Exploring the Transferability of Transformer Language Models on Sign Language Recognition

Gunjan Bhattarai and **Amit Joshi**
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
{gunjan.bhattarai,
amitjoshi24}@utexas.edu

Abstract

In this work, we explored adding an external language model, a Transformer decoder modeled after GPT-2, to an ASL to English fingerspelling translation system. Despite the existing recognition model only using a 1-layer unidirectional LSTM to make predictions, we only saw an increase of 0.13% accuracy with our approach (beam size of 2). While this is likely enough to reach a new state-of-the-art benchmark if scaled to a beam size of 5-7, the dramatic reduction in inference speed made it impossible for us to check. Future research should aim to further improve our language model while dramatically boosting inference speed so that the Deaf community can finally have their own Google Translate-style automatic sign language interpretation system.

1 Introduction and Preceding Work

The last few years has seen an explosion in the capabilities of machine translation models (Wu et al., 2016) (Vaswani et al., 2017) (Liu et al., 2020). Because of these advancements, most of us enjoy access to instantaneous speech- and text-based translation via apps like Google Translate. However, the same cannot be said of the Deaf community. Translating English to sign language in an online setting only exists for single word translations, and sign language to English is still almost exclusively the domain of trained human interpreters. Given that such resources would bring immense benefits to the Deaf community in part by freeing them from the need of an interpreter, it is imperative that further research be done to build state-of-the-art translators, speech recognizers, and sign-to-speech converters.

While there have been efforts by researchers to build ASL recognition systems, many of them have room for considerable architectural updates. To our knowledge, the current state-of-the-art in sign language recognition is held by Yin et al. (2020), which uses an STMC network to translate videos of ASL signs to an intermediate text form called gloss. The decoder of this module, however, still relies on RNNs, which has been shown to trail Transformer decoders in image captioning tasks (Zhu et al., 2018). To convert the gloss into text, they use a two-layer encoder-decoder Transformer for machine translation. The authors also forgo the use of an external language model, which is essential in ultra-low resource settings like ASL translation.

One key challenge that the STMC-Transformer does not address is the large number of out-of-vocabulary words present in ASL. Given the difficulty in creating a new sign for all ASL speakers to use every time a new word is invented, the Deaf community has coalesced around fingerspelling (signing a word letter by letter) to augment the ASL vocabulary. Fingerspelling ends up being used for a wide variety of words, including people's names, places, titles, technical and scientific vocabulary, and even some day-to-day words like “couch”, “bills”, and “gift”. Overall, about 35% (Padden and Gunsals, 2003) of words used in ASL are fingerspelled, with the number increasing further for new ASL speakers and Deaf children with limited knowledge of ASL signs. Some signs have even been derived from fingerspelling (e.g., “TY” for toy, “BSY” for busy, and “WLD” for would), further emphasizing the importance of fingerspelling to ASL recognition and translation tools.

In recognition of this, Shi et al. (2019) introduces an end-to-end system translating fingerspelled signs to characters. They use a AlexNet (Krizhevsky et al., 2012) convolutional encoder pretrained on ImageNet and a LSTM decoder with a novel form of attention called iterative visual attention. Iterative visual attention combines Luong-style attention (Luong et al., 2015), optical flow, and zooming into the image to compute weights. The LSTM decoder is appended to an external LSTM-based character language model with shallow fusion to predict letters. They also introduce a vastly enlarged dataset called ChicagoFSWild+ to assist other researchers in improving upon their work.

However, like the STMC-Transformer, plenty of avenues for architectural improvements remain. First, the bedrock of iterative visual attention is dependent upon an RNN architecture and has not been updated to be compatible with the scaled dot product attention mechanism introduced with the Transformer. Language models have also vastly been improved with the GPT series (Radford et al., 2018) (Radford et al., 2019) (Brown et al., 2020) and the usage of self-attention to model long-range dependencies. However, modern language models make use of Byte Pair Encoding (Sennrich et al., 2015) for their vocabulary instead of characters, which means they cannot be employed directly for ASL to English decoding tasks. Finally, AlexNet has been significantly improved upon for 2D convolution tasks by models such as EfficientNet (Canonne et al., 2019) and NFNet-F4+ (Brock et al., 2021), making it a very dated choice to be building state-of-the-art sign language recognition models.

