# Changing the Representation: Examining Language Representation for Neural Sign Language Production

**Harry Walsh, Ben Saunders, Richard Bowden**

University of Surrey
{harry.walsh, b.saunders, r.bowden}@surrey.ac.uk

**Abstract**

Neural Sign Language Production (SLP) aims to automatically translate from spoken language sentences to sign language videos. Historically the SLP task has been broken into two steps; Firstly, translating from a spoken language sentence to a gloss sequence and secondly, producing a sign language video given a sequence of glosses. In this paper we apply Natural Language Processing techniques to the first step of the SLP pipeline. We use language models such as BERT and Word2Vec to create better sentence level embeddings, and apply several tokenization techniques, demonstrating how these improve performance on the low resource translation task of Text to Gloss. We introduce Text to HamNoSys (T2H) translation, and show the advantages of using a phonetic representation for sign language translation rather than a sign level gloss representation. Furthermore, we use HamNoSys to extract the hand shape of a sign and use this as additional supervision during training, further increasing the performance on T2H. Assembling best practise, we achieve a BLEU-4 score of 26.99 on the MineDGS dataset and 25.09 on PHOENIX14T, two new state-of-the-art baselines.

**Keywords:** Sign Language Translation (SLT), Natural Language Processing (NLP), Sign Language, Phonetic Representation

## 1. Introduction

Sign languages are the dominant form of communication for Deaf communities, with 430 million users worldwide (WHO, 2021). Sign languages are complex multichannel languages with their own grammatical structure and vocabulary (Stokoe, 1980). For many people, sign language is their primary language, and written forms of spoken language are their secondary languages.

Sign Language Production (SLP) aims to bridge the gap between hearing and Deaf communities, by translating from spoken language sentences to sign language sequences. This problem has historically been broken into two steps; 1) translation from spoken language to gloss[1] and 2) subsequent production of sign language sequences from a sequence of glosses, commonly using a graphical avatar (Elliott et al., 2008; Efthimiou et al., 2010; Efthimiou et al., 2009) or more recently, a photo-realistic signer (Saunders et al., 2021a; Saunders et al., 2021b). In this paper, we improve the SLP pipeline by focusing on the Text to Gloss (T2G) translation task of step 1.

Modern deep learning is heavily dependent upon data. However, the creation of sign language datasets is both time consuming and costly, restricting their size to orders of magnitude smaller than their spoken language counterparts. State-of-the-art datasets such as RWTH-PHOENIX-Weather-2014**T** (PHOENIX14**T**), and the newer MineDGS (mDGS), contain only 8,257 and 63,912 examples respectively (Koller et al., 2015; Hanke et al., 2020), compared to over 15 million examples for common spoken language datasets (Vrandečić and Krötzsch, 2014). Hence, sign languages can be considered as low resource languages.

In this work, we take inspiration from NLP techniques to boost translation performance. We explore how language can be modeled using different tokenizers, more specifically Byte Pair Encoding (BPE), WordPiece, word and character level tokenizers. We show that finding the correct tokenizer for the task helps simplify the translation problem.

Furthermore, to help tackle our low resource language task, we explore using pre-trained language models such as BERT (Devlin et al., 2018) and Word2Vec (Mikolov et al., 2013b) to create improved sentence level embeddings. We also fuse contextual information from the embedding to increase the amount of information available to the network. We show that using models trained on large corpuses of data improves translation performance.

Previously the first step of the SLP pipeline used T2G translation. We explore using a phonetic representation based on the Hamburg Notation System (HamNoSys) which we define as Text to HamNoSys (T2H). HamNoSys encodes signs using a set of symbols and can be viewed as a phonetic representation of sign language (Hanke, 2004). There are three main components when representing a sign in HamNoSys; a) its initial configuration b) it's hand shape and c) it's action. An example of HamNoSys can be seen in Fig. 1 along with its gloss and text counterparts.

We evaluate our SLP models on both the mDGS and PHOENIX14**T** datasets, showing state-of-the-art performance on T2G (mDGS & PHX) and T2H (mDGS)

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

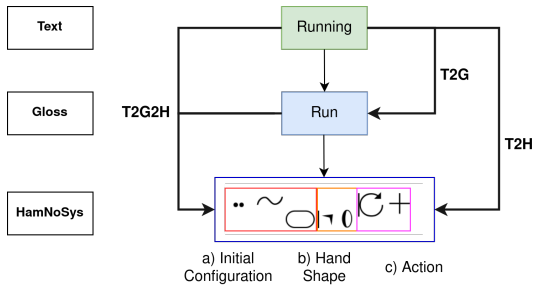


Figure 1: A graph to show the word "running" which would be 'glossed' as RUN and the associated sequence of HamNoSys, Top: Text, Middle: Gloss, Bottom: HamNoSys. HamNoSys is split into: a) it's initial configuration b) it's hand shape 3) it's action

tasks. We achieve a BLEU-4 score of 26.99 on mDGS, a significant increase compared to the state-of-the-art score of 3.17 (Saunders et al., 2022).

