G [2] result; Fourth row: Our method. When comparing the two methods, Hwang et al. method presents movements with hands completely closed and with movements under-represented compared to the ground truth (b). While the movements present by our method has hands with plausible opening and arm positions more close to the ground truth (c).



Source: Author

Chapter 6

Conclusion

This thesis introduces a new method for synthesizing continuous sign language using only written text. Our work exploits rich representations from pretrained LLMs and the discrete nature of sign languages. We proposed a method for synthesizing representative sign-language gestures from the text. Unlike previous methods, we use a intermediate discrete representation learned without supervision. Our model leverages Large Language Models (LLMs) to extract text representations and use a Transformer Decoder architecture to correlate these text representations with the discrete-to-continuous space. Moreover, we use a learning strategy that explores the benefit of two types of representations, quantized vectors (continuous) and their corresponding codes (discrete).

Our work is the first to focus solely on the text-to-sign task. Not relying on glosses considerably facilitates the process of collecting data and training the model to synthesize sign languages, enabling more inclusion for the deaf community. By using only text, we are able to leverage LLMs to increase our model language context. With the recent advances on LLMs, we believe that the SLP community could greatly benefit from this new approach. Our method is the first to take into account the discrete-continuous relationship between text and signs. The proposed architecture naturally fits the problem of translating discrete symbols into continuous sign motions.

We conducted experiments and an ablation study using the PHOENIX14T dataset [9] and the How2Sign dataset [18]. Our ablation study showed the importance of using LLMs feature vectors. The overall metrics worsened an average of 30% in comparison to our complete model when removing the feature vectors. The importance of adding a Discrete Bias in the sign synthesis is also verified in our ablation. In the sensitivity study, we observed that our Residual VQ layer had a positive impact when comparing to a single VQ layer and is robust, as it generalizes for different number of layers. When comparing with other models, our approach surpassed the Progressive Transformers [64] and NSLP-G [32], in terms of both language metrics (BLEU-1 to BLEU-4 n-grams and ROUGE) and movement metrics (FGD and MAEJ), achieving state-of-the-art in the text-to-sign task. These results indicates that our model sythesizes signs closer to what is expected both spatially and semantically. When comparing with models that uses more data, we still stand as state-of-the-art text-based model. We also presented competitive results against models that uses glosses in the training phase, without the

drawback of needing an expensive annotation such as glosses. These results indicate that our intermediate discrete representation strategy, coupled with training to utilize both continuous and discrete representations, enhances the ability to generate accurate and representative signs.

6.1 Limitations and Future Work

Even though our method achieves state-of-the-art results in the 2D text-to-sign synthesis task, it still has some limitations. For example, our architecture depends on the availability of pretrained LLMs in the desired written language. Besides that, our work is trained with 2D landmarks, which imposes restrictions on its usage on downstream tasks compared to a 3D model. Another improvement could be made in the inference phase, where we follow the models in the area and use the length of the input sequence to drive the synthesis and cut the predicted output. Our model is capable of generating outputs given we know this information beforehand, and this is currently a limitation in the field. We also acknowledge that although we present advances in the SLP area, the area still needs to achieve results that could be used in applications that strictly depend on correct signs. Our method is not capable of representing the more complex expressions that sign language has which involves intense facial expressions and varied hand motions.

As future work, we intend to improve the RVQ-VAE module by using GANs [26] in a similar way to Esser *et al*. [21] work. Improving the quality of the face and hand expressions is a key step to improve the conveyed message to the deaf community and its one of the struggling points for the SLP community. This approach could led to more realistic facial and hand expressions. We also would like to extend our model to predict 3D keypoints. When working with 2D points inevitably we lose information. Adding depth will give the model more context and probably improve the results with the trade-off of increasing the model complexity alongside with more noise from the data collection process. In addition, the predicted 3D points can be used to animate 3D virtual models or even humanoids robots, expanding the applications in the user end. Exploring the correlations between sentiments and signs could greatly improve the quality of the results. The intensity and form of sign expressions are directly correlated to the sentiment conveyed by the performer. Adding sentiment information in the synthesis process is a natural approach that we like to test. A crucial point of extension is to focus on investigate approaches that does not rely on knowing the length of the output sentence in the inference phase. Some recent models such as the work of Huang [30] made advances in this direction but at the cost of the quality of the generated signs.

References

[1] Michel Abou-Abdallah and Abigail Lamyman. Exploring communication difficulties with deaf patients. *Clinical Medicine*, 21:clinmed.2021–0111, 06 2021.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikoł aj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736. Curran Associates, Inc., 2022.

[3] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1171–1179, Cambridge, MA, USA, 2015. MIT Press.

