

Moscow Institute of Physics and Technology Phystech School of
Applied Mathematics and Informatics
Department of Machine Learning and Digital Humanities

Alua Musralina

Sign Language Production using the Deep Learning

Master's Thesis

Advisor: Radoslav Neychev

Moscow, 2023

Contents

1	Introduction	3
2	Literature overview	5
3	Dataset	12
4	Metrics	14
5	Problem statement	16
6	Computational experiment	17
7	Conclusion	20
	Bibliography	21

Abstract

Sign language is a visually performed language with distinct signed languages and millions of deaf individuals worldwide. Sign Language Processing involves tasks such as translation and production. This work focuses on sign language production, crucial for interpreters, educators, and accessibility. Sign sequences and spoken language sentences lack direct alignment. Therefore, Gloss production as an intermediate step is crucial. Deep learning approaches, like Neural Machine Translation are applied and different seq2seq architectures application results in this work are evaluated. Fine-tuning pre-trained models is explored as well. With fine-tuned transformer the results are improved from the previous methods and can be a promising step towards to close to natural Sign Language Production.

1 Introduction

According to the United Nations in year 2022, there are as many as 300 distinct signed languages, while the World Federation of the Deaf estimates that there may be up to 70 million deaf individuals worldwide.

Sign language is a visually performed language that relies on hand gestures, body postures, and facial expressions, with the meaning of signs being determined by the combination of all these elements. The Sign Language Processing is the part of both Natural Language Processing (NLP) and Computer Vision (CV) tasks. For it, there are two main tasks can be implemented: Translation from sign language (Sign Language Recognition) and into sign language (Sign Language Production).

In this work the focus will be on the second part. Skilled sign language production is essential for professional sign language interpreters and translators. Interpreters work in a wide range of settings, including conferences, meetings, classrooms, medical appointments, legal proceedings, and public events. Educators, sign language instructors, and teachers of deaf or hard-of-hearing students use sign language production to deliver lessons, provide instructions, and foster a conducive learning environment. Sign language production is commonly used in public service announcements, news broadcasts, and emergency broadcasts to ensure accessibility for the deaf and hard-of-hearing community. Including sign language alongside spoken language ensures that important information is accessible to everyone. Sign language production is utilized in performance arts, such as sign language poetry, storytelling, and theatre. Deaf performers and hearing performers incorporating sign language can express their creativity, engage audiences, and promote cultural diversity and inclusivity.

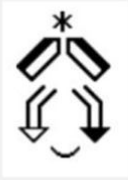



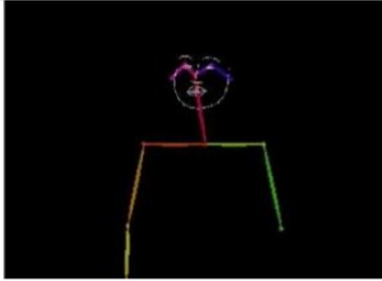


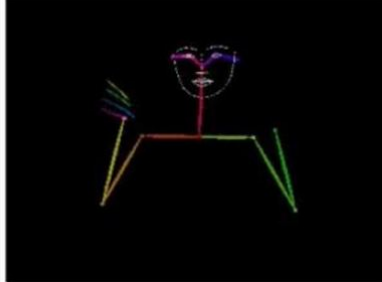

English Translation	Gloss	Notation	Pose Estimation	Video
House	HOUSE			
What's the matter? What's wrong?	Wrong- What			
Different But	DIFFERENT BUT			

Figure 1: English translations, Gloss, Notations of several signs [1].

Each sign in sign language is identified by a unique identifier known as a gloss. There is no direct alignment between sign sequences and spoken language sentences. See example in Fig. 1. Please note that a single sign can have two unrelated glosses, and a single gloss can be translated into multiple valid spoken languages.

In general, sign language processing involves several tasks such as detection, identification, segmentation, recognition, translation, and production. Detection involves identifying whether sign language is present in a video, while identification is about determining which sign language is being used. The segmentation task applied to identify the signs to separate phrases or signs, while recognition aims to comprehend the signs, resulting in labels being assigned to each sign. Translation aims to provide a more natural output than just glosses or labels, and it is an automatic process. Production, on the other hand, involves translat-

ing text into signs [2].