The rest of this paper is structured as follows; In section 2 we review the related work in the field. Section 3 presents our methodology. Section 4 shows quantitative and qualitative results. Finally, we draw conclusions in section 5 and suggest future work.

## 2. Related Work

**Sign Language Recognition & Translation:** Computational sign language research has been studied for over 30 years (Tamura and Kawasaki, 1988). Research started with isolated Sign Language Recognition (SLR) where individual signs were classified using CNNs (Lecun et al., 1998). Recently, the field has moved to the more challenging problem of Continuous Sign Language Recognition (CSLR), where a continuous sign language video needs to be segmented and then classified (Koller et al., 2015). Most modern approaches to SLR and CSLR rely on deep learning, but such approaches are data hungry and therefore are limited by the size of publicly available datasets.

The distinction between CSLR and Sign Language Translation (SLT) was stressed by Camgoz et al. (2018). SLT aims to translate a continuous sequence of signs to spoken language sentences (Sign to Text (S2T)) or vice versa (Text to Sign (T2S)), a challenging problem due to the changes in grammar and sequence ordering.

**Sign Language Production (SLP):** focusses on T2S, the production of a continuous sign language sequence given a spoken language input sentence. Current state-of-the-art approaches to SLP use transformer based architectures with attention (Stoll et al., 2018; Saunders et al., 2020). In this paper, we tackle the SLP task of neural sign language translation, defined as T2G or T2H translation.

HamNoSys has been used before for statistical SLP, with some success (Kaur and Kumar, 2014; Kaur and Kumar, 2016). However, the produced motion becomes robotic and is not practical for real world applications. Note that these approaches first convert the HamNoSys to SiGML, an XML format of HamNoSys (Kaur and Kumar, 2016).

**Neural Machine Translation (NMT):** NMT aims to generate a target sequence given a source sequence using neural networks (Bahdanau et al., 2014) and is commonly used for spoken language translations. Initial approaches used recurrence to map a hidden state to an output sequence (Kalchbrenner and Blunsom, 2013), with limited performance. Encoder-decoder structures were later introduced, that map an input sequence to an embedding space (Wu et al., 2016). To address the bottleneck problem, attention was introduced to measure the affinity between sections of the input and embedding space and allow the model to focus on specific context (Bahdanau et al., 2014). This was improved further with the introduction of the transformer (Vaswani et al., 2017) that used Multi-Headed Attention (MHA) to allow multiple projections of the learned attention. More recently, model sizes have grown with architectures introduced such as GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2018).

Different encoding/decoding schemes have been explored. BPE was first introduced in Sennrich et al. (2015), to create a set of tokens given a set vocabulary size. This is achieved by merging the most commonly occurring sequential characters. WordPiece, a similar tokenizer to BPE, was first introduced in Schuster and Nakajima (2012) and is commonly used when training language models such as BERT, DistilBERT and Electra. Finally, word and character level tokenizers break up a sentence based on white space and unique symbols respectively.

**Natural Language Processing:** NLP has many applications, for example Text Simplification, Text Classification, and Speech Recognition. Recently, deep learning approaches have outperformed older statistical methods (Vaswani et al., 2017). A successful NLP model must understand the structure and context of language, learned via supervised or unsupervised methods. Pre-trained language models have been used to boost performance in other NLP tasks (Clinchant et al., 2019; Zhu et al., 2020), such as BERT (Devlin et al., 2018) achieving state-of-the-art performance. Zhu et al., 2020 tried to fuse the embedding of BERT into a traditional transformer architecture using attention, increasing the translation performance by approximately 2 BLEU score.

Other methods have used Word2Vec to model language, this has been applied to many NLP tasks (Mikolov et al., 2013b). Word2Vec is designed to give meaning to a numerical representation of words. The central idea being that words with similar meaning should have a small euclidean distance between the vector representation.

In this paper, we take inspiration from these techniques to boost performance of the low resource task of T2G and T2H sign language production.

## 3. Methodology

The task of neural sign language production aims to map a source sequence of spoken language sentences, $x = (x_1, x_2, ..., x_W)$ with W words, to a sequence of glosses, $y = (y_1, y_2, ..., y_G)$ with G glosses (Text to Gloss (T2G)), or a sequence of HamNoSys, $z = (z_1, z_2, ..., z_H)$ with H symbols (Text to HamNoSys (T2H)). T2G and T2H tasks thus learn the conditional probabilities $p(y|x)$ and $p(z|x)$ respectively. Sign language translation is not a one to one mapping as several words can be mapped to a single gloss ($W > G$), ($W > H$). This increases the complexity of the problem as the model must learn to attend to multiple words in the input sequence.

