Frozen Pretrained Transformers for Neural Sign Language Translation

Mathieu De Coster mathieu.decoster@ugent.be
IDLab-AIRO - Ghent University - imec, Technologiepark-Zwijnaarde 126, Ghent, Belgium

Karel D'Oosterlinck karel.doosterlinck@ugent.be
Marija Pizurica marija.pizurica@ugent.be
Paloma Rabaey paloma.rabaey@ugent.be
Severine Verlinden severine.verlinden@ugent.be
Ghent University, Jozef Plateaustraat 22, Ghent, Belgium

Mieke Van Herreweghe mieke.vanherreweghe@ugent.be
Ghent University, Blandijnberg 2, Ghent, Belgium

Joni Dambre joni.dambre@ugent.be
IDLab-AIRO - Ghent University - imec, Technologiepark-Zwijnaarde 126, Ghent, Belgium

Abstract

One of the major challenges in sign language translation from a sign language to a spoken language is the lack of parallel corpora. Recent works have achieved promising results on the RWTH-PHOENIX-Weather 2014T dataset, which consists of over eight thousand parallel sentences between German sign language and German. However, from the perspective of neural machine translation, this is still a tiny dataset. To improve the performance of models trained on small datasets, transfer learning can be used. While this has been previously applied in sign language translation for feature extraction, to the best of our knowledge, pretrained language models have not yet been investigated. We use pretrained BERT-base and mBART-50 models to initialize our sign language video to spoken language text translation model. To mitigate overfitting, we apply the frozen pretrained transformer technique: we freeze the majority of parameters during training. Using a pretrained BERT model, we outperform a baseline trained from scratch by 1 to 2 BLEU-4. Our results show that pretrained language models can be used to improve sign language translation performance and that the self-attention patterns in BERT transfer in zero-shot to the encoder and decoder of sign language translation models.

1 Introduction

Despite recent advancements in the domain of automated sign language translation (SLT), substantial challenges remain. One considerable issue is the lack of labeled data. Deep neural networks are data-hungry and neural machine translation models are no exception. The widely used SLT dataset RWTH-PHOENIX-Weather 2014T (Camgoz et al., 2018) contains only 8,257 parallel sentences. As a comparison: for translation between spoken languages, 6,000 parallel sentences is considered a "tiny" amount (Gu et al., 2018).

In this work, we investigate how transfer learning can be used to improve the generalization of *video-to-text* SLT models in the absence of large datasets. We integrate pretrained language models such as BERT (Devlin et al., 2019) into our translation models, evaluating translation

performance with BERT2RND and BERT2BERT models (Rothe et al., 2020). We also use mBART-50 (Tang et al., 2020), which has been trained on corpora of 50 languages, including German, as our goal is to translate from German sign language (DGS) to German.

These pretrained models are prohibitively large for currently available SLT datasets. The base version of BERT, for example, consists of 12 layers compared to 3 layers in the encoder of the translation model of Camgoz et al. (2020). Therefore, we perform aggressive layer pruning and parameter freezing in the pretrained models.

We use the RWTH-PHOENIX-Weather 2014T dataset for which both gloss and text annotations are available. We consider joint continuous sign language recognition (CSLR) and SLT, or Sign2(Gloss+Text), as it is called by Camgoz et al. (2020). We compare several combinations of pretrained transformers and transformers trained from scratch in terms of scores (WER for CSLR and BLEU-4 for SLT) and number of trainable parameters.

Our results show that the BERT based models (BERT2RND, BERT2BERT) perform best. These models allow us to outperform the baseline, i.e., transformers trained from scratch. Due to overfitting, the large mBART-50 model results in significantly worse performance than the baseline. This warrants further investigation in the incorporation of existing language models. We discuss possible options for future work in Section 6.

The source code of this research project is available at `https://github.com/m-decoster/fpt4slt`.

