2024 18th International Conference on Automatic Face and Gesture Recognition (FG)

A Data-Driven Representation for Sign Language Production

Harry Walsh[1], Abolfazl Ravanshad[2], Mariam Rahmani[2] and Richard Bowden[1]
[1] CVSSP, University of Surrey, Guildford, United Kingdom
[2] OmniBridge.ai, an Intel Venture, USA

Abstract— **Phonetic representations are used when recording spoken languages, but no equivalent exists for recording signed languages. As a result, linguists have proposed several annotation systems that operate on the gloss or sub-unit level; however, these resources are notably irregular and scarce.**

Sign Language Production (SLP) aims to automatically translate spoken language sentences into continuous sequences of sign language. However, current state-of-the-art approaches rely on scarce linguistic resources to work. This has limited progress in the field. This paper introduces an innovative solution by transforming the continuous pose generation problem into a discrete sequence generation problem. Thus, overcoming the need for costly annotation. Although, if available, we leverage the additional information to enhance our approach.

By applying Vector Quantisation (VQ) to sign language data, we first learn a codebook of short motions that can be combined to create a natural sequence of sign. Where each token in the codebook can be thought of as the lexicon of our representation. Then using a transformer we perform a translation from spoken language text to a sequence of codebook tokens. Each token can be directly mapped to a sequence of poses allowing the translation to be performed by a single network. Furthermore, we present a sign stitching method to effectively join tokens together. We evaluate on the RWTH-PHOENIX-Weather-2014T (PHOENIX14T) and the more challenging meineDGST (mDGS) datasets. An extensive evaluation shows our approach outperforms previous methods, increasing the BLEU-1 back translation score by up to 72%.

I. INTRODUCTION

Sign language is a rich and complex form of communication that relies on visual-spatial elements rather than spoken words [42]. It serves as the primary mode of communication for the deaf community [45].

Sign languages are composed of cheremes, analogous to the phonemes found in spoken languages [29]. Cheremes from the Greek word for hand, describe features such as handshape, orientation, location, movement and non-manual expressions. These fundamental units can be combined to create a natural sequence of signing. When transcribing sign language, linguists commonly employ sub-unit and gloss[1] level representations [14], [44]. Unfortunately, curating transcriptions is time-consuming and costly, and as a result, linguistic resources are often limited or even non-existent [3].

Sign Language Production (SLP) is the task of translating from a spoken language sentence to a continuous sign language sequence. To facilitate natural communication, SLP must include both manual and non-manual components [2]. Previous works have attempted to learn a direct mapping from Text-to-Pose (T2P). However, they suffer from regression to the mean [38]. Alternative two-step methods (Text-to-Gloss-to-Pose (T2G2P)) rely on expensive linguistic annotation [13], [15], [37].

In this paper, we propose creating a data-driven representation of sign language that can be used as a replacement for expensive linguistic annotation. Our approach learns a codebook of motions from continuous 3D pose data using a Noise Substitution Vector Quantization (NSVQ) model [47]. The codebook can be considered the lexicon of our new representation and used to tokenise a continuous pose sequence into a sequence of discrete codes. As depicted in Fig. 1, we then tackle the problem as a traditional sequence-to-sequence task, translating from a spoken language sentence (Fig. 1.1) to a sequence of codebook tokens (Fig. 1.2). Unlike the previous two-step approaches our intermediary representation can be directly mapped to a sequence of poses (Fig. 1.3) and includes non-manual key points. Furthermore, we show how our representation can be enhanced when limited linguistic annotation is available, and by introducing a novel stitching module we create more natural continuous signing.

We show state-of-the-art results on RWTH-PHOENIX-Weather-2014**T** (PHOENIX14**T**) and the more challenging Meine DGS Annotated (mDGS) dataset. An extensive ablation study reveals the effectiveness of our approach compared to previous works. Increasing the back translation scores by up to 72%. Finally, we share quantitative results.

We can summarise the contribution of this paper as:

- A novel architecture for creating a data-driven representation of sign language.
- Sign stitching, a method to improve the back translation performance.
- A contrastive learning approach that enhances the representation.
- State-of-the-art SLP performance on PHOENIX14**T** and the more challenging mDGS dataset.

II. RELATED WORK

A. Sign Language Translation (SLT)

Over the past three decades, computational sign language translation has remained an active area of interest [46]. Initial

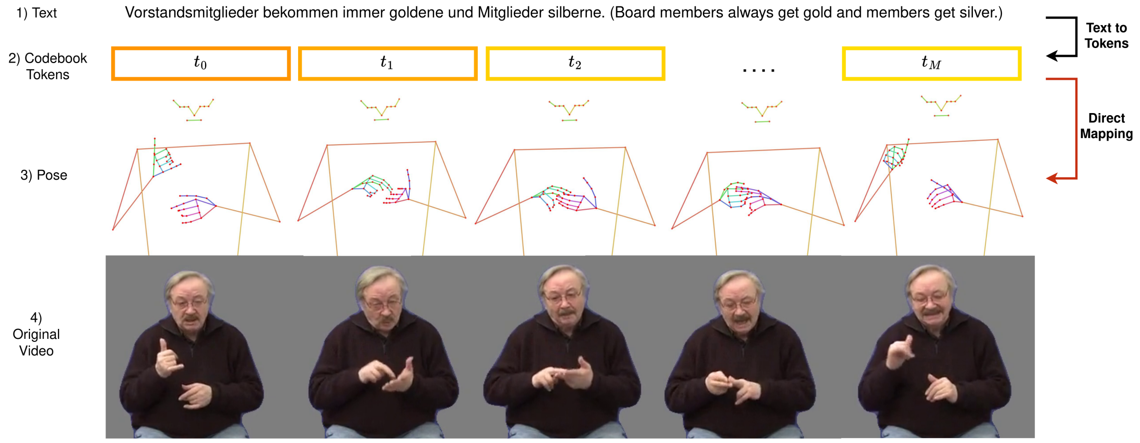
This work was supported by Intel, the SNSF project 'SMILE II' (CRSII5 193686), the European Union's Horizon2020 programme ('EASIER' grant agreement 101016982) and the Innosuisse IICT Flagship (PFFS-21-47). This work reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

[1] Gloss is the written word associated with a sign

[2] Manual components include hand shape and motion. While non-manual components include facial expressions, body movements, eye gaze etc.