2 Literature overview

The skeleton points based approach in recognition and generation tasks makes it possible to focus on the signs themselves, since the subject and background variation becomes independent. For example, the work [3] uses the whole body estimator to find the skeleton points, where 133 points are whole body keypoints, 27 nodes of graph, 10 nodes for each hand and 7 nodes for upper body. In that work the sign language is also represented as a sequence of continuous skeleton points, which models the coordinates of the person, including the manual and non-manual features. In general, there are not so many researches are done for continuous sign generation.

After producing a sequence of skeleton points, given a source spoken language, author in [4] states, that transforming them into a video requires not much effort. It includes plotting the joints and connecting the relevant bones. These 3D joints can then be utilised to animate an avatar, like in [5] [6] or serve as conditioning for a GAN [7]. While the generated sign pose sequence accurately translates the provided text, it might be performed at a different pace compared to the reference data. This variation in speed is not considered incorrect since signers have their own motion and tempo. However, to facilitate a visual comparison with the reference sequences, a Dynamic Time Warping (DTW) was employed [8] to align the temporal aspects of the generated sign pose sequences. This alignment process does not modify the content of the productions, but rather enhances the temporal coherence for visualisation purposes [4].

There are several DL approaches are already introduced for automatic Sign

Language Production. Neural Machine Translation (NMT) is a popular application of deep learning that utilises encoder-decoder architecture to translate text from one language to another. The encoder-decoder architecture consists of an encoder network that processes the source language sentence and a decoder network that generates the target language sentence. Classical Transformers, which are commonly used in NMT, utilise self-attention mechanisms and multi-headed attention to improve the quality of machine translations by allowing the model to focus on important parts of the input and generate more coherent and accurate output sentences. Encoder-decoder attention maps the source sequence into target and performs the sequence-to-sequence task. For example, in [9] there are 3 modules implemented: the first module pre-processes the input text and breaks into words with their types (noun, verb, particle, etc.) with a help of language model. In the second module, the pre-processed words are converted into sign sequences, where RBMT, EBMT, SMT, HBMT, and NMT can be applied. The last module converts the sign sequences into sign glosses, videos, or animated avatars. This might be considered as a general approach and will be implemented almost the same way in this work as well.

There are 2 types of the SLP, as mentioned in [10]: end-to-end system and the intermediate results production. In the second method the glosses are produced, prior to generate the Signs, which described in [11] and [12]. The last method can be called as a tokenisation of signs. According to [13] applying the second method reduces the performance results, and therefore the transformer-based encoder-decoder architecture was proposed there. But in this work, by refinement the Gloss production process the possible results improvements are evaluated.

The [14] introduces a seq2seq model with Luong attention that translates text to gloss first. Instead of using CNN for feature extraction, the model employs

the probabilities generated by the decoder to solve a Motion Graph (MG) for sign language pose data, which enables the text-to-pose translation. The MG is a computer graphics technique used for character animation and conditional image generation. To build the MG for sign language pose data, the authors divided continuous sign sequences into individual glosses and grouped the motion sequences by gloss. The nodes of the MG are populated by these motion sequences, and the probabilities from the NMT decoder are used to transition between nodes at each time step.

To identify and monitor hand movements, one can utilize HandDetector libraries. These libraries are part of the CVzone computer vision package, which simplifies the execution of image processing and AI functions. CVzone relies on OpenCV and Mediapipe libraries to perform its tasks. When using the HandDetector libraries, the hands are detected and tracked, and each hand is represented by a list of 21 landmarks. Each landmark is composed of x, y, and z values. x and y are the image’s width and height respectively. z represents the landmark depth with the depth at the wrist being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.

Contrary to [14] in [4] the authors do not provide the a priori information for conditional image generation. It is focused on direct or automatic signs generation, and the method learns the mapping automatically between the text and the pose sequences.

A previous approach was introduced in which the 7-frame sign was generated for every input word. However, this technique had a fixed output length and did not allow dynamic length output [15].

In case of using traditional DNNs, it would be required to be converted the input text into a fixed-length vector using methods, for example as BOW or Word2Vec. However, it should be noted that in doing so, the original word sequence is lost. This results in the model being unaware of the word order, and consequently, important information about the input is missing.

Recurrent Neural Networks (RNNs) were developed to address the issue of retaining sequence information in neural networks. RNNs are specifically designed to process sequential data, such as text, by maintaining a hidden state that captures information about the preceding inputs in the sequence. This enables the model to preserve the order of words in the input text and utilise it for improved predictions or classifications.