As a follow-up to Shi et al. (2019)’s work, Mahoudeau (2020) switches the pretrained AlexNet backbone with ResNet-50, using the Mixed Convolutional Tube (MiCT) architecture (Zhou et al., 2018). The idea behind MiCT is to convert a 3D convolution problem into a smaller 2D and 3D convolutions to drastically reduce computational costs (due to lower parameter sizes). Zhou et al. (2018)’s model splits the video into overlapping sets of 3 consecutive 2D video frames. In each MiCT block, 3D convolutions of these 3 different video frames are concatenated with either a 2D convolution of the first image of the sequence (if it is the first MiCT block) or the output of the previous MiCT block. After this, the model uses either 2 layers of 2D CNNs or Inception blocks before passing its result to the next MiCT block. After all the video frames have been iterated through, the model runs global pooling and a fully connected layer. Aside from this change, Mahoudeau (2020) also removes the language model entirely, relying entirely on the RNN’s decoders to form predictions. To our knowledge, this is currently the state-of-the-art result.

It is worth noting that both Shi et al. (2019)’s and Mahoudeau (2020)’s models were trained using Connectionist Temporal Classification (CTC) loss (Graves et al., 2006). This loss is measured for an entire sequence, so it is a natural choice for problems in sign recognition. CTC loss is computed by reduction of the ground truth Y into a Hidden Markov Model (HMM)[1]. This is done by taking the ground truth sequence $Y = \{y_1, y_2, \dots y_n\}$ and adding epsilon (blank) characters between each of the elements to get $\{\varepsilon, y_1, \varepsilon, y_2, \dots y_n\} = \{s_1, s_2, \dots, s_n\}$. From these states, we can create a linear HMM with self-loops. The other edges in this graph are added when $\varepsilon \neq s_1 \neq s_{i+2}$. After adding start and end dummy nodes, we define the HMM parameters. The start/initialization probability is fully distributed on the start node, because a sequence must start with the ‘<bos>’ token. All transition edges have value 1. The emission probabilities (probability of observing state o at time t), are given by the probabilities that the final (softmax) layer outputs. Then, the HMM forward algorithm gives us $P(Y \mid X)$, the probability of the ground truth label sequence given the sequence that the model observed. Taking the negative log of this value gives us our loss value. This value is differentiable with respect to our model’s outputs, allowing the model to be trained using backpropagation. CTC loss-calculated values end up being critical to choosing values to rescore with beam search, so its usage should be carefully analyzed.

[1]An excellent resource for understanding and visualizing CTC loss and decoding: https://distill.pub/2017/ctc/

In this project, we aim to investigate improving the external language model appended to the decoder. While numerous studies can attest to the improvement of Language X to English translation/recognition by providing the decoder access to more English data (Sennrich et al., 2015), these results are only generalizable to translations of ASL signs to English words. Given that fingerspelling translation requires the usage of character-level vocabulary and access to a curated English text limited only to fingerspelled words, a newly trained language model is needed for the task. Making things harder is the fact that there is no definitive corpus of words that are either never fingerspelled or always fingerspelled, leaving us heavily dependent on trial and error to train the best language model for fingerspelling translation. Our work aims to formally start that process.

2 Architecture and Training Process

Our architecture largely follows Shi et al. (2019)'s approach with iterative visual attention, aside from two key changes. First, we take a page from Mahoudeau (2020) and employ MiCT with his pretrained weights for our convolutional encoder. Second, we replace the external LSTM language model originally trained on ChicagoFSWild+'s text.

For the language model, we use the 117M parameter GPT-2 Transformer decoder architecture, switching to a 36-character vocabulary from the standard 50,257 BPE tokens. Aside from the 26 lowercased letters of the alphabet, we use '.', '&', '.', ':', ' ', '<unk>', "<mask>', '<s>' (start), '<\s>' (end), and '<pad>'. We train using a NVIDIA Tesla P100 for 10 hours on a set of 6.3 million Wikipedia titles. We use an effective batch size of 600,000 characters (600 sets of 100 characters, with gradient accumulation occurring every 10 steps) and a learning rate of 6.0 x 10-4, taking inspiration from the hyperparameters of GPT-3 Small. Gradient checkpointing [16] (Chen et al., 2016) is essential to fitting non-trivial batches into GPU memory.