Fig. 2 shows the general architecture of our model used to translate from spoken language to gloss/HamNoSys. For means of comparison, our baseline model is an encoder-decoder transformer with MHA. The input and output sequence are tokenized using a word level tokenizer and the embedding for a given sequence is created using a single linear layer. We later build on this base model using different tokenizers, embedding and supervision techniques. We train our model using a cross-entropy loss between the predicted target sequence, $\hat{x}$ and the ground truth sequence, $x^*$, defined as $L_T$.

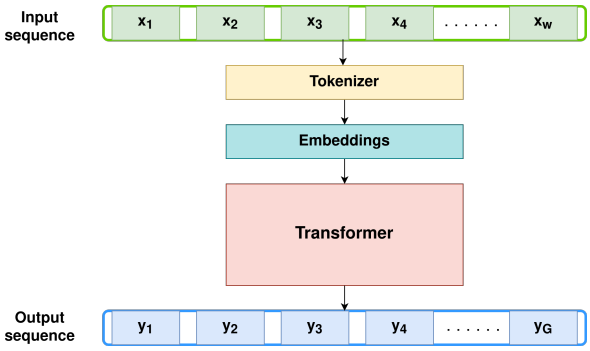


Figure 2: An overview of the different configuration of our architecture for SLT

In this section, we follow the structure of Fig. 2 from top to bottom. We start by describing the different tokenizers used to split the source text and produce tokens (Sec. 3.1). Next, we explain the different embedding techniques used to create a vector from the input tokens (Sec. 3.2). Finally, we talk about the advantages of using extra supervision and explain how this is implemented in conjunction with the translation loss.

### 3.1. Tokenizers

Several tokenizer schemes can be used on both the input and output such as BPE, Word, character and WordPiece. BPE (Sennrich et al., 2015), character and WordPiece (Schuster and Nakajima, 2012) all change the vocabulary size of the model by breaking sentences into sub-units. This reduces the number of singletons and reduces lexical inflections in the input and output sequences (Wolf et al., 2019).

**Word** A word level tokenizer segments the input sentence based on white space. Therefore, a normal sentence is split into whole words.

**Character** A character level tokenizer segments the text based on the individual symbols, reducing the vocabulary to simply the alphabet plus punctuation.

**BPE** BPE creates a base vocabulary containing all the unique symbols in the data, from which it learns a number of merge rules based on the most commonly occurring sequential symbols. An example of the BPE algorithm being applied to HamNoSys is shown in Fig. 3, with the coloured boxes indicating what merges are made at each step. Merging continues until a specific vocabulary size is reached. This helps reduce word inflections e.g. the words low, lowest and lower can be segmented to low, est and er. Over the whole corpus the suffix's (est and er) can be reused, collapsing the vocabulary in this example from 3 to 1.

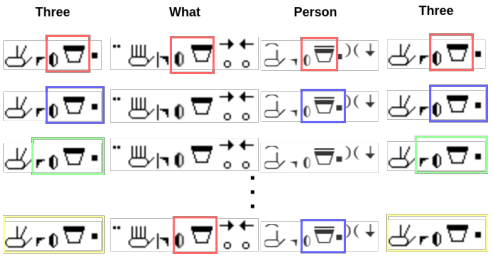


Figure 3: An example of how BPE can be applied to HamNoSys.

**WordPiece** We only apply a WordPiece tokenizer when embedding with BERT, as this is what the BERT model was trained with. WordPiece is another sub-unit tokenization algorithm similar to BPE that evaluates the lost benefit before merging two symbols, ensuring that all mergers are beneficial.

### 3.2. Embedding

After tokenization, the input sequence $x$ is then embedded by projecting the sequence into a continuous space (Mikolov et al., 2013a). The goal of embedding is to minimise the Euclidean distance between words with similar meanings. The most common embedding is a single linear layer, which takes an input sequence $x = (x_1, x_2, ..., x_W)$ with W words and turns it into a matrix of $[W \times E]$ where E is the models embedding width. In models such as BERT and Word2Vec, embeddings are learnt via training on a large corpus of spoken language data. To maximise the benefit from using BERT we fine tune the pre-trained model on the mineDGS dataset using masked-language modeling.