979-8-3503-9494-8/24/$31.00 ©2024 IEEE

Fig. 1. A overview of our approach to Sign Language Production (SLP). Showing 1) the source spoken language sentence, 2) our intermediate representation of sign, 3) the synthesized sequence of signing, and, 4) the original video.


1) Text
Vorstandsmitglieder bekommen immer goldene und Mitglieder silberne. (Board members always get gold and members get silver.)
2) Codebook Tokens
t_0
t_1
t_2
....
t_M
Text to Tokens
Direct Mapping
3) Pose
4) Original Video

research focused on isolated Sign Language Recognition (SLR), where a Convolutional Neural Network (CNN) is used to classify a single instance of a sign [30]. Advances in the field led to Continuous Sign Language Recognition (CSLR), which requires both the segmentation of a video into its constituent signs and their classification [26]. Since then the more challenging task of Sign-to-Text (S2T) was introduced [4], where continuous sign language sequences are directly converted into spoken language sentences. However, it is shown that translating via a gloss intermediary gives better performance (Sign-to-Gloss-to-Text (S2G2T). Transformers [48] have been applied to the problem, and demonstrated state-of-the-art performance [5]. Thus, we utilise this architecture to evaluate the performance of our approach, similar to [37], [38], [39].

B. *Sign Language Production (SLP)*

SLP is the task of translating from spoken language sentences to a continuous sequence of sign language. Early approaches to SLP use animated avatars with a dictionary lookup [10], [20], [33]. The first deep SLP pipeline broke down the task into three steps, first, a translation from Text-to-Gloss (T2G) followed by a second Gloss-to-Pose (G2P) translation and finally a mapping from Pose to Sign (P2S) [43]. Saunders et al. introduced the Progressive transformer [38] an encoder-decoder transformer architecture that generates a continuous pose sequence given a spoken language sentence (T2P), simplifying the SLP pipeline. They demonstrated that better translation results were achieved using a gloss intermediary (T2G2P). However, the approach suffers from regression to the mean, resulting in under-articulated signing. In an attempt to reduce the problem, adversarial training and Mixture Density Network (MDN) are added to the model [37]. Alternatively, Huang et al. proposed a non-autoregressive G2P architecture [15], which produces the sign sequence in a single step.

Alternative representations to gloss have been explored namely the Hamburg Notation System (HamNoSys) [14] and SignWriting [44]. HamNoSys is a transcription system that is used to describe sign language at the phonetic level, where each sign consists of a description of the initial posture and the action over time. HamNoSys can be mapped directly to an avatar making it a suitable alternative for gloss in the SLP pipeline [21]. Furthermore, Walsh et al. defined the task of Text-to-HamNoSys (T2H) and showed improved translation performance by using it as an intermediate representation [49]. A similar translation task has been explored with translating text to SignWriting [18] and animating from SignWriting [2].

Modern deep learning is heavily dependent on data, approximately 15 million sentence pairs are required before deep learning starts to outperform statistical approaches [25]. In contrast, sign language datasets are limited. For example, mDGS has only 50k parallel sentences with gloss and HamNoSys annotations [27]. These annotation systems are time-consuming and costly to create, and this has limited the size of the available datasets. Therefore, in this paper, we propose learning a representation from 3D pose data that can be used as a substitute for gloss or HamNoSys. But unlike gloss, our representation can be mapped directly to a sequence of poses, removing the need for expensive annotation. But to build a discrete vocabulary we need to quantise the data.

C. *Vector Quantized Models*

Kingma et al. [24] introduced the first Variational Autoencoder (VAE) and showed impressive results. However, they struggled to capture fine-grained structures. Van Den Oord [34] improved on this by introducing the Vector Quantized Variational Autoencoder (VQ-VAE) architecture. The model integrates Vector Quantisation (VQ) into the latent space of a VAE. Forcing the embedding space of the VAE to be discrete, allowing state-of-the-art image and audio generation. Since then, VQ has been applied to several problems including music generation using a hierarchical VQ-VAE [6], speech synthesis using self-supervised training [1], and more recently image generation using a diffusion model [13]. The original VQ-VAE architecture used an argmin operation to select the closest matching codebook entry. As a result, the model used straight-through gradient estimation to make the model differentiable. Consequently, the model uses three separate losses to train: a reconstruction loss, a codebook loss and a commitment loss. Kaiser et al. [19] used an exponentially moving average to update the codebook. This simultaneously helped stabilise training and reduced the required loss functions to two. Vali et al. [47] then reduced the required losses to one by introducing the NSVQ technique. Here the vector quantization error is approximated by substituting it for a product of the original error and a normalised noise vector. The result allows for end-to-end training of the model showing faster convergence compared to straight-through estimation and exponential moving averages models. Therefore, we adopt this architecture in our approach.

VQ has been previously applied to the SLP problem [16], [50]. Xie et al. broke the human skeleton into three separate codebooks and used a diffusion model with codeUnet [50] to translate from G2P. The approach still relies on expensive linguistic annotation, and qualitative results show a lack of detail in the hands resulting in under-articulated signing. In contrast, we propose approaching the task using a transformer to construct the codebook and perform the translation. We believe that the attention mechanism is more adept to modelling long-range dependencies and the change in order between the source and target. Consequently, we apply our approach to the more challenging task of T2P translation and show higher back translation scores.

III. METHODOLOGY

The aim of SLP is to enable seamless translation from spoken to signed languages. To accomplish this, we convert a source spoken language sequence, denoted as $X = (x_1, x_2, ..., x_W)$ with W words, into a continuous sequence of poses, denoted as $P = (p_1, p_2, ..., p_U)$ with U frames. Where each pose consists of J joints in D dimensional space e.g. $p_i \in \mathbb{R}^{J \times D}$. SLP is a significant challenge, considering that the target length is substantially greater than the source, such as $U >> W$. This inherent difficulty persists even when employing state-of-the-art sequence-to-sequence models for the translation task [48]. To overcome this, we first learn a codebook of tokens, that each represent a short sequence of signing and can be directly mapped back to a sequence of poses. Then we perform a translation from spoken language text to a sequence of latent codes, as shown in Fig. 1.1 to 1.2. The individual architectures of this pipeline are illustrated in Fig. 2, and we elaborate on each stage in the subsequent sections.