2 Neural Sign Language Translation

Several SLT systems have been proposed in the past, including rule-based systems (Zhao et al., 2000) and statistical methods (Bungeroth and Ney, 2004). We focus specifically on translation from a sign language to a spoken language. To perform this translation, the sign language first needs to be converted into a written or computational form. Various notation systems exist, such as glosses and HamNoSys (Prillwitz, 1989).

Bungeroth and Ney (2004) focus on Text2Gloss and Gloss2Text translation. Recent advancements in deep learning and computer vision now allow for Sign2Text translation, but this requires sizable corpora. Several large datasets exist for sign language *recognition* tasks, in which the goal is to classify glosses from sign language video, e.g., MS-ASL (Vaezi Joze and Koller, 2019), WLASL (Li et al., 2020) and AUTSL (Sincan and Keles, 2020). RWTH-PHOENIX-Weather 2014T is a large public dataset for sign language *translation* (Camgoz et al., 2018). Along with it, a neural SLT model is formalized and introduced: a recurrent encoder-decoder with Luong attention (Luong et al., 2015). This Sign2Gloss2Text model achieves a BLEU-4 score of 18.13 on the test set.

In a follow-up work, the recurrent architecture is replaced by transformers (Camgoz et al., 2020). The authors present a new study showing improvements in BLEU-4 scores for Gloss2Text and Sign2Text translation. By jointly performing CSLR and SLT, they increase the BLEU-4 test score to 21.32.

Yin and Read (2020) further improve the performance of sign language transformers by using multiple cues (face, hand, full frame and pose information, rather than only full frame information) for CSLR and performing Sign2Gloss2Text translation. They achieve a BLEU-4 test score of 24 (25.40 using an ensemble of 5 models). These improvements are related to feature extraction rather than network architecture, whereas we aim to improve the translation model by creating a more powerful encoder-decoder model (an orthogonal approach).

3 Transfer Learning and Frozen Pretrained Transformers

Transfer learning from high-resource to low-resource language pairs can result in better translation performance for low-resource language pairs (Zoph et al., 2016). Pretraining with huge

monolingual corpora can also improve performance of downstream tasks such as sentiment classification and question answering (Devlin et al., 2019). Pretrained models such as BERT can also be adapted as encoders or decoders in a machine translation model to improve translation performance (Rothe et al., 2020).

Transfer learning between different sign languages has previously been used to improve model performance in sign language recognition (Pigou et al., 2017; Albanie et al., 2020). The use of a continuous sign language recognition model as feature extractor for SLT, as performed by Camgoz et al. (2020), is also a form of transfer learning. To the best of our knowledge, so far no one has leveraged transfer learning of encoders and decoders (rather than feature extractors) for SLT from signed to spoken languages.

Lu et al. (2021) show that pretrained language models can replace transformers trained from scratch for downstream tasks, even if those downstream tasks are not related to natural language processing (NLP). This is contrary to earlier research, where language models were used in a parameter-efficient transfer learning set-up from one NLP task to another (Houlsby et al., 2019). Self-attention and feedforward layers are frozen and only the layer normalization parameters are fine-tuned. These are, for large language models, only a tiny fraction of the parameters (often less than 1%). As a result, these models, called Frozen Pretrained Transformers (FPTs), are more robust against overfitting. We combine the approaches of Rothe et al. (2020) and Lu et al. (2021) for SLT: we aim to leverage pretrained language models, while freezing the majority of the parameters to avoid overfitting on our extremely small dataset.

4 Methodology

We use the following experimental set-up. First, we reproduce the baseline of Camgoz et al. (2020) using the code base provided by the authors[1]. Then, we lay out several experiments with different models with either an FPT encoder or FPT encoder and decoder. We compare several degrees of fine-tuning, ranging from only layer normalization to everything except the self-attention layers. Due to computational constraints, we use a sequential approach for hyperparameter tuning. However, we tune two correlated hyperparameters together: the number of layers and degree of fine-tuning.

