CS230

Final Report CS230

Vu Nguyen
lamanhvu@stanford.edu

Elnaz Ansari
elnazans@stanford.edu

Ben Walker
bwalker0@stanford.edu

Abstract

American Sign Language (ASL) is crucial for communication among the deaf community. However, not everyone can communicate in sign language. This paper explores techniques for translating intermediate sign language text to English and German in both PyTorch and TensorFlow.

1 Introduction

Over 10 million Americans have some form of hearing disability (*Disability Impacts All of Us Infographic* 2020). This impacts their daily lives and communication with others. Many use ASL to communicate, which unfortunately is not commonly taught to hearing people. We want to play a role in bridging this gap to provide better experiences for this community by experimenting with ASL translation. In practice, the signs are usually translated to intermediate texts (called "glosses") which are then translated into English. A similar process is used in translating German Sign Language to German gloss to German.

In this project, we focus on the gloss translation problem; inputting German or English gloss texts and outputting the translated sentences. We use a Transformer based model to accomplish this, and explore various data augmentation techniques and model complexities. Additionally, we democratized the state-of-the-art model from Yin and Read 2020 by porting it from OpenNMT-py to OpenNMT-tf, and working on optimizing the TF model to TF-lite for IoT and edge device deployment.

2 Related work

While Sign Language Recognition (SLR) is a popular field, Sign Language Translation (SLT) is less well explored due to the complexity of the translation tasks and lack of available datasets. The first public SLT dataset was released in 2014, and only contained German gloss.

Existing work efforts mainly focus on an Encoder-Decoder approach. The first relevant study (by N. C. Camgoz et al. 2018) explores attention-based encoder-decoder networks. The same team later proposed a novel transformer-based architecture that performs both sign language recognition and translation in an end-to-end manner in Necati Cihan Camgoz et al. 2020. They injected intermediate gloss supervision using a Connectionist Temporal Classification loss function to make the model treat SLR and SLT as a single problem. The model simultaneously solves the co-dependant sequence-to-sequence learning problems (Sign to Gloss, and Gloss to Text) and leads to a significant performance gain. Both of these papers focus on German Glosses.

For the English counterpart, Yin and Read 2020 is the state-of-the-art approach that achieves the highest benchmark BLEU score. Similar to N. C. Camgoz et al. 2018, this paper also uses a Transformer based architecture with encoder-decoder.

A different approach in Li et al. 2020 further incorporates temporal semantic structures of sign videos to learn discriminative features. It skips the entire process of translating to gloss, and translates directly from video to text. However, this is beyond our scope as we want to focus on gloss to text translation.

3 Dataset and Features

We obtained ASLG-PC12 and PHOENIX-Weather 2014T datasets, which contain English and German glosses respectively. Othman and Jemni 2012 introduced the ASLG-PC12 and N. C. Camgoz et al. 2018 contains the PHOENIX-Weather 2014T dataset (Table 1). Before training, all input sentences are converted to lower-case.

In addition to changing the text to lower-case, we implemented several data augmentation methods, known as Easy Data Augmentation (EDA), a simple but effective technique used in many NLP applications to reduce overfitting and achieve more robust models as shown in Wei and Zou 2019. EDA is a word level data augmentation, and our implementation utilises four different techniques: synonym replacement, random synonym insertion, random swapping, and random deletion.

For English, we used the Wordnet dataset from Miller 1995 to randomly replace gloss words with available synonyms, which are selected randomly from a geometric distribution (p=0.5). We incorporated a similar method to insert synonyms for random words. Random swapping is achieved by randomly selecting 10% of the words in a sentence and swapping their positions, while random deletion is the same except that the words are deleted instead. For all of these techniques four additional modified sentences are generated for each original, as seen in Wei and Zou 2019. In addition, we explored the idea of concatenating several lines of glosses and their respective target lines in order to generate a new train dataset with longer sentences. After generating the longer glosses/sentences we applied the EDA to the new dataset. We could achieve BLEU score of 70 for the English model with all these techniques applied.