The encoder receives each token in the input sequence and attempts to capture all the relevant information about it, consolidating it into a fixed-length vector known as the "context vector". Once all tokens have been processed, the encoder passes this vector to the decoder. The context vector is designed to encapsulate the entirety of the input sequence's meaning, allowing the decoder to make precise predictions. The decoder uses the context vector to predict each token in the target sequence, one at a time. The encoder component with an LSTM cell processes the input sequence over time, attempting to capture and store all relevant information in its final internal states, namely the hidden state h_t and cell state c_t . The decoder block is likewise constructed using an LSTM cell. The decoder works by generating its output at each time step t , with the goal of producing the t -th word in the target sequence. Firstly, the decoder receives a special symbol $\langle \text{START} \rangle$ as input, which is used to indicate the beginning of the output sequence. Using this input and the internal states (h_t, c_t) , the decoder produces the output, which is expected to be the first word/token in the target sequence. Then the output of the second step is intended to be the second word in the

target sequence. This process continues, with the output of each step being fed as input to the subsequent step, until the special $\langle \text{END} \rangle$ symbol, which signifies the end of the output sequence, is produced. The decoder's final internal states are disregarded.

The encoder takes in each token/word of the input sequence sequentially and sends the final states to the decoder. Its parameters are updated using back-propagation through time. During the training phase, the decoder operates differently from the encoder. It uses the "Teacher Forcing" technique, so that the actual output/token from the previous step is fed as input to the current step, rather than the predicted output/token. It is important to note that, in the decoder, the output $y_{t_{pred}}$ at any step t represents the probability distribution across the entire output dataset's vocabulary. This distribution is generated using the Softmax activation function. The predicted word is chosen as the token with the highest probability. After generating predictions at each time-step, the loss is computed, and the errors are backpropagated through time to adjust the model parameters during training. The loss function employed is the categorical cross-entropy loss function, which measures the discrepancy between the predicted sequence Y_{pred} and the actual sequence Y_{true} . In testing phase it is a little different, since in a real-world scenario, there will be no access to the actual output sequence Y_{true} , and instead, there will be only the input sequence X . Consequently, it is not possible to employ the same strategy used during the training phase. During testing, the predicted output is fed from the previous time-step as input to the current time-step, rather than the actual output. The remainder of the process is similar to the training phase. To reduce the dimensionality of the input word vectors, the input sequence is passed through an embedding layer in both the encoder and the decoder components. One-hot-encoded vectors can be prohibitively large in practice, making embedded vectors a superior word representation. The embedding layer can be trained together with the model, or it can be pre-trained,

such as in the case of Word2Vec embeddings [17].

Before the transformers the RNN was the base of the Neural SL translation [18]. Both GRU and LSTM are recurrent neural network variants used in seq2seq architectures. LSTM is generally considered more powerful in capturing long-term dependencies, but GRU can be computationally more efficient. In Gloss production since the input and output are texts, the pre-trained model could be applied as well. Those different architectures and models are applied for evaluation in this work.

A Transformer was firstly introduced in [16]. In general, it is a model architecture that avoids using recurrence and instead relies solely on an attention mechanism to capture global dependencies between input and output. This design allows for a greater degree of parallelization and has led to significant advancements in natural language processing and other areas of machine learning. The queries in the "encoder-decoder attention" layers come from the preceding decoder layer, while the memory keys and values are sourced from the encoder's output. Within the encoder and decoder are self-attention layers, where keys, values, and queries all originate from the same source, namely the previous encoder or decoder layer's outputs, correspondingly. Each layer within encoder and decoder not only contains attention sub-layers, but also a fully connected feed-forward network. This network operates on each position individually and uniformly and is composed of two linear transformations separated by a ReLU activation. Although the linear transformations are constant across various positions, they are configured with different parameters from layer to layer. Alternatively, these transformations can be thought of as two convolutions with a kernel size of 1. To generate predicted probabilities for the next token, a learned linear transformation and softmax function to the decoder output is applied.