4.1 Sign Language Translation

Our SLT approach follows that of Camgoz et al. (2020). We use a transformer encoder-decoder model with 3 layers and 8 attention heads as a baseline. We use the same features, which were obtained from a model trained on CSLR (Koller et al., 2019). Camgoz et al. (2020) report BLEU-4 scores of 22.38 and 21.32 on the development and test set of RWTH-PHOENIX-Weather 2014T, respectively. These scores are obtained for loss weights $\lambda_R = 10$ and $\lambda_T = 1$. Our reproduction using the code base provided by the authors yields 20.18 and 19.86. The difference is possibly due to a different random seed, resulting in a less than optimal weight initialization in our case. From communication with the authors, we learned that they reported the best result out of ten runs with different random seeds. We select only a single seed and report all of our results using the resulting random initialization.

4.2 Frozen Pretrained Transformers

We run three experiments. We first aim to improve CSLR and SLT performance by integrating an FPT (BERT) in the encoder of the Sign2(Gloss+Text) model. The decoder is trained from scratch. We name this method BERT2RND, according to the nomenclature of Rothe et al. (2020). Secondly, we evaluate a BERT2BERT model, where a cross-attention module is added to the BERT model integrated in the decoder; this module is trained from scratch. Finally, we

[1] https://github.com/neccam/slt

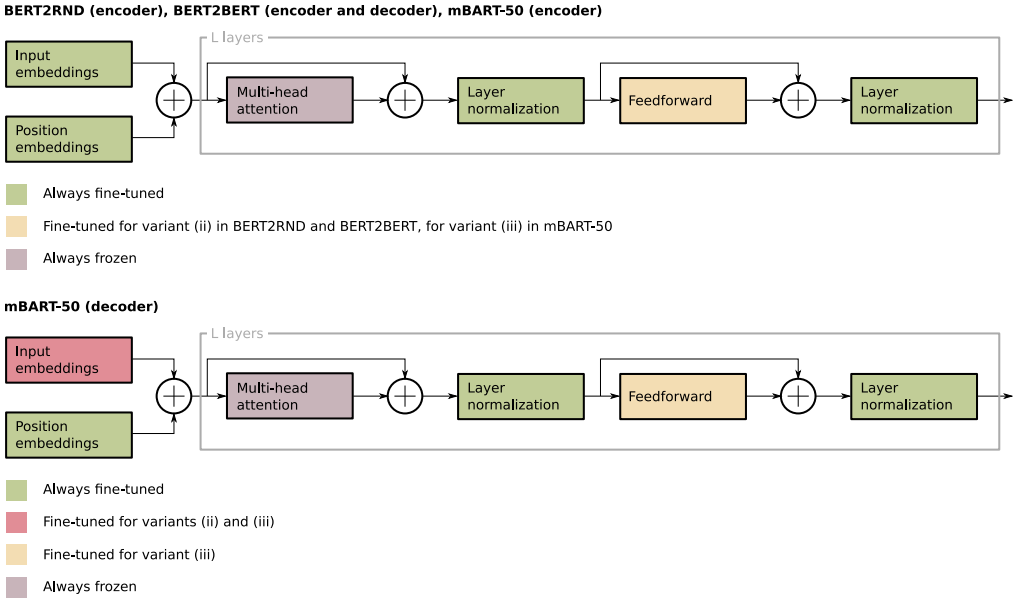

Figure 1: Model variants and the degrees of freezing. BERT2RND only has a pretrained encoder, while for BERT2BERT and mBART-50 both the encoder and decoder are pretrained.

investigate whether integrating a language model pretrained on the target language (German) improves the quality of translations (measured in BLEU-4 scores). We use mBART-50 to initialize both the encoder and decoder. We freeze the pretrained cross-attention module in the decoder, but add a randomly initialized linear layer to transform the encoder outputs such that they better align with the pretrained cross-attention patterns. The German target sentences are tokenized using the mBART-50 tokenizer so that we are able to reuse the token embeddings in the decoder. By default, we freeze the mBART-50 token embeddings as they contain over 250 thousand elements. However, we investigate the impact of freezing and fine-tuning them.