Dataset	ASLG-PC12	PHOENIX-Weather 2014T
Train	82,710	7,096
Validation	4,000	519
Test	1,000	642

Table 1: Breakdown statistics of each dataset.

Due to limited German language understanding of the team, we did not have access to a comprehensive German thesaurus and only tried the random deletion and random swapping techniques, with the same parameters as with the English.

4 Methods

Our model is based on the same Transformer architecture in Vaswani et al. 2017 and Yin and Read 2020. We pass processed sentences into the Encoder, where each input word is turned into an embedding of size 512. Each word position is encoded, with dropout of of 0.1. We have 2 layers in each Encoder and Decoder component. We use the same number of layers described in Yin and Read 2020 instead of the 6 used in Vaswani et al. 2017. However, we did confirm that an increased number of layers don't seem to have an impact on BLEU score. Within each layer, we have a Multi-headed Attention layer followed by LayerNorm, a Feedforward layer, and another LayerNorm. For the Feedforward layer, we use softmax loss. See appendix 7.1 for a simplified model architecture overview.

The model in Yin and Read 2020 uses a PyTorch implementation with the OpenNMT-py framework, and the code for this is provided by the authors. In order to add flexibility, we have ported this model to TensorFlow (OpenNMT-tf). This framework has more configurability, partly as a result of it being more mature.

Aligning with our hope to democratize this model even more, we quantized the TensorFlow model to reduce its footprint. The resulting model could be deployed in environment with scarce resources like IoT or mobile devices. We leveraged Ctranslate2 project to perform quantization of the model into the supported Float16, Int16, and Int8 options, where we trade off size with performance. These quantized models then translate on the same test set to obtain a BLEU score for comparison.

5 Experiments/Results/Discussion

5.1 Metrics

For our evaluation, we rely mainly on BLEU Score as a metric. BLEU is a machine translation standard and is believed to correlate well with human judgement. We use validation accuracy synonymously with BLEU score, and a higher score is better. Exported model size is also important, as too large a model cannot be efficiently used in resource constrained environments like IoT devices.

5.2 Error Analysis Discussion

Our test set errors generally contain: wrong tense or plurality (#2 in Table 2), or wrong word choice (#3 in Table 2). Out of 100 randomly sampled from mismatched translations, the majority are wrong tense or plurality (76%), and the rest are wrong word choice (24%) (See Figure 2). This means we should focus on trying to correct wrong tense or plurality cases to improve BLEU score. However, an additional question to ask is whether human can perform better than the current model. Upon examining the original gloss, we see many vague cases that would be hard for human as well. For example, for the gloss *"X-WE MAKE DEBT AND PASS X-Y ON TO X-WE CHILD ."*, our human translation is *"We make debt and pass onto our children"*, same as the model's. However, the target text is *"We made debts and passed them on to our children ."*. When we translate the

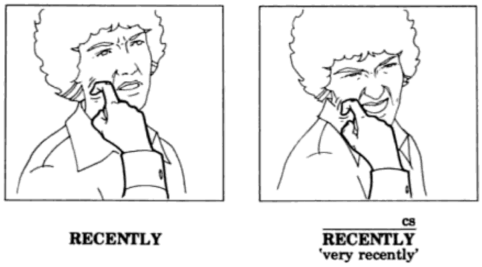


Figure 1: Example Temporal Clue in Sign Language (Baker-Shenk and Cokely 2002).

76 cases of model's wrong translations using the original gloss, our human translation matches the wrong model's translation 65.8%. Therefore, there might be a small room for improvement.