When using a BERT model, we define the transformation as follows. Given an input sequence $x$ we first

apply WordPiece tokenization.

$$X_{WP} = WordPiece(x) \tag{1}$$

Then apply the BERT embeddings as:

$$X_{BERT} = BERT(X_{WP}) \tag{2}$$

Note that we take the embedding from the last layer of BERT. We define the Word2Vec transformation as:

$$X_{W2V} = Word2Vec(x) \tag{3}$$

Additionally, we experiment with concatenating or fusing contextual information into the input $x$. We define the contextual information as $x_{ave}$ and the scaling factor as $S$, used to place additional emphasis on the contextual information. In the case of Word2Vec we take a mean average of each word's embedding in the sentence and treat this as a vector that contains information about the whole sentence. For BERT we use the embedding of the classification token ([CLS]), which contains contextual information about the sentence (Devlin et al., 2019). We either concatenate the information to the beginning of a sequence $x = (x_{ave} * S, x_1, x_2, ..., x_W)$ (CON), or we fuse it into each step of the sequence $x = ((x_{ave} * S) + x_1, (x_{ave} * S) + x_2, ..., (x_{ave} * S) + x_W)$ (ADD).

### 3.3. Supervision

In sign language, there exists a strong correlation between hand shape and meaning (Stokoe, 1980). Therefore, we investigate forcing the transformer to predict the hand shape alongside the gloss or HamNoSys sequences, to enrich the learnt representation. We scale the loss from the hand shape prediction $\mathcal{L}_{\mathcal{H}}$ by factor $F$. We combine both losses from the translation $\mathcal{L}_{\mathcal{T}}$ and hand shape prediction $\mathcal{L}_{\mathcal{H}}$ to create $L_{total}$ as:

$$L_{total} = L_T + (L_H * F) \tag{4}$$

In this setup, the model learns the joint conditional probability of

$$p(y|x) * p(H|x) \tag{5}$$

where $H$ is the sequence of hand shape symbols:

$$H = (h_1, h_2, ..., h_G) \tag{6}$$

and G is the number of glosses in the sequence. Overall this forces that model to focus on hand shape during training. We show that by forcing the model to predict hand shape we improve the performance on T2H.

## 4. Experiments

In this section we test the translation performance of our models in both the T2G and T2H setups. We first explain the experimental setup of our models. Next, we compare quantitative results against previous state-of-the-art and our own baselines. Finally we provide qualitative results.

### 4.1. Experimental Setup

When training our T2G model, we experiment with different embedding sizes, number of layers and heads. We observe a large change in performance based on these three parameters, and search for the best configurations for further tests. Our transformer uses a xavier initializer (Glorot and Bengio, 2010) with zero bias and Adam optimization (Kingma and Ba, 2014) with a learning rate of $10^{-4}$. We also employ dropout connections with a probability of 0.2 (Srivastava et al., 2014). When decoding, we use a beam search with a search size of 5.

Our code base comes from Kreutzer et al. (2019) NMT toolkit, JoeyNMT (Kreutzer et al., 2019) and is implemented using Pytorch. While our BPE and word piece tokenizers come from Huggingface's python library transformers (Wolf et al., 2019). When embedding with BERT, we use an open source pre-trained model from Deepset (Chan et al., 2020). Finally we used fasttext's implementation of Word2Vec for word level embedding (Mikolov et al., 2013b).

The publicly available mDGS dataset contains aligned spoken German sentences and their gloss counter parts, from unconstrained dialogue between two native deaf signers (Kaur and Kumar, 2014). The providers of this dataset also have a dictionary for all glosses in the corpus, of which some contain HamNoSys descriptions. Following the translation protocols set in Saunders et al. (2022), we created a subset of the mDGS dataset with aligned sentences, glosses and HamNoSys. mDGS is a larger dataset compared to PHOENIX14**T** (7.5 times more parallel examples, with a source vocabulary of 18,457) with 330 deaf participants performing free form signing. The size of mDGS overcomes some of the limitation of PHOENIX2014T. Note we remove the gloss variant numbers to reduce singletons.

We use the PHOENIX14**T** (Camgoz et al., 2018) dataset to compare our best model to previous NMT baseline results (Saunders et al., 2020; Stoll et al., 2018; Moryossef et al., 2021; Li et al., 2021). PHOENIX14**T** contains parallel monolingual German data, with approximately 7000 examples of aligned gloss and text.