The goal of the work in [4] was to determine the conditional probability $p(Y/X)$ for generating a sign sequence with U frames, given a text X . However, there were some challenges due to the non-monotonic relationship between the source and target language, as well as the fact that sign language utilises a continuous vector space and different channels for manual and non-manual features. To address these challenges, Progressive Transformers were introduced, which made it possible to translate from a symbolic sequence to a continuous sequence. The encoder in this approach encodes the source sequence to a latent representation before being mapped to a target. Source embedding and the temporal encoding is applied to the symbolic source tokens, when Counting Decoding is applied to the target. As mentioned before, Progressive transformer is for continuous sequences, where there is no pre-defined vocabulary. Here important to mention, that it is achieved using the counter decoding, which predicts continuous sequences of different length by predicting the end of sequence. To deal with the possible prediction drifts, data augmentation can be applied. The transformer is called Progressive, since it checks the progress of sequence generation and knows the length of sequence given a source text.

Initially, the source sentences are tokenised and transformed into embeddings to place tokens with similar meanings closer in a high-dimensional space. The purpose of embeddings is to create a dense vector representation for each word that captures its semantic and syntactic properties. Following this, positional encoding is applied to the word embeddings to account for the order of words in the sequence. Since the embedding layer does not consider the word order, positional encoding provides information about the position of each word in the sequence. This allows the model to learn the relative positions of words and their importance in the sequence. The target is also a sequence and they are continuous signs, which are represented by coordinates. Here the counter encoding is applied as well, which includes information about the frame position relative to

Length distribution in Train data
Average length of source sentences: 20.69
Average length of target sentences: 13.71

Length distribution in Test data
Average length of source sentences: 20.81
Average length of target sentences: 14.02

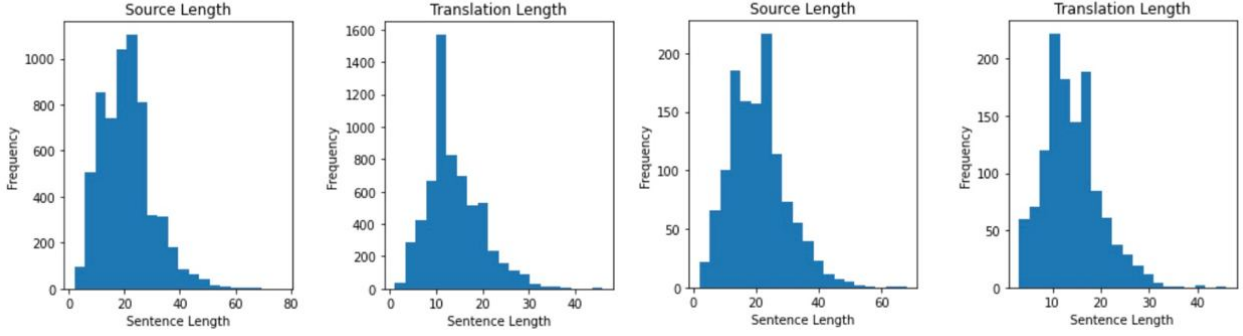


Figure 2: Phoenix Dataset Distribution for SLP.

the total sequence length. Progressive encoder and decoder have several layers, each of which consist of 2 sub-layers: Multi-head attention (MHA) and non-linear feed-forward layer, whereas the progressive decoder generates a sign frame at each time, which is represented by the joined embeddings including the counter. MHA sub-layer is applied in both sides, but in decoder additionally with some extra masking to prevent the future steps decoding [4].

Those seq2seq architectures with RNN and attentions for text2gloss production are implemented and evaluated in this work, therefore described in details.

3 Dataset

The most of the datasets are done for SLR task, and only some of them for SLT [2]. Phoenix German weather dataset is dedicated for SLT task and will be considered for our purpose as well. This dataset contains the pre-recorded 8257 videos of 9 different signers, with 2887 German words and 1066 sign glosses. The distribution of the dataset see in Fig. 2. For production purposes, a annotated dataset is required, as Swiss German dataset, which is prepared for continuous Sign Language Generation. It utilizes linguistic corpora with subunit annotation to acquire cross-language subunit models. These subunit models can be employed

in machine learning applications on gloss-based corpora. Specifically, the research focused on employing deep convolutional neural networks to classify finger and palm orientation [20]. This dataset was requested from author, but not provided. Therefore, in this work only the roughly annotated dataset is prepared and applied for production.

4 Metrics

For tasks like machine translation or summarization, various metrics have been proposed, including BLEU, ROUGE, and METEOR. These metrics provide a way to evaluate the quality of generated text, which can be difficult due to the subjective nature of language.

In general, BLEU provides a standardized and objective way to evaluate the quality of machine translations by comparing them to reference translations. BLEU, which stands for Bilingual Evaluation Understudy, is the most widely used metric for evaluating machine translation systems. It utilizes modified n-gram precision and best match length to approximate precision and recall.