Two major factors contributed to the wrong translation that both human and model make. Firstly, the original gloss usually contain raw word without verb conjugation. While key words such as "we", "he", "she" can be used for conjugations verbs, the gloss does not usually contain such clue for past tense. As the above example demonstrates, both present tense and past tense could be applicable. Secondly, the model appears to use the noun of the sentence to conjugate the verb. However, there are examples where the nouns themselves are turned into plurals in the target, which then impact the verb conjugation. Both of these factors make it difficult for the model to correctly translate the input. However, we assert that the translation still makes sense to an English speaker. As an extra discussion point, there are subjective temporal clues for ASL (see example in Figure 1 for "Recently" and "Very Recently", where the level of exaggeration indicates how recent an event was). While this could be captured as part of a end-to-end translation from video to English text, we do not have this captured in our gloss dataset. This suggests some hints as to why an approach that trains both video-to-gloss and gloss-to-text such as Li et al. 2020 might yield better results, as it could capture these clues better.

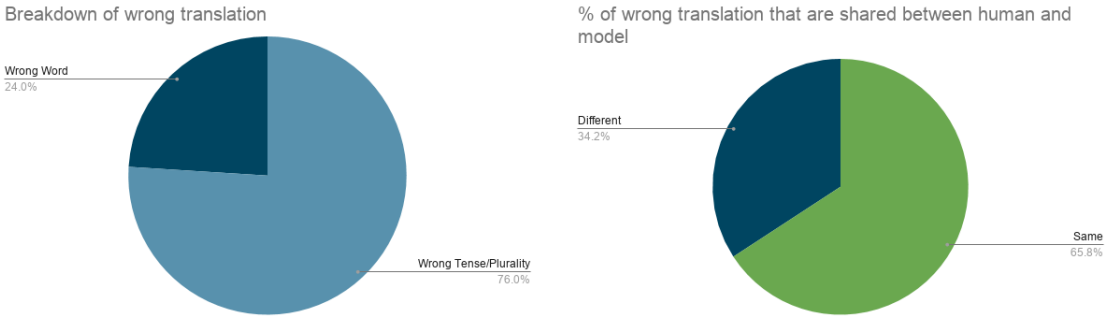


Figure 2: Error Analysis Breakdown.

[Example Translation #1 (Correct)]

- **Input:** ALLOW X-I TO START BY SAY THAT X-MY GROUP THINK THIS BE DESC-VERY DESC-GOOD REPORT .
- **2-Layers Pred:** allow me to start by saying that my group thinks this is a very good report .
- **3-Layers Pred:** allow me to start by saying that my group thinks this is a very good report .
- **6-Layers Pred:** allow me to start by saying that my group thinks this is a very good report .
- **Translation Target:** allow me to start by saying that my group thinks this is a very good report .

[Example Translation #2 (Wrong Tense)]

- **Input:** RESULT SPEAK FOR X-MSELVES .
- **2-Layers Pred:** the result spoke for themselves .
- **3-Layers Pred:** the result speaks for themselves .
- **6-Layers Pred:** the result spoke for themselves .
- **Translation Target:** the results speak for themselves .

[Example Translation #3 (Wrong word)]

- **Input:** IN THIS WAR X-WE BE DESC-NOT HOSTAGE BUT COMBATANT .
- **2-Layers Pred:** in this war we are not hostage but the robber.
- **3-Layers Pred:** in this war we are not hostages but the combatants.
- **6-Layers Pred:** in this war we are not hostage but a disproportion.
- **Translation Target:** in this war we are not hostages but combatants .

Table 2: Example English Gloss and English Translation.

5.3 Beam Search

Investigating errors further, we tried to understand whether issues stemmed from Beam Search or Modeling. We manually exported the constructed sentences that Beam Search came up with - if the optimal choice is among the options but not chosen, then the beam search is at fault. Otherwise, if the optimal choice is missing, then the models themselves are not learning the right translation.

Beam Width	BLEU Score
10	**91.86**
8	90.41
5	90.42
3	90.34

Table 3: Beam width and BLEU Score.