We wish to minimize the amount of trainable parameters to mitigate overfitting. We compare different degrees of fine-tuning (ordered by increasing number of trainable parameters). We name these the "model variants". For BERT2RND and BERT2BERT there are two variants:

(i) fine-tune layer normalization parameters, positional embeddings, sign embeddings and decoder token embeddings,

(ii) fine-tune all of the above and feedforward layers.

For mBART-50 there is an additional variant, because the token embeddings in the decoder are not fine-tuned by default:

(i) fine-tune layer normalization parameters, positional embeddings and sign embeddings only,

(ii) fine-tune all of the above and token embeddings,

(iii) fine-tune all of the above and feedforward layers.

A schematic overview of which layers are frozen and which are not is shown in Figure 1.

For every model, we train a linear input layer for which the weights are initialized orthogonally in the encoder and as Gaussian in the decoder - based on findings by Lu et al. (2021). No such input layer is trained for the decoder in BERT2RND as the decoder is trained from scratch.

4.3 Hyperparameter Tuning

We tune several hyperparameters, including the learning rate, number of layers, loss weights, decoding beam size and beam alpha. Note that we aim to optimize the translation score (BLEU-4) rather than the continuous sign language recognition score (WER). All hyperparameter tuning and model selection is therefore performed based on the BLEU-4 score. We first tune the learning rate per model architecture on a model with 3 layers in both encoder and decoder (the same as the baseline model). We then perform all further tuning and experiments with the learning rate that yields the highest development set BLEU-4 score. Ideally, one would tune the learning rate together with the other hyperparameters (number of layers, loss weights), but the computational cost prohibits this for our experiments. The optimal learning rates obtained for each model are $3\mathrm{e}{-4}$ for BERT2RND, $1\mathrm{e}{-4}$ for BERT2BERT and $1\mathrm{e}{-3}$ for mBART-50.

We then perform simple layer pruning to reduce the model complexity. The amount of layers that we keep is a hyperparameter n (between 1 and the number of layers in the pretrained transformer) that is tuned to maximize the development set BLEU-4 score. When pruning, we always keep the first n layers, and replace deeper ones by identity functions. The number of layers n is tuned for all model variants as we hypothesize that freezing more parameters per layer allows for deeper networks. Because of computational constraints, we cannot check all values of n for all models. Because deeper models rapidly start overfitting, we choose $n = 1, 2, 3, 6, 12$ (12 being the original number of layers in BERT-base and mBART-50). For BERT2RND, the decoder is kept as it is in the original implementation: a 3 layer transformer trained from scratch.

We select the optimal number of layers per experiment and tune the loss weight $\lambda_R = 1, 5, 10$, as Camgoz et al. (2020) show that this can have a large impact on the translation score.

After training of each experiment, the best beam size and alpha are found by tuning for both WER and BLEU-4 on the development set. The results are reported on the development set and test set using these parameters.

All models are optimized with a batch size of 32, with the Adam optimizer (Kingma and Ba, 2015). The optimizer parameters are set to $\beta = (0.9, 0.998)$ and the weight decay to $1\mathrm{e}{-3}$, as per Camgoz et al. (2020). We also apply a similar learning rate scheduling approach: we decrease the learning rate with a factor 0.7 whenever the development set BLEU-4 score has not increased for 800 iterations. We stop training when the learning rate is smaller than $1\mathrm{e}{-7}$.

5 Results

We find an optimal number of layers and loss weights for the different models based on the development set BLEU-4 score. We discuss the amount of trainable parameters for the optimal models found for each variant, while at the same time listing the development set score for the optimal models. We finally compare the development set and test set results with the baseline.