A. Pose Codebook

The objective of the codebook is to learn a set of motions from a dataset of continuous signing. Our approach employs a transformer encoder-decoder architecture with a NSVQ. Next, we explain each module in turn, following Fig. 2.a from left to right;

Encoder: Given a short sequence of poses, $P = (p_1, p_2, ..., p_{U_{cb}})$ with U_{cb} frames, we add positional encoding to each pose. We then embed the sequence using a linear layer which acts only in the spatial dimension. Then the sequence is passed to the spatial-temporal transformer encoder, allowing the network to learn long-range dependencies within the sequence. The embedded features can be defined as $z \in \mathbb{R}^{U_{cb} \times H}$, where H is the embedding size. Note we train each codebook entry to represent a sub-unit of a full continuous sequence, hence $U_{cb} << U$.

VQ: The NSVQ codebook learns a set of tokens from the encoder, we denote the codebook as $CB = [t_1, t_2, ..., t_N]$, where N is the number of tokens in the codebook and each $t_i \in \mathbb{R}^{U_{cb} \times H}$. Therefore, the length of each pose sequence, U_{cb}, determines how many frames each codebook token represents. To train this module, each output from the encoder, z, is mapped to a single codebook token t_i. This is called VQ and is defined as;

$$t_i \ ; \ i = \arg\min_{t_i} ||z - t_i||^2 \tag{1}$$

Eq. (1) is non-differentiable. To overcome this the NSVQ simulates the quantization error by adding noise to the input vector, such that the simulated noise forms the same distribution as the original VQ error. The NSVQ is trained end-to-end and the output to the decoder can be defined as;

$$\hat{z} = z + ||z - t_i|| * \frac{V}{||V||} \tag{2}$$

Where V is a normally distributed noise source. Fig. 2.a (NSVQ) depicts how Eq. (1) and (2) are used during training.

Decoder: The decoder learns to reconstruct the original pose sequence from the quantized embedding. Here we use counter decoding from Saunders et al. [38] to drive the decoder, and therefore, we use non-autoregressive decoding. Meaning a sequence is processed in a single step, for reduced computational cost and faster inference speeds. We find using this approach outperforms a simple multilayer perceptron on reconstruction error. The value of the counter is defined as;

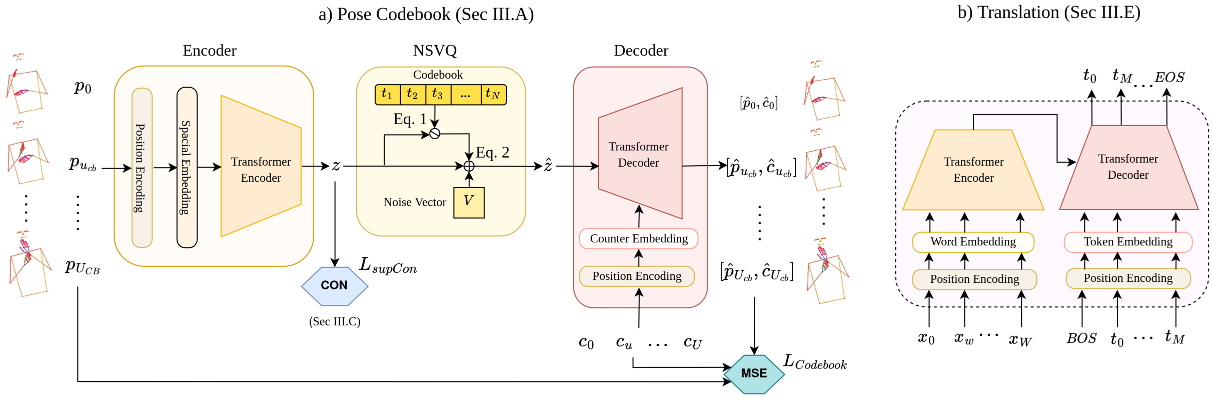
$$c_u = \frac{u + 1}{U_{cb}} \tag{3}$$

Where u is the current position in the sequence and U_{cb} is the total sequence length. As shown in Fig. 2.a (Decoder), we add positional encoding and use a linear layer to embed the counter values. We then apply a spatial-temporal transformer decoder, that uses cross and self-attention to produce the output embedding. From this, we project the embedding back to the pose and counter values using two linear layers. Our architecture is trained end-to-end using the following loss function;

$$L_{Codebook} = \frac{1}{U_{cb}} \sum_{u=1}^{U_{cb}} (P_u - \hat{P_u})^2 + \alpha(c_u - \hat{c_u})^2 \tag{4}$$

where α is a scaling factor that we determine empirically and $\hat{P}$, $\hat{c}$, are the predicted pose and counter values.

Fig. 2. An overview of the architecture used in our approach. Showing a) The Codebook training architecture and b) the Text-to-Codebook Tokens Translation architecture.



B. Codebook Replacement

Codebook collapse is a significant challenge when training codebooks with VQ [7]. This is when several tokens within the codebook are no longer selected during the quantization process, resulting in dead codebook tokens. This can occur when the data distribution of the embedding space no longer matches the tokens. Strategies exist to detect and replace these dead tokens [9], [47], [51].

We employ two replacement strategies to reduce dead codebook entries and evenly distribute active entries. Codebook entries whose usage fall below a threshold percentage are replaced with either, 1) a randomly selected active entry, plus a small magnitude of normal noise, or, 2) a randomly selected embedding from the encoder, z. By tracking the usage of each token over a given number of batches we can determine active tokens when the percentage used is greater than β and dead tokens when the percentage used is less than γ. We set a schedule for training, initially using replacement more often and slowly decreasing the frequency throughout training. Once the learning rate decreases past a given factor we stop using replacement allowing the network to fine tune its parameters.