To calculate modified n-gram precision, the fraction of n-grams in the candidate text that are present in any of the reference texts is determined. However, using only this value can lead to issues. For example, if two candidates have identical unigrams but different word orders, they will receive the same score. To address this, modified n-gram precision weights each n-gram based on its maximum count in any single reference text.

Additionally, BLEU incorporates a best match length measure to approximate recall. This is the length of the reference sentence which has the closest length to the candidate sentence. The closer the candidate length is to the best match length, the better the recall score will be.

ROUGE, which stands for Recall-Oriented Understudy for Gisting Evaluation, is a metric commonly used for evaluating the quality of text summarization. It provides a quantitative measure of the effectiveness of a summarization system and helps researchers to compare different models or algorithms.

ROUGE is primarily focused on recall and uses different features to calculate it. There are several types of ROUGE, including ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. ROUGE-N calculates recall based on n-grams, where "N" represents the number of consecutive words or characters in a sequence. For instance, ROUGE-1 counts recall based on matching unigrams, and ROUGE-2 counts recall based on matching bigrams. For any value of N, the metric calculates the total number of n-grams across all reference summaries and measures how many of them are present in the candidate summary. This fraction represents the ROUGE-N score.

In contrast, ROUGE-L, ROUGE-W, and ROUGE-S use different features such as the longest common subsequence (LCS), weighted LCS, and skip-bigram co-occurrence statistics, respectively, to calculate recall. These metrics use an F-score that combines both precision and recall values. For ROUGE-L, precision and recall values are calculated based on the length of the longest common subsequence between the candidate summary and reference summaries.

METEOR, which stands for Metric for Evaluation of Translation with Explicit ORdering, is a machine translation evaluation metric that claims to have a higher correlation with human judgment compared to other metrics such as BLEU. While BLEU is widely used, it can be problematic for evaluating individual sentences as its scores can be influenced by the length of the entire corpus.

METEOR addresses this problem by modifying the precision and recall computations and replacing them with a weighted F-score based on mapping unigrams and a penalty function for incorrect word order. To calculate the weighted F-score, the metric first finds the largest subset of mappings that can form an alignment between the candidate and reference translations. This is done by considering

exact matches, matches after Porter stemming, and matches based on WordNet synonymy. Once the alignment is found, the precision and recall are calculated based on the number of mapped unigrams (m) and the length of the candidate (c) and reference (r) texts, as m/c and m/r , respectively. The F-score is then computed using the harmonic mean of precision and recall.

In addition to the weighted F-score, METEOR also includes a penalty function for incorrect word order, which takes into account the number of unaligned words that are between aligned words. This penalty function helps to ensure that the metric penalizes translations that are not only incorrect in terms of content but also in terms of word order. In general, METEOR is a useful metric for evaluating machine translation systems, as it takes into account both content and word order to provide a more accurate evaluation [21]. The author in [4] proposes the use of back-translation as a method for evaluating sign language processing (SLP). This approach involves translating the generated sign pose sequences back into spoken language.

In our work BLEU-4 and Rouge-1 is chosen as a metric for evaluation, in order to be able to compare with the previous works results, where the same metrics are used.

5 Problem statement

As mentioned above, there is no direct alignment between sign sequences and spoken language sentences. The direct production of signs using the Progressive Transformers gave not high BLEU-4 = 10.51 as well. Even when the method with generation of Gloss in between is applied. In that case BLEU-4 = 9.68 [4]. Gloss production could be still improved with different seq2seq architectures described in literature overview. This work is dedicated to evaluate those possible solutions.

When the gloss production achieves better results, then the mapping from Gloss to Sign will be dependent mostly on dataset dedicated for sign production.

Moreover, there is a pre-trained model, which is trained for NMT task from German to English translation might be used. To fine-tune such encoder-decoder model, typically a pre-trained model that has been trained on a large dataset or a related task is used. The pre-training phase helps the model learn general representations of the data. However, these pre-trained models may not perform optimally on a specific task or dataset due to domain differences or task-specific requirements. This is checked in this work too.

During fine-tuning, the model's parameters are updated using a smaller task-specific dataset. For SLP this dataset contains examples of German sentences and their Gloss representations. By training the model on this dataset, its parameters are adapted to better fit specifically to the task, thereby improving its performance. During the fine-tuning process, some layers of the model may be frozen to prevent them from being updated. This is especially common for lower-level layers, that capture more general features. By freezing these layers, the pre-trained knowledge is preserved and fine-tuning process is focused on the task-specific layers, which are closer to the output.