5.1 Optimal Number of Layers

We find that for all three model architectures, the optimal number of layers is low. This is as expected due to the small dataset size. For BERT2RND, the difference in performance between freezing approaches is small for a low layer count. For larger layer counts, fine-tuning only the layer normalization parameters (model variant (i)) yields better performance, because this model overfits less than the others. For BERT2BERT, fine-tuning the feedforward layers consistently results in better performance. The gap between the model variants is larger than for

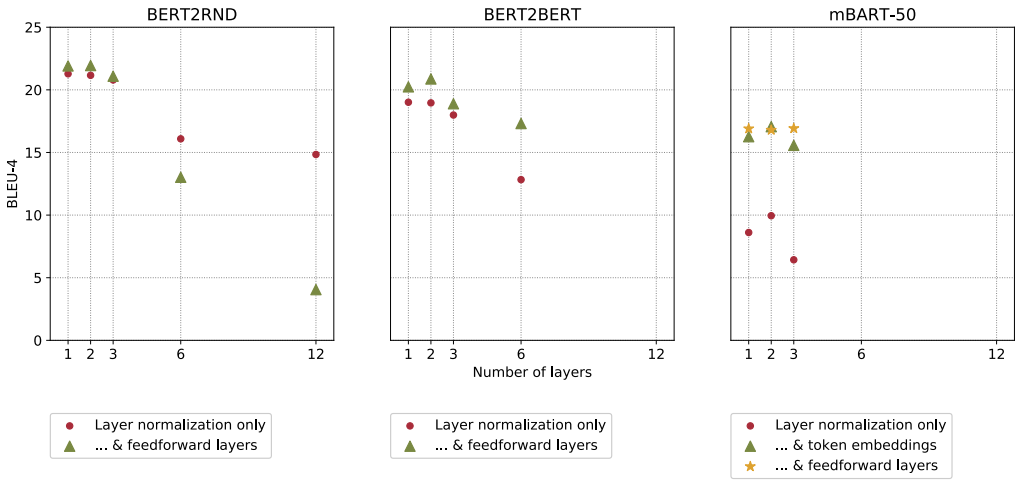

Figure 2: BLEU-4 scores for different model sizes of the three architectures, for $\lambda_R = \lambda_T = 1$. The models reach the highest BLEU-4 scores for 1, 2 or 3 layers. The missing values for BERT2BERT with 12 layers and for mBART-50 with 6 and 12 layers are due to VRAM constraints.

Table 1: The number of trainable parameters of different models and their best obtained development set BLEU-4 score. The best result is indicated in bold.

Model	Variant	Layers	Trainable parameters	BLEU-4
Transformer	N/A	3	38,878,270	20.18
BERT2RND	(i)	1	30,292,542	21.28
	(ii)	2	39,740,478	**22.47**
BERT2BERT	(i)	2	34,195,518	20.87
	(ii)	2	53,085,246	21.26
mBART-50	(i)	2	7,180,606	10.64
	(ii)	2	263,235,902	17.06
	(iii)	3	313,608,510	16.92

BERT2RND. We assume that the decoder benefits more from having more degrees of freedom than the encoder. The mBART-50 model underfits when the token embeddings are not fine-tuned: model variant (i) only achieves a maximum BLEU-4 score of 9.95 for 2 layers. The scores are consistently lower than for BERT2RND and BERT2BERT. Fine-tuning the decoder token embeddings is clearly required. However, even then we are unable to achieve baseline performance with mBART-50. An overview of the results is shown in Figure 2.

5.2 Amount of Trainable Parameters

We list the number of trainable parameters for every model variant in Table 1. We only consider the optimal number of layers (see Figure 2). The baseline count is 39 million. While several of our FPT models have more trainable parameters (including our best model with 40 million), we are able to match the baseline performance using a 1-layer BERT2RND model where only the layer normalization parameters are fine-tuned (30 million parameters). In this model, the self-attention encoder only has 600 thousand trainable parameters (1 layer), compared to 9 million when training a 3-layer encoder from scratch.

Table 2: Comparison of WER and BLEU-4 scores for the baseline and our best FPT models. The best results are indicated in bold.