C. Contrastive Learning

When linguistic annotation is available we apply an additional contrastive loss. Specifically, we add a supervised contrastive loss [22] to the encoder of the codebook transformer, as shown in Fig. 2. This makes use of gloss labels and time stamps to tag each input sequence with its corresponding gloss ID. For long input sequences that contain frames from multiple glosses, we select the most common ID. The contrastive loss pulls sequences belonging to the same gloss together, while simultaneously pushing apart sequences belonging to different glosses. We hypothesise that the additional loss allows the encoder to overcome the natural variation between signers, helping the model become person-invariant. We define the loss as,

$$L_{supCon} = \sum_{i=0}^{I} \frac{-1}{|A(i)|} \sum_{a=0}^{A(i)} \left(\frac{exp(z_i \cdot z_a/\tau)}{\sum_{b=1}^{B(i)} exp(z_i \cdot z_b/\tau)} \right) \quad (5)$$

Here i is the index of the anchor. $A(i)$ is the set of indices that correspond to positive samples in the batch and $|A(i)|$ is the number of samples in a batch. While, $B(i)$, is the set of the negative samples. τ is a scalar temperature parameter. We define a sequence to be positive if it shares the same gloss ID with the anchor, while we define a negative sample if it has a different ID. The contrastive loss is scaled by δ before being added to the codebook loss. Therefore, the total loss is defined as;

$$L_{Total} = L_{Codebook} + \delta L_{supCon} \quad (6)$$

D. Pose Sequence Tokenization

To perform a translation from text to tokens, we first tokenize the continuous pose sequence. We build the codebook to be a sub-unit representation. Thus, given a continuous sequence of poses, P, we create a sequence of tokens, $T = (t_1, t_2, ..., t_M)$ where M is the number tokens, which corresponds to the length of the original sequence, such that $M = \lfloor U_{cb}/U \rfloor$. Therefore, when tokenizing a sequence we lose any tailing frames. We freeze the encoder and codebook and pass each segment through the encoder. To find the corresponding token we then apply Eq. (1).

De-tokenization: A mapping between the codebook tokens and their corresponding pose sequences is obtained by passing each token through the decoder, such that;

$$P = D(T) \quad (7)$$

We apply this mapping when evaluating the translation model's performance in the pose space.

E. Text-to-Codebook Translation

Given a spoken language sequence, $X = (x_1, x_2, ..., x_W)$ we aim to produce the corresponding sequence of codebook tokens, $T = (t_1, t_2, ..., t_M)$, therefore the translation model learns the conditional probability $p(T|X)$. First, positional encoding is added to the spoken language sequence, X. After it is embedded with a linear layer and passed through the encoder giving the context embedding used by the decoders cross attention layers. We apply autoregressive decoding, starting with the beginning of sentence token and we continue decoding until the end of sentence token is predicted, as illustrated in Fig. 2. Similar to the encoder, positional encoding is added before each token is embedded using a linear layer.

F. Codebook Stitching

As discussed, the predicted sequence of tokens, T, can be directly mapped to a sequence of poses, P. However, discrepancies may arise between the final pose of one token and the initial pose of the next, resulting in discontinuities. To address this we employ linear interpolation to stitch codebook entries together, as a result we generate more natural continuous sequences. Furthermore, to maintain temporal consistency with the original sequence we fit a high-order spline curve [8] and re-sample. This maintains the number of poses in the sequence.

IV. EXPERIMENTAL SETUP

A. Implementation Details

In our experiments, we search for the best hyper-parameters and find the following settings the most effective. We build our encoder-decoder translation model using a single layer with four heads, opting for an embedding size of 512 and a feed-forward size of 1024. The resultant architecture contains 7.8 million parameters. When decoding we employ a beam search algorithm with a size of 5 and a length penalty of 2.0.

Our codebook model consists of a smaller encoder-decoder that has 1.2 million parameters. The model has 2 layers with 4 heads and is built with an embedding and feed-forward size of 128. We set a codebook replacement threshold of 0.1%. Initially, we conduct replacement once per epoch and gradually reduce the frequency by 10 every 50 epochs. We set the initial learning rate to 10^{-4} and stop the codebook replacement once the learning rate reaches 10^{-6}.

Both models employ dropout with a probability of 0.1 [41]. We use Relu activation between the layers and apply pre-layer normalisation for regularisation and training stability. We train with a reduce on plateau scheduler with a patience of 5 and a decrease factor of 0.9. To initialize the transformer encoder and decoder layers we employ a Xavier initializer [11] with zero bias and Adam optimization [23]. The learning rate is initially set to 10^{-4} and we train the model till convergence. Our translation model code base comes from the Kreutzer et al. NMT toolkit, JoeyNMT [28] and is implemented using Pytorch [36].

For comparison on the mDGS dataset, we train two variants of the progressive transformer till convergence with the settings presented in [38].

B. Dataset

To assess our models, we employ the Public Corpus of German Sign Language, 3rd release, the mDGS dataset [27], and the PHOENIX14**T** dataset (as introduced by Camgoz et al., 2018 [4]). The mDGS dataset is comprised of aligned spoken German sentences and gloss sequences, obtained from unconstrained dialogues between two native deaf signers. Whereas the PHOENIX14**T** dataset comes from German weather broadcasts and includes 8257 sequences performed by 9 signers. Resulting in 1066 signed glosses and a spoken language vocabulary of 2887. Notably, the mDGS is 7.5 times larger than PHOENIX14**T**, featuring 330 deaf participants engaging in free-form signing and a spoken language vocabulary of 18,457. We follow the formatting conventions set out by Saunders et al. in [40]. In addition, we eliminate gloss variant numbers to mitigate singletons when translating G2P, in Section V-A.5.

To obtain the pose from the original video, we employ Mediapipe to extract 61 2D keypoints (comprised of 21 for each hand, 9 for the body, and 10 for the face) [32]. To ensure the accurate elevation from 2D to 3D, we adopt the methodology outlined in [17]. This approach utilises forward kinematics and a neural network to predict both bone lengths and angles from the 2D pose. Each pose is represented as a hierarchical tree, enforcing physical limits to constrain the pose and ensure it remains valid. When two camera angles are available (such as in the mDGS dataset) we extract 3D pose using the method above and run an additional optimization, minimizing the error between the two predicted poses. The mDGS and PHOENIX14**T** datasets are captured at 50 and 25 frames per second (fps), respectively. We reduce the frame rate by subsampling each pose sequence by a factor of 3, this removes redundant information and speeds up training.