6 Computational experiment

First, encoder-decoder architecture with LSTM is implemented as a baseline. Then the model with attention and with bidirectional GRU is evaluated. Note, that the bidirectional GRU does not give the possibility for online simultaneous translation.

As the third method, the huggingface pre-trained transformer with almost 74 mln. paramaters [22] is applied.

Method	Number of parameters	Bleu-4	Rouge-1
Stoll: Enc-Dec with attention	Not available	19.10	54.55
Saunders: Progressive transformer case, text2gloss part	Not available	15.26	48.10
Our: Enc-Dec without attention and with LSTM	2,452,438	7.35	29.15
Our: Encoder-Decoder model with attention and with bidirectional GRU	4,937,430	19	49.69
Our: pre-trained Huggingface transformer	73, 886, 208	23.72	64.92

Figure 3: Methods evaluation results comparison table.

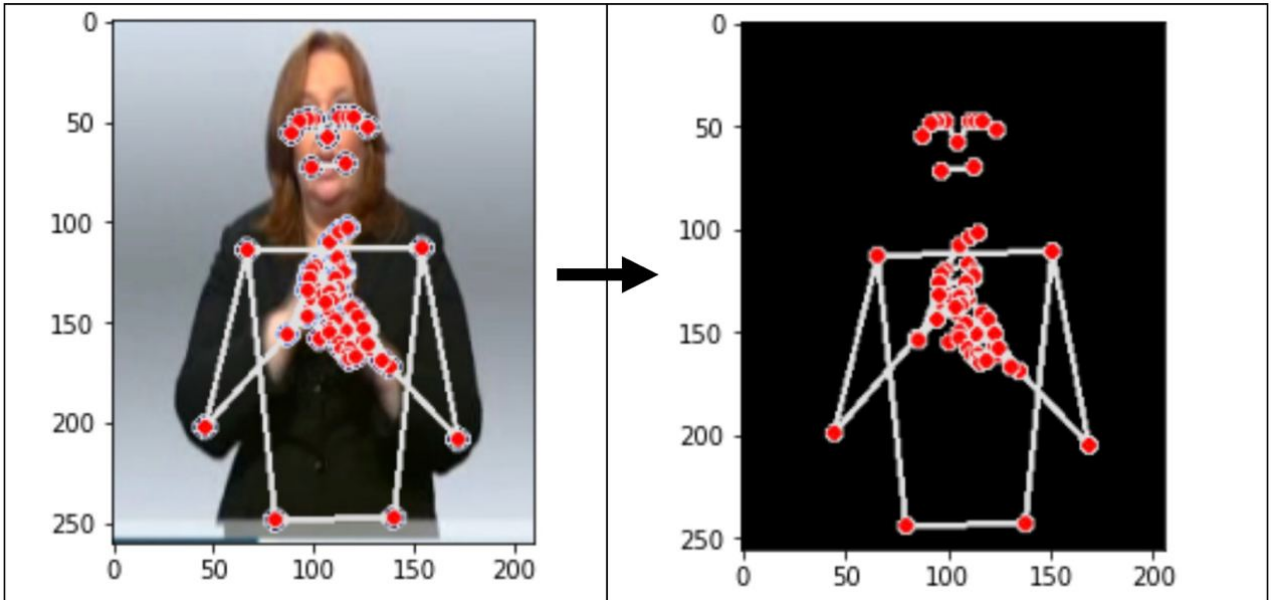


Figure 4: Keypoints generation with mediapipe library example.

After the relatively good results of Gloss production is achieved, the keypoints of the gestures from the glosses are produced. The annotations of the images are done roughly, by calculating the average number of frames for each gloss in sentence and cropping the videos. 2D upper body joint and facial landmark positions are first extracted using MediaPipe library. 21 keypoints for each hand and 33 keypoints for pose are extracted from frames. See the example of keypoints detection in Fig. 4. When the lookup table was ready, the mapping from the generated glosses to keypoints are performed. The results can be shown in videos and visually can be compared with original, otherwise expertise of the sign language interpreter is required.