When scoring specific letter combinations as possible sign translations, we make use of three different factors: the negative log likelihood of the sequence as calculated using CTC Loss, our custom GPT-2 model's negative log likelihood, and an insertion bonus to favor longer sequences. Each of these can be multiplied by a tunable hyperparameter to optimize results. After selecting specific multiplicative hyperparameters, we summed up these three values with weights (i.e. shallow fusion) to get our overall decoding objective score, given by:

$$\text{Score}(W) = -\log(P(Y \mid X)) + \beta * LM(Y) + \gamma * |W|$$

One problem that we faced was that doing a forward propagation on our GPT-2 model (character LM) was that it was exceptionally slow (although part of this may be because we only used 1 GPU, foregoing the parallelization advantages Transformers offer). Shi et al. (2019) had done work with memoizing their LSTM LM scores in a history (lookup) table. However, they only kept one lookup table per beam search iteration of each sample. This design choice perplexed us, so we decided to keep one large lookup table throughout the entire testing process. Each character sequence would map to a probability score given by our GPT-2 Character LM, and if the character sequence had been seen before, the score would simply be retrieved from the lookup table. However, we understand that this isn't a "real-world" task, meaning that using our model in real life would not require it to decode 1735 samples. As a result, we also conducted memoization experiments where we cleared the lookup table after every sample. This allowed us to conduct experiments much more quickly, although not enough to do significant experimentation on beam sizes larger than 2. Speed improvements are noted in Figure 3.

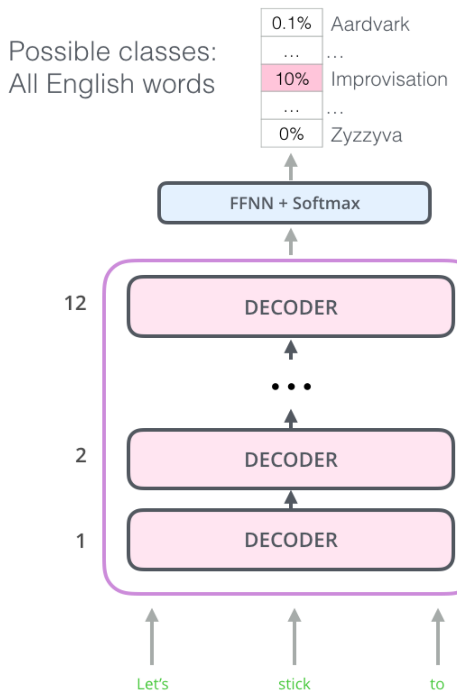

Figure 1: Diagram of a standard GPT-2 model. The decoder Transformer tokens as inputs, feeds it into several multi-head self-attention and feedforward layers, and then predicts a set of probabilities corresponding to what it believes is the next token. We use this structure almost identically in our experiments except for changing the standard byte-pair encoding to a character-level vocabulary.

3 Experiments

We ran our trained language model on the ChicagoFSWild dataset, which had 5,455 training sequences from 87 signers, 981 development (validation) sequences from 37 signers, and 868 test sequences from 36 signers, with no overlap in signers in the three sets (Shi et al., 2018). Of these, the test sequences were the only relevant ones given that we had used the pretrained weights from Mahoudeau (2020)'s work. These pretrained weights came from the ASL translation system being run on the ChicagoFSWild+ dataset, which had 50,402 training sequences from 216 signers (Table 1), 3,115 development sequences from 22 signers, and 1,715 test sequences from 22 signers.

We measure our results in terms of letter accuracy, measured the exact same way as proposed by Shi et al. (2019):

$1 - \frac{S+D+I}{N}$, where S, D, and I are the number of substitutions, insertions, and deletions in the alignments and N is the total number of ground-truth letters.

Greedy search (beam size of 1) did not lead to any changes in results when applying the language model. This occurred because even if the language model disagreed with the ASL sign recognition system's predictions, it didn't have any other characters to choose from when rescoring.

LM Weight	Insertion Bonus	Accuracy
0	0	77.68% (Baseline)
0.025	0	77.71% (+0.03%)
0.05	0	77.81% (+0.13%)

0.05	0.3	77.81% (+0.13%)
0.1	-3	73.97% (-3.71%)
0.1	0	77.78% (+0.10%)
0.1	0.3	77.71% (+0.03%)
0.1	3	73.61% (-4.07%)
0.2	0	77.68% (unch)
0.2	0.3	77.81% (+0.13%)
1	0	74.21% (-3.67%)
10	0	44.64% (-33.04%)

Figure 2: Table of results for beam size 2, with iterative zooming scale of 3. Baseline comes from Mahoudeau (2020), which is equivalent to not having an external language model at all. Overall, usage of the language model tended to slightly improve results, with a modest insertion bonus also improving results.