C. Evaluation Metrics

For evaluation purposes, we employ the back translation metric [38]. For which we use the state-of-the-art CSLR architecture (Sign Language Transformers [5]), the same as [15], [37], [39], [50]. The model is a transformer encoder-decoder which predicts a spoken language sentence given a pose sequence. The model employs Connectionist Temporal Classification (CTC) loss [12] as additional supervision to predict gloss tokens. We train a model for each dataset and freeze the parameters so results are consistent across runs. We compute BLEU scores (BLEU-1,2,3, and 4) [35] and ROUGE scores [31] against the original input text or gloss.

To evaluate the accuracy of the poses we use Dynamic Time Warping Mean Joint Error (DTW-MJE), this metric aligns two-time series by stretching or compressing them locally in time to find the optimal match, minimizing the overall distance between the ground truth (GT) and predicted. Thus, we first calculate the index alignment;

$$A_{i,j} = DTW(p_u, \hat{p}_u) \tag{8}$$

After the alignment, we compute the mean joint error between the two;

$$\text{DTW-MJE} = \sum_{u=0}^{U} |p_u[A_i] - \hat{p}_u[A_j]| \tag{9}$$

Note we normalise the skeletons between the range of zero and one before calculating DTW-MJE.

TABLE I

The results of translating from spoken language Text-to-Pose with different codebook vocabulary sizes on the RWTH-PHOENIX-Weather-2014**T** dataset.

PHOENIX14**T**	TEST SET						DEV SET					
Vocabulary Size:	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE
500	**0.08563**	24.24	13.11	9.31	7.41	25.064	**0.08548**	23.71	12.68	8.79	6.84	24.19
1,000	0.09401	24.78	13.96	9.85	7.71	25.90	0.09336	24.37	13.31	9.08	6.97	25.34
3,000	0.1029	25.70	14.48	10.179	7.90	26.68	0.1039	25.08	13.72	9.53	7.316	26.27
4,000	0.1047	**27.74**	**16.36**	**11.75**	**9.20**	**27.93**	0.1036	**27.85**	**16.71**	**12.19**	**9.64**	**28.87**
5,000	0.1036	24.71	13.39	9.39	7.22	26.03	0.1029	24.99	13.47	9.20	6.955	25.74
6,000	0.1044	22.81	12.35	8.93	7.16	24.33	0.1045	25.37	14.65	10.66	8.36	27.14

TABLE II

The results of translating from spoken language Text-to-Pose with different codebook vocabulary sizes on the Meine DGS Annotated (mDGS) dataset.

mDGS	TEST SET						DEV SET					
Vocabulary Size:	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE
500	0.1090	14.94	3.05	0.57	0.15	**21.54**	0.1078	14.80	3.01	0.632	0.21	**21.14**
1,000	**0.1071**	14.72	3.04	0.73	0.27	21.25	**0.1069**	14.84	2.83	0.59	0.18	21.01
2,500	0.1153	**15.83**	2.99	0.77	0.19	21.21	0.1157	**15.71**	2.80	0.72	0.00	20.82
3,000	0.1251	15.08	2.84	0.70	0.21	20.66	0.1245	15.25	2.78	0.71	0.23	20.54
3,500	0.1241	15.45	**3.06**	**0.79**	**0.29**	21.41	0.1227	15.43	**3.01**	0.72	0.21	20.91
5,000	0.1288	14.95	2.63	0.68	0.24	21.12	0.1306	14.87	2.74	**0.86**	0.38	20.99
10,000	0.1262	7.55	1.37	0.36	0.00	20.71	0.1271	7.58	1.29	0.31	**0.098**	20.58
25,000	0.1203	10.23	1.75	0.34	0.00	21.49	0.1193	10.19	1.72	0.41	0.14	21.13

V. EXPERIMENTS

A. *Quantitative Evaluation*

In this section, we provide a quantitative evaluation of our SLP approach. Initially, we search for the optimal vocabulary size and window size for our codebooks, showing the result is dataset dependent. Following this, we conduct an ablation study on the mDGS dataset, demonstrating the advantages of both the contrastive loss and the stitching model. We then apply an enhanced codebook trained on the mDGS to the PHOENIX14**T** dataset. Finally, to facilitate a meaningful comparison on the PHOENIX14**T**, we train a G2P model and assess its performance against prior works.

1) Codebook Vocabulary Size: Our first experiment searches for the best vocabulary size for each dataset. We fixed the window size to 4 frames and trained each codebook till convergence. After we freeze the codebook and use it to tokenize each dataset for translation.

As shown in Table I the best vocabulary size is found to be 4,000 on PHOENIX14**T**, achieving an impressive 27.85 BLEU-1 and 28.87 ROUGE score. The optimum is roughly 4 times larger than the original gloss vocabulary, suggesting the model is distinguishing between lexical variations. Whereas, on the larger dataset, mDGS, a smaller vocabulary is found to be optimal at 2,500 as shown in Table II. Suggesting the model is finding a subunit representation given the original gloss vocabulary was approximately 4,500. At the optimal vocabulary, a reasonable score of 15.83 BlEU-1 and 21.21 ROUGE were achieved. However, the model showed limited performance on BLEU-2 to 4 metrics. This is due to the limits of the back translation model, as on the ground truth data, the model achieved only 0.8 BLEU-4 (shown in row 1 Table V, GT). Relative to this theoretical maximum our model performs well.

The smaller the codebook size the more data points map to a single token, as a result, tokens can suffer from regression to the mean, resulting in under-articulated signing. Hence, on both datasets, the lowest DTW-MJE is achieved at small codebook sizes as each token is more likely to contain a mean pose. Therefore we choose to follow BLEU and ROUGE scores when deciding the best vocabulary size.

On PHOENIX14**T** we find our best codebook used 3985 tokens to tokenize the training data, a 99.6% vocabulary usage. mDGS has a similar result, with 99.8% of tokens being used. The aggressive codebook replacement strategy effectively removes dead tokens, enabling the high codebook utility.