Model	Development		Test	
	WER ↓	BLEU-4 ↑	WER ↓	BLEU-4 ↑
Transformer (Camgoz et al., 2020)	**24.98**	22.38	**26.16**	21.32
Transformer (reproduced)	30.48	20.18	29.96	19.86
BERT2RND (ii, 2 layers, $\lambda_R = 5$)	36.59	**22.47**	35.76	**22.25**
BERT2BERT (ii, 2 layers, $\lambda_R = 10$)	40.99	21.26	39.99	21.16
mBART-50 (ii, 2 layers, $\lambda_R = 1$)	40.25	17.06	39.43	16.64

5.3 Comparison with the Baseline

We compare the best results for each model architecture with the best results reported by Camgoz et al. (2020). The comparison is shown in Table 2. The best model is the 2-layer BERT2RND model with fine-tuning of feedforward layers (variant (ii)). With a test BLEU-4 score of 22.25, it outperforms the baseline. We see an increase of 0.93 compared to the baseline established by Camgoz et al. (2020) and of 2.39 compared to our reproduction of that model. A 2-layer BERT2BERT model with fine-tuning of feedforward layers outperforms the reproduced baseline by 1.3 BLEU-4, and achieves comparable performance to the model reported in the previous work. For mBART-50, we find that a 2-layer model with frozen attention patterns and feedforward layers achieves 16.64 BLEU-4.

6 Discussion

In this work, we have proposed using FPTs for neural SLT. We have compared several encoder-decoder architectures:

- BERT2RND, where the encoder is an FPT initialized from BERT-base
- BERT2BERT, where both encoder and decoder are FPTs initialized from BERT-base
- mBART-50, where encoder and decoder are FPTs initialized from mBART-50

6.1 Comparison Between Architectures

Out of these architectures, BERT2RND achieves the best results, with a BLEU-4 test score of 22.25 (an increase of 0.93 compared to the baseline established by Camgoz et al. (2020) and of 2.39 compared to our reproduction of that model). The best BERT2BERT model achieves 21.16. When freezing more parameters, we see a larger performance drop than for BERT2RND. Likely, the decoder benefits more from being trained from scratch or fine-tuned.

Contrary to our expectations, using mBART-50 pre-trained on (among others) German text, did not improve the translation quality for SLT with German as the target language. In fact, mBART-50 yields results below the baseline: 16.64 BLEU-4. We observe that fine-tuning the token embeddings is essential: freezing them results in a catastrophic drop in performance. Fine-tuning the cross-attention module to allow better alignment between the sign language encoder representations and spoken language decoder representations could prove useful, but brings with it the risk of overfitting.

6.2 Perspective and Future Work

The use of FPTs in low-resource scenarios such as SLT appears promising. FPTs can be integrated in translation models to improve translation performance. Our experiments show that

for comparable performance, FPTs can be slightly shallower than transformers trained from scratch. This could prove useful for reducing the computational cost during inference.

Techniques such as tokenization and (neural) embedding computation of written text have matured more than feature extraction for SLT due to machine translation between spoken languages receiving more research attention than SLT. Furthermore, current state of the art machine translation models are designed first and foremost for translation between spoken languages. Architectural changes may prove beneficial or even necessary to obtain better SLT performance. This is reflected in our results. Modelling sign language (in the encoder) appears to benefit more from pretrained language models than the modelling of spoken languages (in the decoder).

Further research should investigate the use of smaller, bilingual, models as FPTs. Fine-tuning the translation model on written texts concerning the topics present in the SLT data (before training on actual sign language data) may also result in better translations. Finally, another interesting research track is the integration of a translation model as a prior rather than as an FPT (Baziotis et al., 2020).

Next to trying to find better architectures for sign language translation, better feature extraction methods are also being researched. Any advancements in feature extraction for SLT can be combined with architectural improvements such as FPTs to further increase the quality of the translation model.