### 4.2. Quantitative Evaluation

In this section, we evaluate our models on both mDGS and PHOENIX14**T** using BLEU (BLEU-1,2,3 and 4) and Rouge (F1-score) scores for both dev and test sets. We group our experiments in five sections:

1. Baseline T2G, T2H and Text to Gloss to HamNoSys (T2G2H) with a standard transformer.
2. T2G and T2H with different embedding layers and sentence averaging.
3. T2G and T2H with different tokenizers (BPE, Word, and Character).
4. T2G and T2H with additional supervision.
5. Comparison of our approach on PHOENIX14**T** and mDGS.

| Approach: | DEV SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | TEST SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
|---|---|---|---|---|---|---|---|---|---|---|
| Linear Layer | **16.26** | **24.14** | **32.83** | **43.05** | **42.02** | **16.47** | **24.51** | **33.27** | **43.58** | **41.53** |
| BERT | 14.69 | 21.51 | 29.39 | 38.66 | 30.87 | 14.2 | 21.19 | 29.09 | 38.33 | 30.31 |
| BERT SA ADD | 13.23 | 19.41 | 26.43 | 34.75 | 32.38 | 13.43 | 19.47 | 26.31 | 34.3 | 32.34 |
| BERT SA CON | 14.89 | 21.45 | 28.73 | 36.85 | 34.73 | 15.14 | 21.57 | 28.79 | 36.91 | 34.44 |
| Word2Vec | 11.47 | 17.59 | 24.68 | 34.21 | 29.45 | 11.73 | 17.83 | 25.14 | 34.90 | 30.22 |
| Word2Vec SA ADD | 13.8 | 21.07 | 29.72 | 42.29 | 30.65 | 13.31 | 20.56 | 29.31 | 42.13 | 30.67 |
| Word2Vec SA CON | 0.03 | 0.05 | 0.06 | 0.04 | 9.44 | 0.03 | 0.06 | 0.06 | 0.04 | 9.32 |

(a) MineDGS (mDGS) on Text to Gloss to HamNoSys (T2G2H)

| Approach: | DEV SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | TEST SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
|---|---|---|---|---|---|---|---|---|---|---|
| Linear Layer | 14.46 | 23.27 | 32.62 | 47.44 | 50.85 | 14.80 | 23.54 | 32.89 | 47.36 | 50.87 |
| BERT | **20.26** | **29.14** | **38.01** | **48.92** | **53.67** | **21.03** | **29.87** | **38.79** | **49.77** | **53.93** |
| BERT SA ADD | 14.64 | 22.33 | 30.91 | 43.99 | 50.30 | 15.16 | 22.92 | 31.41 | 44.21 | 50.33 |
| BERT SA CON | 11.82 | 19.2 | 27.39 | 40.58 | 53.36 | 12.21 | 19.39 | 27.44 | 40.48 | 53.67 |
| Word2Vec | 16.43 | 24.77 | 33.71 | 46.62 | 51.14 | 17.09 | 25.23 | 34.22 | 47.31 | 51.52 |
| Word2Vec SA ADD | 16.72 | 25.14 | 34.39 | 48.00 | 51.28 | 16.98 | 25.31 | 34.59 | 48.08 | 51.12 |
| Word2Vec SA CON | 14.98 | 22.49 | 30.65 | 42.42 | 51.11 | 15.18 | 22.65 | 30.80 | 42.75 | 50.10 |

(b) MineDGS (mDGS) on Text to HamNoSys (T2H)

Table 1: Embedding transformer results for Text to Gloss (T2G) and Text to HamNoSys (T2H) translation.

| Tokenizer Input | Output | DEV SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | TEST SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | Word | 16.47 | 24.35 | 33.06 | 43.41 | 36.32 | 16.55 | 24.45 | 33.14 | 43.54 | 36.34 |
| Word | BPE | **22.06** | **28.53** | **36.32** | **47.55** | 36.20 | **21.87** | **28.31** | **36.02** | **47.08** | 35.74 |
| Word | Char | 16.47 | 24.35 | 33.06 | 43.41 | 36.32 | 16.55 | 24.45 | 33.14 | 43.54 | 36.34 |
| BPE | Word | 20.84 | 26.77 | 34.02 | 44.77 | 35.31 | 20.84 | 26.80 | 34.12 | 44.97 | 35.35 |
| BPE | BPE | 21.39 | 27.28 | 34.31 | 43.86 | **36.61** | 21.28 | 27.25 | 34.34 | 43.86 | **36.86** |
| BPE | Char | 1.99 | 5.5 | 10.35 | 30.01 | 2.61 | 1.46 | 5.18 | 10.0 | 29.77 | 2.61 |

(a) MineDGS (mDGS) on Text to Gloss to HamNoSys (T2G2H)

| Tokenizer Input | Output | DEV SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | TEST SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | Word | 21.81 | **31.86** | **42.05** | **54.88** | **55.39** | 21.89 | **31.92** | **42.16** | **55.04** | **55.23** |
| Word | BPE | 25.41 | 29.28 | 34.11 | 41.25 | 48.03 | 25.54 | 29.39 | 34.25 | 41.35 | 48.09 |
| Word | Char | 21.63 | 31.76 | 41.99 | 54.94 | 55.3 | 21.59 | 31.79 | 42.09 | 55.01 | 55.14 |
| BPE | Word | 20.98 | 30.29 | 39.38 | 50.04 | 55.24 | 21.18 | 30.37 | 39.4 | 49.84 | 55.04 |
| BPE | BPE | **26.14** | 30.83 | 36.47 | 44.35 | 49.95 | **26.21** | 30.84 | 36.43 | 44.14 | 50.05 |
| BPE | Char | 1.91 | 6.01 | 11.72 | 37.56 | 37.22 | 1.92 | 5.88 | 11.59 | 37.63 | 37.31 |

(b) MineDGS (mDGS) on Text to HamNoSys (T2H)

Table 2: Tokenizer transformer results for Text to Gloss (T2G) and Text to HamNoSys (T2H) translation.