2) Codebook Window Size: Next, we investigate the best window size for each codebook entry. The vocabulary is fixed to the optimum found in the previous experiment, 4,000 on PHOENIX14**T** and 2,500 on mDGS. On PHOENIX14**T** we find a BLEU and ROUGE of 27.85 and 28.87 respectively, as shown in Table III. On mDGS a window size of 8 frames was found to be the optimum (Table IV). However, both correspond to 0.48 seconds of signing, since PHOENIX14**T** and mDGS were captured at 25 and 50 fps, respectively.

Examination of both Table IV and Table III reveals that increasing the window size beyond 4 and 8, respectively, decreased the BLEU and ROUGE scores while improving the DTW-MJE. The minimum DTW-MJE was observed at a window size of 24. Possibly due to the reduced number of tokens in the target sequence that led to fewer discontinuities in the pose sequence.

3) Ablation Study: We start by sharing the results of training the back translation model (GT Table V). The model achieves good BLEU-1 and ROUGE scores. However, the

TABLE III

THE RESULTS OF TRANSLATING FROM SPOKEN LANGUAGE TEXT-TO-POSE WITH DIFFERENT CODEBOOK WINDOW SIZES ON THE RWTH-PHOENIX-WEATHER-2014**T** DATASET.

PHOENIX14**T**	TEST SET						DEV SET					
Window Size:	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE
2	0.0975	23.85	12.26	8.26	6.38	24.15	0.1004	23.03	11.85	7.88	5.89	24.08
4	0.1047	**27.74**	**16.36**	**11.75**	**9.20**	**27.93**	0.1036	**27.85**	**16.71**	**12.19**	**9.64**	**28.87**
8	0.0983	23.37	12.67	8.97	7.05	24.77	0.9953	23.32	12.91	9.34	7.58	25.34
12	0.1000	21.12	11.27	8.16	6.58	23.49	0.0997	21.48	11.40	8.02	6.30	23.74
16	0.0924	19.17	9.59	6.63	5.21	21.37	0.0917	19.43	9.83	6.85	5.36	22.17
24	**0.0893**	19.07	9.77	7.00	5.63	22.33	**0.0899**	17.94	9.20	6.42	5.07	21.67
32	0.0914	19.03	9.72	7.32	5.89	21.74	0.8925	17.82	8.80	5.83	4.48	20.93

TABLE IV

THE RESULTS OF TRANSLATING FROM SPOKEN LANGUAGE TEXT-TO-POSE WITH DIFFERENT CODEBOOK WINDOW SIZES ON THE MEINE DGS ANNOTATED (MDGS) DATASET.

mDGS	TEST SET						DEV SET					
Window Size:	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE
2	0.1321	14.59	2.49	0.54	0.00	21.05	0.1326	14.38	2.37	0.51	0.14	20.70
4	0.1277	15.45	**3.12**	0.70	0.00	**21.38**	0.1269	15.46	**2.90**	0.62	**0.25**	**20.95**
8	0.1153	**15.83**	2.99	**0.77**	**0.29**	21.21	0.1157	**15.71**	2.80	**0.71**	0.00	20.82
12	0.1147	14.75	2.72	0.57	0.21	20.82	0.1142	15.05	2.79	0.63	0.00	20.93
16	0.1120	14.22	2.72	0.74	0.26	19.79	0.1115	14.31	2.54	0.63	0.23	19.73
24	**0.1113**	12.05	1.89	0.44	0.13	17.95	**0.1092**	12.01	1.76	0.33	0.00	18.05
32	0.1124	11.35	1.20	0.32	0.10	16.51	0.1118	11.34	1.80	0.35	0.00	16.62

TABLE V

THE RESULTS OF TRANSLATING FROM TEXT-TO-POSE WITH DIFFERENT APPROACHES ON THE MEINE DGS ANNOTATED (MDGS) DATASET.

mDGS	TEST SET						DEV SET					
Approach:	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE
GT	0.000	20.87	5.60	1.89	0.80	23.78	0.000	20.75	5.43	1.81	0.76	23.41
Quantization	0.0612	17.88	3.59	0.76	0.22	20.86	0.0611	17.97	3.35	0.81	0.25	20.73
PT	0.2291	6.11	0.94	0.21	0.05	8.36	0.2284	6.22	0.98	0.17	0.00	8.44
PT + GN	0.2245	7.18	1.48	0.40	0.01	8.38	0.2241	9.22	1.63	0.38	0.01	8.57
Codebook	0.1153	15.83	2.99	0.77	0.19	21.21	0.1157	15.71	2.80	0.72	0.00	20.82
Codebook + Stitching	0.1135	16.32	3.12	0.80	0.00	20.96	0.1137	16.11	3.02	0.79	0.19	20.72
Codebook + Contrastive	0.0882	17.44	3.66	0.99	0.35	**22.25**	0.0879	17.45	**3.78**	**1.20**	**0.46**	21.98
Codebook + Contrastive + Stitching	**0.0865**	**17.62**	**3.72**	**1.04**	**0.39**	22.23	**0.0866**	**17.53**	3.76	1.15	0.43	**22.15**

performance was limited on BLEU-2 to 4. The mDGS dataset is challenging with a spoken language lexicon of 29,275 (10 times that of PHOENIX14**T**) which limits the back translation results. As such, row 1, GT, should be considered the upper bound for all experiments on mDGS.

Tokenizing a pose sequence with a codebook causes two quantization errors. Firstly, tailing frames are lost if the sequence is not a multiple of the window size, and, secondly, an error in the pose is caused by selecting the closest codebook token. As shown in Table V the accumulation of these errors reduced the performance from the GT (row 1) to Quantized (row 2), a decrease in BLEU-1 to 4 of 2.78, 2.08, 1.00 and 0.51 on the Dev set, a relatively small decrease in performance.

For comparison, the following two rows (PT and PT + GN) of Table V show the results of training a Progressive Transformer on the same pose data with the same normalisation. Adding Gaussian noise (GN) to the input at a rate of 5 increased the BLEU-1 and ROUGE scores by 3.00 and 0.13, respectively on the Dev set (shown in row 4, PT + GN). Despite this augmentation, our baseline approach was shown to outperform both versions of the progressive transformer. Showing an impressive BLEU-1 increase of 8.65 and 6.49 on the Test and Dev set.