While SLT models have significantly improved in terms of translation metrics over the past few years, there still remains a large gap to bridge before they can be applied in real-world settings. In particular, these metrics, such as BLEU scores, do not always match well with the perceived quality of the translations by human evaluators (Callison-Burch et al., 2006). For this reason, future work will focus on identifying the main shortcomings that need to be addressed in SLT in order to achieve its performance acceptable for Deaf and Hard of Hearing (DHH) communities. This question needs to be addressed from, both, a technical and a linguistic perspective, and in close collaboration with the end users of potential SLT applications. Co-creation with DHH community members is therefore key.

7 Conclusion

We have presented and compared three different approaches to using pretrained language models for a Sign2(Gloss+Text) SLT task: BERT2RND, BERT2BERT and mBART-50. We outperform the baseline, which is a transformer trained from scratch, by replacing the encoder with the first 2 layers of BERT-base. We freeze the attention patterns and show that the patterns learned during the training of the BERT model transfer in zero-shot to SLT. The decoder can also be initialized using a pretrained language model, but we obtain better results if the decoder is trained from scratch. We attempt to integrate mBART-50, a multi-lingual translation model, hoping to obtain translations of a higher quality due to this model having been pretrained on German texts. However, we are unable to reach baseline performance with mBART-50. Further research should investigate whether smaller translation models obtain better performance, or whether language models can be used as priors to regularize the SLT model. Our best result, a BLEU-4 score of 22.25 on the test set of RWTH-PHOENIX-Weather 2014T, is obtained using a BERT2RND model. The BERT2RND methodology is easy to implement and can be combined with other advances such as improvements in feature extraction.

Acknowledgements

Mathieu De Coster's research is funded by the Research Foundation Flanders (FWO Vlaanderen): file number 77410. This work has been conducted within the SignON project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017255.

References

Albanie, S., Varol, G., Momeni, L., Afouras, T., Chung, J. S., Fox, N., and Zisserman, A. (2020). BSL-1K: Scaling up co-articulated sign language recognition using mouthing cues. In *European Conference on Computer Vision*, pages 35–53. Springer.

Baziotis, C., Haddow, B., and Birch, A. (2020). Language model prior for low-resource neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7622–7634.

Bungeroth, J. and Ney, H. (2004). Statistical sign language translation. In *Workshop on representation and processing of sign languages, LREC*, volume 4, pages 105–108. Citeseer.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the role of BLEU in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

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*, pages 7784–7793.

Camgoz, N. C., Koller, O., Hadfield, S., and Bowden, R. (2020). 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.

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 American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Gu, J., Hassan, H., Devlin, J., and Li, V. O. (2018). Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Koller, O., Camgoz, N. C., Ney, H., and Bowden, R. (2019). Weakly supervised learning with multi-stream CNN-LSTM-HMMs to discover sequential parallelism in sign language videos. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2306–2320.

Li, D., Rodriguez, C., Yu, X., and Li, H. (2020). Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. (2021). Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Pigou, L., Van Herreweghe, M., and Dambre, J. (2017). Gesture and sign language recognition with temporal residual networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3086–3093.

Prillwitz, S. (1989). *HamNoSys: Version 2.0; Hamburg notation system for sign languages; an introductory guide*. Signum-Verlag.

Rothe, S., Narayan, S., and Severyn, A. (2020). Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Sincan, O. M. and Keles, H. Y. (2020). AUTSL: A Large Scale Multi-Modal Turkish Sign Language Dataset and Baseline Methods. *IEEE Access*, 8:181340–181355.

Tang, Y., Tran, C., Li, X., Chen, P.-J., Goyal, N., Chaudhary, V., Gu, J., and Fan, A. (2020). Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Vaezi Joze, H. and Koller, O. (2019). MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. In *The British Machine Vision Conference (BMVC)*.

Yin, K. and Read, J. (2020). Better sign language translation with STMC-transformer. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5975–5989.

Zhao, L., Kipper, K., Schuler, W., Vogler, C., Badler, N., and Palmer, M. (2000). A machine translation system from English to American Sign Language. In *Conference of the Association for Machine Translation in the Americas*, pages 54–67. Springer.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.