7 Conclusion

[results so far] Fine-tuning the encoder-decoder model allows you to leverage the pre-trained knowledge while adapting the model to the specific task, leading to improved performance and better results. This was shown in Sign Language Production task, where the source is the German sentences and the target is the Glosses of German Sign Language. With Huggingface pre-trained model, with small dataset (ca. 8000 pairs) a good BLEU 23.72 for generating the Gloss is achieved, which further could be used for avatar generation. The good results shows that, with more data there might be even better results. Additionally, the simpler encoder-decoder models are used, for comparison.

At the end, the keypoints from the Glosses are generated. There are other challenges may occur, as not smooth and not natural movements. This might be the scope of the further research.

References

- [1] <https://research.sign.mt>
- [2] Adrián Núñez-Marcos, Olatz Perez-de-Viñaspre, Gorka Labaka: A survey on Sign Language machine translation. *Expert Systems With Applications*, Spain (March, 2023)
- [3] Songyao Jiang, Bin Sun, Lichen Wang, Yue Bai, Kunpeng Li and Yun Fu: A Skeleton Aware Multi-modal Sign Language Recognition. *CVPR*, USA (May, 2021)
- [4] Ben Saunders, Necati Cihan Camgoz, Richard Bowden: Text2Sign: Continuous 3D Multi-Channel Sign Language Production via Progressive Transformers and Mixture Density Networks. *International Journal of Computer Vision*, (May, 2021)
- [5] Michael Kipp, Alexis Heloir, Quan Nguyen: Sign Language Avatars: Animation and comprehensibility. *International Workshop on Intelligent Virtual Agents (IVA)*, (2011)
- [6] John McDonald: Automated Technique for Real-Time Production of Lifelike Animations of American Sign Language. *Universal Access in the Information Society (UAIS)*, (2016)
- [7] Caroline Chan: Everybody Dance Now. *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, (2019)
- [8] Donald J Berndt, James Clifford: Using Dynamic Time Warping to Find Patterns in Time Series. *AAA1 Workshop on Knowledge Discovery in Databases (KDD)*, (1994)
- [9] Navroz Kaur Kahlon, Williamjeet Singh: Machine translation from text to sign language: a systematic review. *CVPR*, USA (July, 2021)

- [10] Amanda Duarte: Cross-modal Neural Sign Language Translation. The 27th ACM International Conference, (October, 2019)
- [11] Sujay S Kumar, Tenzin Wangyal, Varun Saboo, Ramamoorthy Srinath: Time Series Neural Networks for Real Time Sign Language Translation. The 17th IEEE International Conference on Machine Learning and Applications (ICMLA), (December, 2018)
- [12] Hao Zhou, Wengang Zhou, Weizhen Qi, Junfu Pu, Houqiang Li: Improving Sign Language Translation with Monolingual Data by Sign Back-Translation. CVPR, (May, 2021)
- [13] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, Richard Bowden: Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation. CVPR, (March, 2020)
- [14] Stephanie Stoll, Necati Cihan Camgoz, Simon Hadfield, Richard Bowden: Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks. International Journal of Computer Vision, (January, 2020)
- [15] Zelinka, J., Kanis, J: Neural sign language synthesis: Words are our glosses. Neural sign language synthesis: Words are our glosses. The IEEE winter conference on applications of computer vision (WACV), (May, 2020)
- [16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin: Attention Is All You Need, Computation and Language, (December, 2017)
- [17] Kriz Moses: <https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186fbf49b>

- [18] Biyi Fang, Jillian Co, Mi Zhang: DeepASL: Enabling Ubiquitous and Non-Intrusive Word and Sentence-Level Sign Language Translation. CVPR, (October, 2018)
- [19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Computation and Language, (May, 2019)
- [20] Oscar Koller, Hermann Ney, Richard Bowden: [https://www-i6.informatik.rwth-aachen.de/publications/download/999/KollerOscarNeyHermannBowdenRichardAutomaticAlignmentofHamNoSysSubunitsforContinuousSign LanguageRecognition-2016.aux](https://www-i6.informatik.rwth-aachen.de/publications/download/999/KollerOscarNeyHermannBowdenRichardAutomaticAlignmentofHamNoSysSubunitsforContinuousSignLanguageRecognition-2016.aux)
- [21] Desh Raj: <https://medium.com/explorations-in-language-and-learning/metrics-for-nlg-evaluation-c89b6a781054>
- [22] <https://huggingface.co/mariav/helsinki-opus-de-en-fine-tuned-wmt16/tree/main>