We printed out the options that Beam Search had to choose from. For a lot of cases, we saw a translation similar to target text. Therefore, one potential avenue for improvements could be tuning beam width. We experimented with a beam width of 10 and scaled down as Yang, Huang, and Ma 2018 discussed the phenomenon of the "beam search curse", where translation quality degrades with beam sizes larger than 5 or 10. Incidentally, this is among the six greatest challenges for NMT described in Koehn and Knowles 2017. In Yin and Read 2020, a beam width of 5 is deemed as optimal for ASLG dataset. Based on our experiments (Table 3), beam width 10 is the most optimal.

5.4 Data Augmentation

When testing our data augmentation techniques, we saw a negligible drop in evaluation accuracy for synonym-based techniques on the English dataset. We believe that while the data augmentation techniques do help us acquire more diverse data, accuracy is difficult to increase when it is already high at 91%. Unfortunately, random swap and random deletion did not improve the accuracy for either the English or the German datasets. From table 4, we suspect that random swap and deletion in a dataset with majority shorter sentences makes it harder for the model to learn.

	Number of words in a sentence				
Dataset	<5 words	5<=x<10	10<=x<15	15<=x<20	20<=x
English	17.9%	48.1%	31.9%	1.73%	0.2%
German	18.9%	53.7%	22.2%	4.2%	0.8%

Table 4: Statistics of word usage (%) in sentences in each train dataset.

5.5 Hyper-parameter tuning

To improve on the cases where the model translates to the wrong word, we also vary the Encoder and Decoder layer sizes in the Transformer. As we add more layers, the model becomes larger and more complex, without a corresponding accuracy increase (Figure 3). Analyzing some examples (Table 2) suggests that models with more layers might be overfitting. In Example #2, models with 2, 3, and 6 layer translations are roughly equivalent, with some plurality and tense differences. However, in the longer sentence of Example #3 the differences are more apparent. Having 2 layers gave a result much closer to the correct translation compared to 3, 4, or 6 layers. This is particularly interesting as 6 layers is a common industry suggestion from Vaswani et al. 2017, which leads to a much larger model without much gain in performance in this case.

One of the theories we had was whether the model was struggling with out-of-vocab words. To test this, we also increased the vocab size. Eventually, we used 50,000 as vocab size, but realized that the vocab built from the dataset only consisted of 21,000 words. With a bigger set of gloss to build vocab from, the model might be able to perform better. However, we are limited by the available gloss dataset currently, and building our own is expensive even with sign language expertise in the team.