Figure 2 shows our results when testing with a beam size of 2. While a language model was found to improve results, it was often by disappointing margins. Three of our runs ended up reaching our best score of 77.81%, which only improved upon Mahoudeau (2020)'s work by a paltry 0.13%. Anecdotally, very few changes were made to predictions to begin with, with only 1-3 letters changing every 50 samples or so (at least for the models that didn't have enormous LM or insertion weights that drowned the model's original predictions). Given that a beam size of 2 took 30-40 (15-20 after our speed improvements) minutes to run in full, we opted not to experiment with larger beam sizes.

Our worst results (0.1/-3, 0.1/+3, 1/0, and 10/0 in terms of language model weight / insertion bonus) did not come as a surprise to us. We had specifically chosen these hyperparameter values to investigate what would happen if either insertion bonus or the GPT-2 language model came to dominate the rescoring process. 10/0's results were particularly impressive given that the results came almost entirely from the language model (the original model's scores were only used as tiebreakers in extremely rare cases and to determine the top 2 characters to be tested in the first place). As for the other hyperparameter choices, we started with the 0.1/0 weights used by Shi et al. (2019) and changed them up or down to investigate the effect of small changes in weights in terms of overall accuracy. Ultimately, their hyperparameter selections were fairly effective -- 0.1/0 got our fourth best results and was only 0.03% off from tying our top model.

Model	Mean Sample Running Time (s)	FPS
Mahoudeau with LM (no memoization)	2.840 (Baseline)	10.5 (Baseline)
Our model (memoization over all samples)	1.411 (-50.32%)	21.2 (+101.90%)
Our model (memoization where the lookup table was cleared after each sample)	1.464 (-48.45%)	20.5 (+95.24%)

Figure 3: Table of results noting speed of running beam size 2 without memoization, with memoization over all samples, and memoization where the lookup table was cleared after each

sample. When we decoded the entire test set, we found much faster decoding since the previous samples had memoized a lot of the character sequences already, so we report a decrease in the mean sample running time of 50.32%. Surprisingly, the effect was only slightly less when we cleared the lookup table after each sample--we got a decrease in the mean sample running time of 48.45%.

4 Discussion

Overall, the inclusion of our language model was only able to bring modest improvements to test accuracy. Shi et al. (2019) found similar results with their LSTM-based language model, although attributed this to fingerspelling words being rare (as exemplified by their development set perplexity of 17.3). Our experiments cast considerable doubt on that hypothesis. First, by pretraining on a dataset of Wikipedia titles, our language model was indirectly able to see a significant amount of the test data (e.g., "Joe Biden", "Marlee Matlin", "circle") and thus was able to factor it into its predictions. However, this still was not enough for a significant improvement in results. Second, this explanation does not account for the decoder LSTM with iterative visual attention, which implicitly learns its own language model. The decoder LSTM has twice as many parameters as their external language model (and trained on the same data), which could help explain why the additional LM did not help much.

As for why our language model faced similar limitations, there are a few possible explanations. First, due to computational limitations, we only used a beam size of 2 to conduct experiments. This meant that the ASL translator only gave two options per frame for the language model to choose from, so even if the language model was much more accurate in scoring the predictions, the lack of predictions it had access to heavily limited its effectiveness. It is worth noting, however, that the implicit language model of the RNN decoder in Mahoudeau (2020)'s approach only improved by 0.51% (77.68 -> 78.19) when going from a beam size of 2 to 7. While we cannot speak to how predictive this would be with our GPT-2 character vocabulary language model, we recommend future researchers with either much more computational resources or time to run experiments explore increasing the beam size to improve results.

While the improvements over the baseline were certainly underwhelming, we do believe pretraining on English text still has merit for ASL recognition. ChicagoFSWild+'s entire training dataset was the equivalent of a single batch for our GPT-2 pretraining, which speaks to just how limited it is relative to the possible number of words expressible by fingerspelling. Given that new words are added all the time to ASL via this method, having a language model with broad access to English (the primary source of borrowing for ASL) should be enormously helpful in doing so. The fact that adding a lot more training data did not hurt performance at least confirms that this approach is not counterproductive.

Given that we did not do any optimizations aside from memoization, adding GPT-2 for language model scoring made inference far slower (from 0.3 seconds/sample to 3 seconds/sample, dropping to 1.4 seconds/sample post-memoization). Aside from using multiple GPUs (which due to parallelization should provide significant speed improvements), we expect that quantization, pruning, distillation, and other neural network compression models would likely help significantly in alleviating this issue.

5 Future Work

We hope to expand our project in upgrading ASL translation systems such that the Deaf community can take advantage of Google Translate-like apps in the same way that hearing individuals do.