Now we present two additional techniques to improve the performance of our baseline model. 1) A supervised contrastive learning technique, described in Section III-C and, 2) a codebook stitching approach, described in III-F.

When training the codebook we apply an additional loss to the encoder of the model, this encourages sequences from the same gloss to have a similar embedding, while simultaneously pushing them away from sequences with a different ID. We believe this helps the model to ignore the natural variation between signers, allowing it to focus on the core similarities. As shown in Table V ”Codebook + Contrastive” (row 6) the incorporation of the loss improved the performance on all metrics. The DTW-MJE improved by 23% and 24% on the test and dev set, respectively, while the BLEU-1 score increased by 1.61 and 1.74. Showing that linguistic annotation can be used to further improve the approach.

The stitching module is applied to the predicted sequence to create smoother transitions between codebook tokens, as described in Section III-F. Table V ”Codebook + Stitching”

TABLE VI

THE RESULTS OF TRANSLATING FROM SPOKEN LANGUAGE TEXT-TO-POSE ON THE RWTH-PHOENIX-WEATHER-2014**T** WITH A CODEBOOKS TRAINED ON DIFFERENT DATASET.

PHOENIX14**T**	TEST SET			DEV SET		
Approach:	DTW-MJE	BLEU-1	ROUGE	DTW-MJE	BLEU-1	ROUGE
PHIX Codebook	0.1014	24.68	23.89	0.0997	24.40	24.21
mDGS Codebook	0.1108	22.51	23.38	0.1106	22.71	23.92
mDGS Codebook+	0.1064	24.32	24.03	0.1046	24.03	23.31

TABLE VII

THE RESULTS OF TRANSLATING FROM GLOSS-TO-POSE ON THE RWTH-PHOENIX-WEATHER-2014**T** DATASET.

PHOENIX14**T** Approach:	DTW-MJE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE
GT	0.000	32.41	20.19	14.41	11.32	32.96
PT [38]	0.191	11.45	7.08	5.08	4.04	-
NAT-AT [15]	0.177	14.26	9.93	7.11	5.53	-
NAT-EA [15]	0.146	15.12	10.45	7.99	6.66	-
PoseVQ-MP [50]	0.146	15.43	10.69	8.26	6.98	-
PoseVQ_Diffusion [50]	0.116	16.11	11.37	9.22	7.50	-
G2P Ours	**0.098**	**25.46**	**14.40**	**10.33**	**8.17**	**26.898**
T2P Ours	**0.105**	**27.74**	**16.36**	**11.75**	**9.20**	**27.93**

shows the stitching module increased the back translation BlEU-1 score by 0.49, while also improving the mean joint error. This also had qualitative improvements reducing the number of discontinuities in the predicted sequences, and as a result, the sequence was more realistic. To evaluate this experimentally we average the velocity of the signer's skeleton and find that the original data has a standard deviation of 0.044. In contrast, the quantized sequence has a deviation of 0.056. By applying the stitching module to the output the standard deviation moves closer to the original data to 0.039.

4) Cross-Corpus Codebook Study: Here we investigate if a codebook trained on a high-resource dataset can be applied to another. We take a codebook trained on the mDGS and use it to perform translation on the PHOENIX14**T** dataset. As a baseline, we train a codebook with the same hyperparameters (Table VI row 1, PHIX Codebook). We use two codebooks, a normal model (mDGS Codebook) and an enhanced codebook with stitching plus contrastive learning (mDGS Codebook+).

The results show codebooks can be shared across datasets, although with some reduction in performance, compared to training on the original data. Between Table VI rows 1 and 2, we see a small decrease in BLEU-1 and ROUGE of 1.69 and 0.29, on the Dev set respectively. However, applying the enhanced codebook to the PHOENIX14**T** dataset recovered the majority of the lost performance.

5) State-of-the-art Comparison: Finally, to compare against previous works we take our best-performing parameters found in Section V-A.1 and V-A.2 and apply them to the G2P task. Results for comparison are provided by [50]. We find our approach outperforms all previous methods on all metrics, including the progressive transformer [38], a non-autoregressive transformer [15] and a diffusion-based approach [13]. Compared to the next best model we achieve improvements of 10.5% and 72.2% in DTW-MJE and BLEU-1.

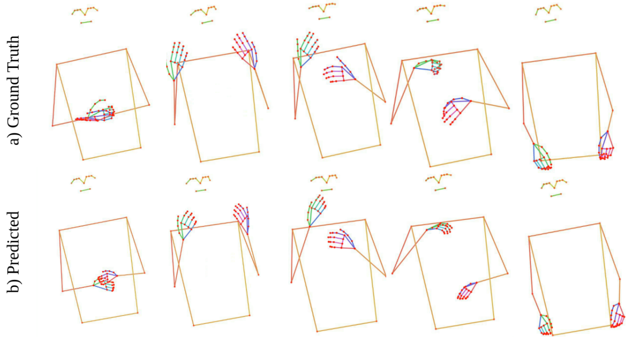
Surprisingly, we find the best performance when translating from text. This could be because of two reasons, 1) gloss is not a perfect representation of sign language and lacks many essential channels (mouthing, body posture and facial expression) increasing the difficulty in this context. 2) the additional context within the spoken language assists the model. Overall we find the BLEU-1 and 4 scores increase by 2.28 and 1.42 when translating from text. Our approach circumvents the need for expensive gloss annotation, paving the way for better communication with the deaf community.

B. Qualitative Evaluation

Fig. 3 shows a translation example from the PHOENIX14**T** dataset. The figure shows the model is able to faithfully translate a given sentence. However, note that some details are missing in the hands caused by the quantization error from the codebook.

In addition, we share video outputs from our best models on the mDGS and PHOENIX14**T** datasets[3]. To ensure a realistic evaluation we also share failure cases. Finally, we share PCA plots of the codebook's embedding space, showing the effect of our replacement strategy and contrastive loss.

Fig. 3. A Translation example produced by our best model on the RWTH-PHOENIX-Weather-2014**T** dataset.