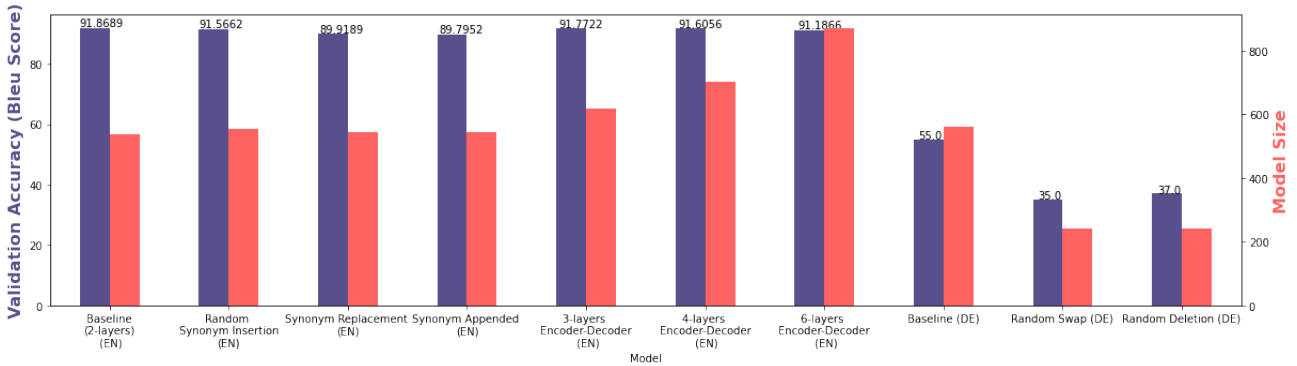


Figure 3: Comparison of different experimental models

5.6 TensorFlow vs. PyTorch

The similarity in test scores between the PyTorch and TensorFlow implementations indicate that the accuracies of the trained models are very close. The left most graph in Figure 4 shows that the models train in about the same speed, with PyTorch reaching a maximum BLEU of 91.8 in 4600 seconds, and TensorFlow reaching a similar BLEU of 91.9 in 4500 seconds. These models are running on the same AWS instance with no other significant CPU/GPU usage. Looking at the right most graph in Figure 4, we can see that the TensorFlow model performs slightly "slower" in that it takes more steps to converge than the PyTorch model, even though it runs those steps faster. The TensorFlow and PyTorch models both use gradient accumulation, building up the gradient updates for 3 batches of 2048 examples before applying them to the model, resulting in a larger effective

batch size of 6144. If we turn this off for the TensorFlow model and update after every batch, we see that it trains significantly faster while still reaching a similar level of accuracy.

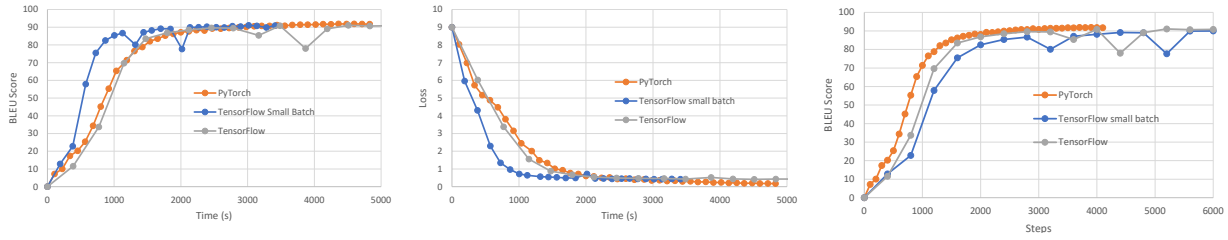


Figure 4: Comparison of PyTorch and TensorFlow

5.7 Inference Performance

The inference speed between the two models is similar, with the PyTorch model running slightly faster. One significant difference is that when inferring, TensorFlow re-configures the network and retraces the model, taking on the order of 10-20 seconds which significantly adds to the time. If the model is being served, however, this time is a one-off cost, and so in a production environment would not have a big impact.

Model	BLEU Score	Inference Time (s)
PyTorch	91.4	13.2
TensorFlow	91.86	13.8

Table 5: Inference Time and BLEU Score.

5.8 Quantized Models

To reduce the size of the model, we explore serialization and computing support weights with reduced precision: 16-bit floating points, 16-bit integers, and 8-bit integers. While the model size drops significantly, it appears that our BLEU score remains high (Table 6). In fact, going from float16 to int16 doesn't seem to sacrifice any weights. Once the model is quantized, it can only be used for inference as it merely translates the weights into formats that took up less space. Yet, this work poses an exciting avenue for embedding the model in different environments where the gloss-to-text translation will become less of a resource bottlenecks.

Model	BLEU Score	Model Size (Mb)
Baseline	91.86	536
Float16	81.0	86
Int16	81.0	86
Int8	80.9	44

Table 6: Quantized Models and BLEU Score.