Note we expect the performance to be lower than 100 BLEU. As this is a translation problem there are several valid answers for a given input, thus human evaluation is still necessary. We are also unable to provide T2H results on PHOENIX14**T**, as HamNoSys is not available for some words in its vocabulary.

#### 4.2.1. Baseline Results

Our baseline models achieved a BLEU-4 score of 2.86 (T2G), 16.26 (T2G2H) and 14.46 (T2H) on the mDGS dev set. Our baseline setup uses a word level tokenizer on both the input and output, providing a baseline to ablate our proposed techniques in the next three sections. We perform a hyper-parameter search and make modification to the model architecture (number of heads, layers and embedding size) to find the best performance.

In general, a sequence of HamNoSys is significantly longer than it's gloss counter part, ($H >> G$). As a result our T2H performance is artificially higher than our T2G. Therefore, in order to make our T2G and T2H results comparable, we perform a dictionary lookup to convert the gloss to HamNoSys (T2G2H) before calculating the BLEU score. Given these results, we conclude a transformer architecture is the best baseline approach and continue with this setup for all future experiments.

#### 4.2.2. Embedding

Next we experiment with using different embedding techniques for the T2G and T2H tasks. As discussed in Section 3.1 we use a linear layer, BERT and Word2Vec in combination with sentence averaging. From the results in Table 1 we make several observations. Firstly, using a language model improves the translation performance on the T2H task (Tab. 1a). While on the T2G task, using language models was detrimental to the translation performance (Tab. 1b). We assume this is due to the reduced information within the gloss and smaller sequence length. Secondly, we observe that applying sentence averaging to the BERT embedding has a negative effect on the scores, independent of what type of average was used (adding or concatenating). On the other hand, adding the sentence averaging to the Word2Vec embedding marginally improved performance compared to the stand alone Word2Vec embeddings on T2H. But note

| | | DEV SET | | | | | TEST SET | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Approach: | Supervision | BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
| T2G2H | ✗ | **22.06** | **28.53** | **36.32** | **47.55** | **36.20** | **21.87** | **28.31** | **36.02** | **47.08** | **35.74** |
| T2G2H | ✓ | 21.79 | 27.98 | 35.45 | 46.21 | 35.79 | 21.49 | 27.76 | 35.27 | 46.11 | 35.99 |
| T2H | ✗ | 26.14 | 30.83 | **36.47** | **44.35** | **49.95** | 26.21 | 30.84 | **36.43** | **44.14** | **50.05** |
| T2H | ✓ | **26.99** | **31.07** | 35.99 | 42.73 | 48.89 | **27.37** | **31.42** | 36.3 | 42.92 | 48.85 |

Table 3: HamNoSys hand shape supervision results for Text to Gloss to HamNoSys (T2G2H) and Text to HamNoSys (T2H) translation.

that Word2Vec plus sentence averaging still has lower performance than just using a linear layer. Overall, we find the best performing embedding to come from using BERT, which scored 5.8 BLEU-4 higher than using a linear layer. This demonstrates that using a pre-trained language model can enhance translation.

#### 4.2.3. Tokenizer

We next experiment with using different tokenizers, as described in Section 3.2. We performed a parameter search to find the best vocabulary size for the BPE algorithm, which we find to be 2250 and 7000 on the input and output respectively. The result of our experiments are shown in Table 2.

When using a character level tokenizer each input contains a minimal amount of information (one letter). As expected this increases the difficulty of the problem, and reduces performance. When applied to the input it was extremely detrimental for the performance on both T2G2H and T2H, independent of which output tokenizer was used. Therefore to save space, we do not present the input character level results. Using a word level tokenizer achieved very reasonable results, supporting our theory that using larger units of language that contains more information is beneficial for translation. But as BPE outperformed the word level tokenizer on the BLEU-4 score, we assume that by using whole words we create a harder problem, as the dataset contains several word inflections. We conclude that BPE is the best algorithm to use when translating from T2H. This is due to the algorithms ability to reduce inflections and reduce the vocabulary size which simplifies the networks task. Our results also show that the biggest impact comes from having BPE on the output, suggesting that most of the challenge comes from the decoding section of the network. Similarly, the best T2G result came from using a word level and BPE tokenizers on the input and output respectively.