Text: gewitterschauer örtlich mit hagel starkregen und sturmböen ziehen in der nacht ostwärts im norden regnet es länger

VI. CONCLUSION

In this work, we presented a novel approach to T2P translation. Previously the task was treated as a pose regression problem, where the goal was to synthesize a pose sequence directly from text. As a result, the resulting poses suffered from regression to the mean. Here we propose performing a discrete sequence-to-sequence translation using a transformer. To accomplish this we create a discrete representation of sign language, in which the tokens can be combined to create continuous natural expressive signing. We explored the application of sign stitching to generate seamless, more natural sequences. Furthermore, we showed how linguistic annotation can be leveraged to improve our approach. In cases where linguistic annotation is absent, we demonstrated the feasibility of sharing codebooks across datasets. We evaluate our approach on the PHOENIX14**T** and mDGS dataset, showing state-of-the-art back translation and DTW-MJE scores.

[3] `https://github.com/walsharry/VQ_SLP_Demos`

References

[1] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[2] Y. Bouzid and M. Jemni. An avatar based approach for automatically interpreting a sign language notation. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*, pages 92–94. IEEE, 2013.

[3] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudreault, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, et al. Sign language recognition, generation, and translation: An interdisciplinary perspective. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, pages 16–31, 2019.

[4] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. Neural sign language translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[5] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10023–10033, 2020.

[6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[7] S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *Advances in neural information processing systems*, 31, 2018.

[8] P. Dierckx. Algorithms for smoothing data with periodic and parametric splines. *Computer Graphics and Image Processing*, 20(2):171–184, 1982.

[9] A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[10] J. R. Glauert, R. Elliott, S. J. Cox, J. Tryggvason, and M. Sheard. Vanessa–a system for communication between deaf and hearing people. *Technology and disability*, 18(4):207–216, 2006.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

[12] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[13] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706, 2022.

[14] T. Hanke. Hamnosys-representing sign language data in language resources and language processing contexts. In *LREC*, volume 4, pages 1–6, 2004.

[15] W. Huang, W. Pan, Z. Zhao, and Q. Tian. Towards fast and high-quality sign language production. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3172–3181, 2021.

[16] E. J. Hwang, H. Lee, and J. C. Park. Autoregressive sign language production: A gloss-free approach with discrete representations. *arXiv preprint arXiv:2309.12179*, 2023.

[17] M. Ivashechkin, O. Mendez, and R. Bowden. Improving 3d pose estimation for sign language. In *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 1–5, 2023.

[18] Z. Jiang, A. Moryossef, M. Müller, and S. Ebling. Machine translation between spoken languages and signed languages represented in signwriting. *arXiv preprint arXiv:2210.05404*, 2022.

[19] L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2390–2399. PMLR, 2018.

[20] 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.

[21] K. Kaur and P. Kumar. Hamnosys to sigml conversion system for sign language automation. *Procedia Computer Science*, 89:794–803, 2016.

[22] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[24] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] P. Koehn and R. Knowles. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*, 2017.

[26] O. Koller, J. Forster, and H. Ney. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding*, 141:108–125, 2015. Pose & Gesture.

[27] R. Konrad, T. Hanke, G. Langer, D. Blanck, J. Bleicken, I. Hofmann, O. Jeziorski, L. König, S. König, R. Nishio, A. Regen, U. Salden, S. Wagner, S. Worseck, O. Böse, E. Jahn, and M. Schulder. Meine dgs – annotiert. öffentliches korpus der deutschen gebärdensprache, 3. release / my dgs – annotated. public corpus of german sign language, 3rd release, 2020.

[28] J. Kreutzer, J. Bastings, and S. Riezler. Joey nmt: A minimalist nmt toolkit for novices. *arXiv:1907.12484*, 2019.

[29] J. Laver. Linguistic phonetics. *The handbook of linguistics*, pages 150–179, 2001.

[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[31] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[32] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019.

[33] J. McDonald, R. Wolfe, J. Schnepp, J. Hochgesang, D. G. Jamrozik, M. Stumbo, L. Berke, M. Bialek, and F. Thomas. An automated technique for real-time production of lifelike animations of american sign language. *Universal Access in the Information Society*, 15:551–566, 2016.

[34] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[35] K. Papineni, S. Roukos, T. Ward, and W.-J. 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, 2002.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[37] B. Saunders, N. C. Camgoz, and R. Bowden. Adversarial training for multi-channel sign language production. *arXiv preprint arXiv:2008.12405*, 2020.

[38] B. Saunders, N. C. Camgoz, and R. Bowden. Progressive transformers for end-to-end sign language production. In *European Conference on Computer Vision*, 2020.

[39] B. Saunders, N. C. Camgoz, and R. Bowden. Mixed signals: Sign language production via a mixture of motion primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1919–1929, 2021.

[40] B. Saunders, N. C. Camgoz, and R. Bowden. Signing at Scale: Learning to Co-Articulate Signs for Large-Scale Photo-Realistic Sign Language Production. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

[42] W. C. Stokoe. Sign language structure. *Annual Review of Anthropology*, 1980.

[43] S. Stoll, N. C. Camgoz, S. Hadfield, and R. Bowden. Text2sign: towards sign language production using neural machine translation and generative adversarial networks. *International Journal of Computer Vision*, 128(4):891–908, 2020.

[44] V. Sutton. *Lessons in SignWriting*. SignWriting Press, 2022.

[45] R. Sutton-Spence and B. Woll. *The linguistics of British Sign Language: an introduction*. Cambridge University Press, 1999.

[46] S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 1988.

[47] M. H. Vali and T. Bäckström. Nsvq: Noise substitution in vector quantization for machine learning. *IEEE Access*, 10:13598–13610, 2022.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[49] H. Walsh, B. Saunders, and R. Bowden. Changing the representation: Examining language representation for neural sign language production. *arXiv preprint arXiv:2210.06312*, 2022.

[50] P. Xie, Q. Zhang, Z. Li, H. Tang, Y. Du, and X. Hu. Vector quantized diffusion model with codeunet for text-to-sign pose sequences generation. *arXiv preprint arXiv:2208.09141*, 2022.

[51] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.