[5] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[6] Danielle Bragg, Oscar Koller, Mary Bellard, Larwan Berke, Patrick Boudreault, Annelies Braffort, Naomi Caselli, Matt Huenerfauth, Hernisa Kacorri, Tessa Verhoef, Christian Vogler, and Meredith Ringel Morris. Sign language recognition, generation, and translation: An interdisciplinary perspective. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '19, page 16–31, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Mary Brennan. Deafness, disability and inclusion: The gap between rhetoric and practice. *Policy Futures in Education*, 1(4):668–685, 2003.

[8] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[9] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. Neural sign language translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7784–7793, 2018.

[10] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, and Richard Bowden. Neural sign language translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7784–7793, 2018.

[11] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[12] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[13] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST@EMNLP*, 2014.

[14] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[15] Carl d. Boor. *A Practical Guide to Splines*. Springer Verlag, New York, 1978.

[16] Mirella De Sisto, Vincent Vandeghinste, Santiago Egea Gómez, Mathieu De Coster, Dimitar Shterionov, and Horacio Saggion. Challenges with sign language datasets for sign language recognition and translation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2478–2487, Marseille, France, June 2022. European Language Resources Association.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] Amanda Duarte, Shruti Palaskar, Lucas Ventura, Deepti Ghadiyaram, Kenneth DeHaan, Florian Metze, Jordi Torres, and Xavier Giro-i Nieto. How2Sign: A Large-scale Multimodal Dataset for Continuous American Sign Language. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[19] Sarah Ebling and Matt Huenerfauth. Bridging the gap between sign language machine translation and sign language animation using sequence classification. In *Proceedings of SLPAT 2015: 6th Workshop on Speech and Language Processing for Assistive Technologies*, pages 2–9, Dresden, Germany, September 2015. Association for Computational Linguistics.

[20] Ralph Elliott, John Glauert, Richard Kennaway, Ian Marshall, and Eva Sáfár. Linguistic modelling and language-processing technologies for avatar-based sign language presentation. *Universal Access in the Information Society*, 6:375–391, 02 2008.

[21] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, June 2021.

[22] Jeffrey De Fauw, Sander Dieleman, and Karen Simonyan. Hierarchical autoregressive image models with auxiliary decoders. *arXiv preprint arXiv:1903.04933*, 2019.

[23] João P. Ferreira, Thiago M. Coutinho, Thiago L. Gomes, José F. Neto, Rafael Azevedo, Renato Martins, and Erickson R. Nascimento. Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio. *Computers & Graphics*, 94:11–21, 2021.

[24] Anindita Ghosh, Noshaba Cheema, Cennet Oguz, Christian Theobalt, and Philipp Slusallek. Synthesis of compositional animations from textual descriptions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1396–1406, October 2021.

[25] John R. W. Glauert, Ralph Elliott, Stephen J. Cox, Judy Tryggvason, and Mary Christine Anne Sheard. Vanessa - a system for communication between deaf and hearing people. *Technology and Disability*, 18:207–216, 2006.

[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[29] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. Scaling up vision-language pretraining for image captioning. pages 17959–17968, 06 2022.

[30] Wencan Huang, Wenwen Pan, Zhou Zhao, and Qi Tian. Towards fast and high-quality sign language production. *Proceedings of the 29th ACM International Conference on Multimedia*, 2021.

[31] Wencan Huang, Zhou Zhao, Jinzheng He, and Mingmin Zhang. Dualsign: Semi-supervised sign language production with balanced multi-modal multi-task dual transformation. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 5486–5495, New York, NY, USA, 2022. Association for Computing Machinery.

[32] Euijun Hwang, Jung-Ho Kim, and Jong-Cheol Park. Non-autoregressive sign language production with gaussian space. In *The 32nd British Machine Vision Conference (BMVC 21)*, 2021.

[33] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*, 34(6, Part A):2515–2528, 2022.

[34] K. Karpouzis, G. Caridakis, S.-E. Fotinea, and E. Efthimiou. Educational resources and implementation of a greek sign language synthesis architecture. *Computers & Education*, 49(1):54–74, 2007. Web3D Technologies in Learning, Education and Training.

[35] Dilek Kayahan and Tunga Güngör. A hybrid translation system from turkish spoken language to turkish sign language. *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6, 2019.

[36] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[37] Dimitris Kouremenos, Klimis S. Ntalianis, Georgios Siolas, and Andreas Stafylopatis. Statistical machine translation for greek to greek sign language using parallel corpora produced via rule-based machine translation. In *CIMA@ICTAI*, 2018.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[39] Doyup Lee, Chiheon Kim, Kim Saehoon, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[40] Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. Pretrained language model for text generation: A survey. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4492–4499. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Survey Track.

[41] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++, 2021.

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

[43] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[44] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. cite arxiv:1907.11692.

[45] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[47] John Mcdonald, Rosalee Wolfe, Jerry Schnepp, Julie Hochgesang, Diana Gorman Jamrozik, Marie Stumbo, Larwan Berke, Melissa Bialek, and Farah Thomas. An automated technique for real-time production of lifelike animations of american sign language. *Univers. Access Inf. Soc.*, 15(4):551–566, nov 2016.