#### 4.2.4. Supervision

Our final ablation study investigates an additional loss explained in Section 3.3. This had a positive effect on the translation performance for T2H. As can be seen from Table 3, the use of supervision increased the BLEU-4 scores by 0.85. We conclude supervision enriches the learnt sign language representation due to the correlation between hand shape and context. Supervision forces the model to focus more on hand shape, allowing the model to group signs and find better trends in the data. Although the use of supervision marginally decreased the T2G2H BLEU score, we suggest this is due to reduced information in the target gloss.

### 4.3. State-of-the-art Comparisons

Finally, in Table 4 (PHOENIX14**T**) and 5 (mDGS) we compare our best performing models to state-of-the-art work. Note in Table 4 our baseline is marginally higher than (Saunders et al., 2020), we assume this is due to a larger hyper-parameter search. On both datasets, our best model for T2G and T2G2H uses a word level and BPE tokenizer on the input and output respectively. While our best T2H result comes from adding additional supervision on to this setup. As can be seen from Table 4 and 5 our models outperformed all other methods (Moryossef et al., 2021; Li et al., 2021; Saunders et al., 2020; Saunders et al., 2022; Stoll et al., 2018), setting a new state-of-the-art on PHOENIX14**T** and mDGS. Note we can only compare scores that are publicly available, therefore '-' denotes where the authors did not provide results.

### 4.4. Qualitative Evaluation

For qualitative evaluation, we share translation examples from our best models and our baseline model in Fig. 4, to allow the reader to better interpret the results. Note, we add a vertical black line after each word of HamNoSys to mark the end of a given sign. These results show how our BPE model has learnt richer translations than our baseline model.

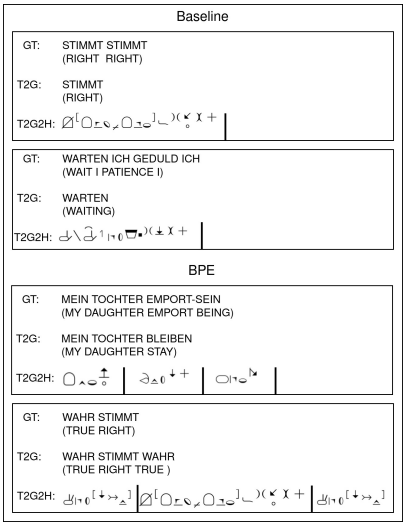


Figure 4: Translation examples from our baseline and best model.

| Approach: | DEV SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | TEST SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
|---|---|---|---|---|---|---|---|---|---|---|
| T2G (Stoll et al., 2018) | 16.34 | 22.30 | 32.47 | 50.15 | 48.42 | 15.26 | 21.54 | 32.25 | 50.67 | 48.10 |
| T2G (Saunders et al., 2020) | 20.23 | 27.36 | 38.21 | 55.65 | 55.41 | 19.10 | 26.24 | 37.10 | 55.18 | 54.55 |
| T2G (Li et al., 2021) | 18.89 | 25.51 | - | - | 49.91 | - | - | - | - | - |
| T2G (Moryossef et al., 2021) | 23.17 | - | - | - | - | - | - | - | - | - |
| **T2G Baseline (ours)** | 22.47 | 30.03 | 41.54 | 58.98 | 57.96 | 20.95 | 28.50 | 39.99 | 58.32 | 57.28 |
| **T2G Best Model (ours)** | **25.09** | **32.18** | **42.85** | **60.04** | **58.82** | **23.19** | **30.24** | **40.86** | **58.74** | **56.55** |

Table 4: Baseline comparison results for Text to Gloss (T2G) translation on PHOENIX14**T**.

| Approach: | DEV SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE | TEST SET BLEU-4 | BLEU-3 | BLEU-2 | BLEU-1 | ROUGE |
|---|---|---|---|---|---|---|---|---|---|---|
| T2G (Saunders et al., 2022) | 3.17 | - | - | - | 32.93 | 3.08 | - | - | - | 32.52 |
| **T2G Our best** | 10.5 | 14.35 | 20.43 | 33.56 | 35.79 | 10.4 | 14.21 | 20.2 | 33.59 | 35.99 |
| **T2G2H Our best** | 22.06 | 28.53 | 36.32 | 47.55 | 36.20 | 21.87 | 28.31 | 36.02 | 47.08 | 35.74 |
| **T2H Our best** | **26.99** | **31.07** | **35.99** | **42.73** | **48.89** | **27.37** | **31.42** | **36.3** | **42.92** | **48.85** |

Table 5: Baseline comparison results for Text to Gloss (T2G), Text to Gloss to HamNoSys (T2G2H) and Text to HamNoSys (T2H) translation on mDGS.