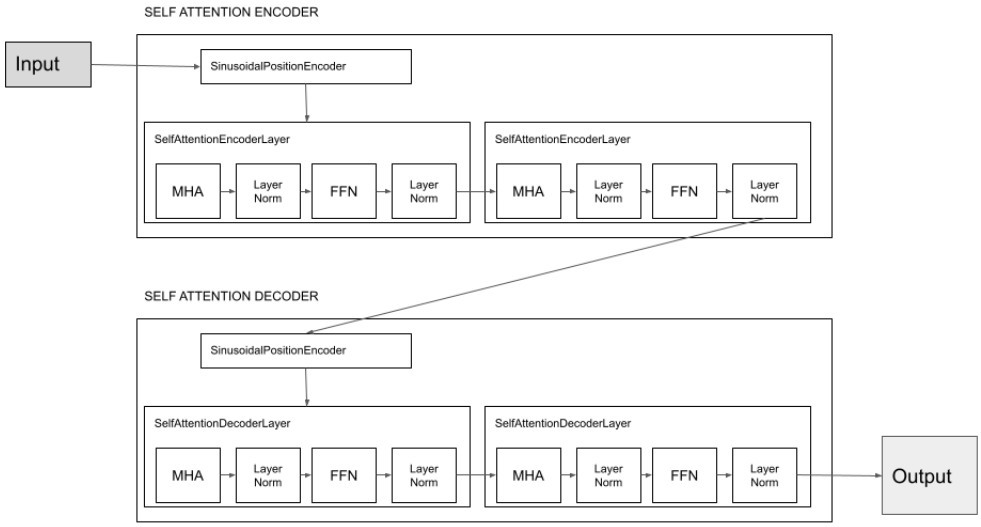
6 Conclusion

In conclusion, we think that the model performance is comparable to human translation, making it difficult for hyperparameter tuning alone to significantly increase BLEU score. We believe that in order to improve this performance more, additional datasets might be required. Otherwise, an expanded model that trains on both video signals to capture some temporal clue needed for the translation is necessary.

Another potential route for improving this model is to test how to reduce the number of parameters without sacrificing performance significantly. We have shown some success with quantization, hinting at the possibility that we could lose some precision but save tremendous on resource footprint. In such a world where ASL translation should be an integral part of any accessibility system, having a smaller model that performs decently could change many lives in impactful ways. We hope that any future work would focus more on this aspects of the project.

7 Appendix

7.1 Model Architecture



7.2 Contributions

- Everyone contributed to the project ideation, reports, presentation slides, video, and literature review.
- Vu: Worked on error analysis, hyperparameters tuning for beam search width and layer size, quantization of TensorFlow models.
- Elnaz: Worked on dataset preparation, Easy Data Augmentation (EDA), concat data augmentation, and final result analysis for both PyTorch and TensorFlow models (for English as well German datasets).
- Ben: Worked on porting to TensorFlow, training analysis, performance tuning for model training and inference

References

[20] *Disability Impacts All of Us Infographic*. Sept. 2020. URL: https://www.cdc.gov/ncbddd/disabilityandhealth/infographic-disability-impacts-all.html.

[BC02] Charlotte Baker-Shenk and Dennis Cokely. *American sign language: a teachers resource text on grammar and culture*. Clerc Books, 2002.

[Cam+18] N. C. Camgoz et al. "Neural Sign Language Translation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7784–7793. DOI: 10.1109/CVPR.2018.00812.

[Cam+20] Necati Cihan Camgoz et al. *Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation*. 2020. arXiv: 2003.13830 [cs.CV].

[KK17] Philipp Koehn and Rebecca Knowles. "Six Challenges for Neural Machine Translation". In: *CoRR* abs/1706.03872 (2017). arXiv: 1706.03872. URL: http://arxiv.org/abs/1706.03872.

[Li+20] Dongxu Li et al. *TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation*. 2020. arXiv: 2010.05468 [cs.CV].

[Mil95] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

[OJ12] Achraf Othman and Mohamed Jemni. "English-ASL Gloss Parallel Corpus 2012: ASLG-PC12". In: May 2012.

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[WZ19] Jason Wei and Kai Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. DOI: 10.18653/v1/D19-1670. URL: https://www.aclweb.org/anthology/D19-1670.

[YHM18] Yilin Yang, Liang Huang, and Mingbo Ma. "Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation". In: *CoRR* abs/1808.09582 (2018). arXiv: 1808.09582. URL: http://arxiv.org/abs/1808.09582.

[YR20] Kayo Yin and Jesse Read. "Better Sign Language Translation with STMC-Transformer". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5975–5989. DOI: 10.18653/v1/2020.coling-main.525. URL: https://www.aclweb.org/anthology/2020.coling-main.525.