[48] Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[49] Medhini Narasimhan, Anna Rohrbach, and Trevor Darrell. Clip-it! language-guided video summarization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13988–14000. Curran Associates, Inc., 2021.

[50] Evonne Ng, Hanbyul Joo, Liwen Hu, Hao Li, , Trevor Darrell, Angjoo Kanazawa, and Shiry Ginosar. Learning to listen: Modeling non-deterministic dyadic facial motion. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[51] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[52] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[54] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10985–10995, October 2021.

[55] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-conditioned 3D human motion synthesis with transformer VAE. In *International Conference on Computer Vision (ICCV)*, 2021.

[56] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

[57] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[58] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

[59] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[60] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[61] Wendy Sandler and Diane Lillo-Martin. *Sign Language and Linguistic Universals*. Cambridge University Press (CUP), The Edinburgh Building, Cambridge CB2 2RU, UK, 02 2006.

[62] Ben Saunders, Necati Cihan Camgoz, and R. Bowden. Continuous 3d multi-channel sign language production via progressive transformers and mixture density networks. *Int. J. Comput. Vis.*, 129:2113–2135, 2021.

[63] Ben Saunders, Necati Cihan Camgoz, and Richard Bowden. Adversarial Training for Multi-Channel Sign Language Production. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2020.

[64] Ben Saunders, Necati Cihan Camgoz, and Richard Bowden. Progressive Transformers for End-to-End Sign Language Production. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[65] Ben Saunders, Necati Cihan Camgoz, and Richard Bowden. Continuous 3d multi-channel sign language production via progressive transformers and mixture density networks. volume 129, pages 2113–2135, Jul 2021.

[66] Ben Saunders, Necati Cihan Camgöz, and Richard Bowden. Mixed signals: Sign language production via a mixture of motion primitives. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 1899–1909. IEEE, 2021.

[67] Ben Saunders, Necati Cihan Camgöz, and Richard Bowden. Signing at scale: Learning to co-articulate signs for large-scale photo-realistic sign language production. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[68] Stephanie Stoll, Necati Cihan Camgöz, Simon Hadfield, and R. Bowden. Sign language production using neural machine translation and generative adversarial networks. In *BMVC*, 2018.

[69] Stephanie Stoll, Necati Cihan Camgoz, Simon Hadfield, and Richard Bowden. Text2sign: Towards sign language production using neural machine translation and generative adversarial networks. volume 128, pages 891–908, Apr 2020.

[70] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[71] Shengeng Tang, Richang Hong, Dan Guo, and Meng Wang. Gloss semantic-enhanced network with online back-translation for sign language production. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 5630–5638, New York, NY, USA, 2022. Association for Computing Machinery.

[72] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[74] Lilian Weng. Attention? attention! *lilianweng.github.io*, 2018.

[75] World Health Organization (WHO). Deafness and hearing loss. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss, 2023. Accessed: 2023-03-06.

[76] Ronnie Wilbur. Phonological and prosodic layering of nonmanuals in american sign language. *The signs of language revisited: An anthology to honor Ursula Bellugi and Edward Klima*, pages 190–214, 01 2000.

[77] Sijie Yan, Zhizhong Li, Yuanjun Xiong, Huahan Yan, and Dahua Lin. Convolutional sequence generation for skeleton-based action synthesis. In *International Conference on Computer Vision (ICCV)*, pages 4394–4402, 2019.

[78] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.

[79] Sheng Ye, Yu-Hui Wen, Yanan Sun, Ying He, Ziyang Zhang, Yaoyuan Wang, Weihua He, and Yong-Jin Liu. Audio-driven stylized gesture generation with flow-based model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 712–728, 2022.

[80] Kayo Yin, Amit Moryossef, Julie Hochgesang, Yoav Goldberg, and Malihe Alikhani. Including signed languages in natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7347–7360, Online, August 2021. Association for Computational Linguistics.

[81] Youngwoo Yoon, Bok Cha, Joo-Haeng Lee, Minsu Jang, Jaeyeon Lee, Jaehong Kim, and Geehyuk Lee. Speech gesture generation from the trimodal context of text, audio, and speaker identity. *ACM Transactions on Graphics*, 39(6), 2020.

[82] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*, 2022.

[83] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[84] Unai Zabala, Igor Rodriguez Rodriguez, Jose Maria Martinez-Otzeta, and Elena Lazkano. Modeling and evaluating beat gestures for social robots. *Multimedia Tools and Applications*, 81, 01 2022.

[85] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30:495–507, jan 2022.

[86] Jan Zelinka and Jakub Kanis. Neural sign language synthesis: Words are our glosses. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3384–3392, 2020.