## 5. Conclusion

In this paper, we employed a transformer to translate from spoken language sentences to a sequence of gloss or HamNoSys. We introduced T2H translation, showing the advantages of translating to HamNoSys instead of just gloss, and set baseline results for future work on mDGS. We showed that language models can be used to improve translation performance, but using more advanced tokenization algorithms like BPE gives a larger performance gain. Additionally, we have shown that translation can be improved by training the model to jointly predict hand shape and HamNoSys. We achieved a BLEU-4 score of 26.99 and 25.09, a new state-of-the-arts for SLT on the mDGS and PHOENIX14**T** datasets.

As future work, it would be interesting to create a representation, gloss++. This could combine the benefits of gloss and HamNoSys, including non-manual features as well as hand shape information, as this has been shown to be useful for translation. Furthermore, this could be beneficial for down stream tasks in the SLP pipeline.

## 6. Acknowledgements

We thank Adam Munder, Mariam Rahmani and Marina Lovell from OmniBridge, an Intel Venture, for supporting this project. We also thank Thomas Hanke and University of Hamburg for use of the mDGS data. We also thank the SNSF Sinergia project 'SMILE II' (CRSII5 193686), the European Union's Horizon2020 research project EASIER (101016982) and the EPSRC project 'ExTOL' (EP/R03298X/1). This work reflects only the authors view and the Commission is not responsible for any use that may be made of the information it contains.

## 7. Bibliographical References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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

Chan, B., Schweter, S., and Möller, T. (2020). German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*.

Clinchant, S., Jung, K. W., and Nikoulina, V. (2019). On the use of bert for neural machine translation. *arXiv preprint arXiv:1909.12744*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North*.

Efthimiou, E., Fotinea, S.-E., Vogler, C., Hanke, T., Glauert, J., Bowden, R., Braffort, A., Collet, C., Maragos, P., and Segouat, J. (2009). Sign language recognition, generation, and modelling: A research effort with applications in deaf communication. In *International Conference on Universal Access in Human-Computer Interaction*.

Efthimiou, E., Fontinea, S.-E., Hanke, T., Glauert, J., Bowden, R., Braffort, A., Collet, C., Maragos, P., and Goudenove, F. (2010). Dicta-sign–sign language recognition, generation and modelling: A research effort with applications in deaf communication. In *Proceedings of the 4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies*.

Elliott, R., Glauert, J. R., Kennaway, J., Marshall, I., and Safar, E. (2008). Linguistic modelling and language-processing technologies for avatar-based sign language presentation. *Universal Access in the Information Society*.

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

Hanke, T. (2004). Hamnosys-representing sign lan-

guage data in language resources and language processing contexts. In *LREC*.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Kaur, R. and Kumar, P. (2014). Hamnosys generation system for sign language. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*.

Kaur, K. and Kumar, P. (2016). Hamnosys to sigml conversion system for sign language automation. *Procedia Computer Science*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

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

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

Li, D., Xu, C., Liu, L., Zhong, Y., Wang, R., Petersson, L., and Li, H. (2021). Transcribing natural languages for the deaf via neural editing programs. *arXiv preprint arXiv:2112.09600*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*.

Moryossef, A., Yin, K., Neubig, G., and Goldberg, Y. (2021). Data augmentation for sign language gloss translation. *arXiv:2105.07476*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*.

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

Saunders, B., Camgoz, N. C., and Bowden, R. (2021a). Anonysign: Novel human appearance synthesis for sign language video anonymisation. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*.

Saunders, B., Camgoz, N. C., and Bowden, R. (2021b). Mixed signals: Sign language production via a mixture of motion primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Saunders, B., Camgoz, N. C., and Bowden, R. (2022). 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)*.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *CoRR*.

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

Stokoe, W. C. (1980). Sign language structure. *Annual Review of Anthropology*.

Stoll, S., Camgöz, N. C., Hadfield, S., and Bowden, R. (2018). Sign language production using neural machine translation and generative adversarial networks. In *Proceedings of the 29th British Machine Vision Conference*.

Tamura, S. and Kawasaki, S. (1988). Recognition of sign language motion images. *Pattern Recognition*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

WHO. (2021). Deafness and hearing loss.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., Li, H., and Liu, T.-Y. (2020). Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*.

## 8. Language Resource References

Hanke, T., Schulder, M., Konrad, R., and Jahn, E. (2020). Extending the Public DGS Corpus in size and depth. In Eleni Efthimiou, et al., editors, *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages*, pages 75–82, Marseille, France, May. ELRA.

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