With regards to language modeling, being able to use a Transformer decoder with iterative visual attention would probably be best. Iterative visual attention would first need to be updated to work with scaled-dot product attention before using such a model, however. One concern with this approach is that it would be very difficult to pretrain with a large English corpus given that iterative visual attention expects to do attention on images, not text. Without significant attempts at collecting large amounts of ASL-English "bitexts", an external language model will probably still be necessary to cover words not seen in the training set.

We would also recommend that any future language modeling training (whether by implicit decoder or external pretrained LM) train over both fingerspelling and standard ASL words. Some special notation (e.g. adding a '<F>' character) should be used to denote fingerspelling so that each letter can be recognized as an individual token. We recommend that unigram LM tokenization be used to create the vocabulary due to recent research showing it to beat the traditional BPE approach (Bostrom and Durrett, 2020).

If two different language models must be used for discerning fingerspelling and normally signed words, keeping a threshold in the beam decoder to differentiate between the two is one potential approach. If our beam decode function is run on the observed sequence and every hypothesis has a sufficiently low probability, then we can conclude that the observed sequence was of a typical signed word rather than a fingerspelled sequence. This approach can be used to choose which of your two dedicated language models should be used for scoring.

Finally, custom external language models should be trained for much longer than the 10 hours we trained ours for. Training on much more data in broader contexts (Reddit, Wikipedia articles instead of just titles, books, TV scripts, etc.) should be helpful for generalization purposes in both fingerspelling and standard ASL recognition, although our experiments cast doubt on it helping on data that the decoder has already seen. Given that we only used a 117M parameter model, we are hopeful that increasing the model size would lead to better results like has been seen in NLP research for the past few years.

6 Conclusion

In this work, we aimed to improve the accuracy of fingerspelling recognition for ASL by rescoring predictions with a GPT-style Transformer decoder language model. While we were able to make modest gains (0.13%), we had expected far more from upgrading from the baseline of an implicit 1-layer decoder RNN. These small improvements came at great cost in inference speed, meaning that future investigation is needed to both reduce the model's size and improve its accuracy by larger amounts.

7 Acknowledgements

We express our thanks to David Harwath for introducing us to much of the relevant literature needed to complete this project and for his wonderful lectures that we enjoyed throughout the semester, to the PyTorch and Huggingface communities for building the libraries we used to build our models, and to the crew at *Switched at Birth* for introducing us to the day-to-day lives of the Deaf community and for inspiring us to explore sign language recognition tools in the first place. Lastly, we have decided to make our code open source, which you can view at this link: [2]

8 Citations

Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High performance large-scale image recognition without normalization. arXiv preprint arXiv: 2102.06171, 2021.

Kaj Bostrom and Greg Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining arXiv preprint arXiv: 2004.03720, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.

Clement L. Canonne, Gautam Kamath, Audra McMillan, Jonathan Ullman, and Lydia Zakynthinou. Private Identity Testing for High-Dimensional Distributions. arXiv preprint arXiv 1905.11947, 2019.

Tianqi Chen, Bing Xu, Chiyuan Chang, and Carols Guestrin. Training deep nets with sublinear memory

[2] https://github.com/amitjoshi24/MiCT-RANet-ASL-FingerSpelling

cost. arXiv preprint arXiv:1604.06174, 2016.

Alex Graves, Santiago Fernandez Faustino Gomez, and Jurgen Schmidhuber. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets. In ICML'06: Proceedings of the 23rd International Conference on Machine Learning, pages 369–376.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad,. Mike Lewis. and Luke Zettlemoyer. 2020. Multilingual Denoising Pre-training for Neural Machine Translation URL https://arxiv.org/abs/2001.08210.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention- based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 (NIPS'2012).

Carol A. Padden and Darline C. Gunsals. How the alphabet came to be used in a sign language. Sign Language Studies, pages 10–33, 4 (1) 2003.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper.pdf, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 2019

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709, 2016.

Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Jonathan Michaux, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. American Sign Language fingerspelling recognition in the wild. In SLT, 2018

Bowen Shi, Aurora Martinez Del Rio, Jonathan Keane, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. Fingerspelling recognition in the wild with iterative visual attention. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5399–5408 (2019)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, and Mohammad Norouzi, "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv preprint arXiv:1609.08144, 2016.

Kayo Yin, Jesse Read. (2020). Sign Language Translation with Transformers. arXiv preprint arXiv:2004.00588v2, 2020.

Yizhou Zhou, Xiaoyan Sun, Zheng-Jun Zha, and Wenjun Zeng. Mict: Mixed 3d/2d convolutional tube for human action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 449–458, 2018.