GestureAIde:

A blend of 'gesture' and 'aid', indicating assistance in understanding sign language through AI.

Group 2: Paul Parks, Bin Lu, Eyoha Girma, Jeremy Cryer

Department of Applied Artificial Intelligence, University of San Diego

AAI-521: Intro to Computer Vision

Dr. Saeed Esmaili

12/11/2023

Authors' Note

Team 2 Github repository link: https://github.com/p-parks/AAI-521_FinalProject_Team2

Abstract

There are hundreds of millions of people in the world who have some sort of difficulty communicating due to hearing impairment. However, in comparison less than a million people actually know how to use ASL to communicate. To try and solve this issue, this team attempted to create a model that can translate in real-time ASL performance and convert it to text. This paper will go over all approaches attempted, including the final model and then will go over the few other approaches that were also attempted, just were not as accurate. The output performance is what leads the team to abandon those for reasons that will be explained. The team studied multiple different items to better their understanding of ASL and utilized those ideas to better evolve the model, that way limitations such as computing power were not as restricting with processing the large amount of data used.. At the end of this paper, the team will go over the challenges that were seen as possible restrictions to a better outcome and recommendations on how to get past them with a second iteration.

GestureAIde:

Introduction

In today's world AI is becoming a bigger tool to help with everyday tasks. One of those areas to help better assist people in, is in the world of American Sign Language (ASL). Now in this article it states, "According to the WHO (World Health Organization) report, over 466 million people are speech or hearing impaired, and 80% of them are semi-illiterate or illiterate [1]. Non-verbal manner conveys and communicates our views, emotions, and thoughts visually through sign language" (MADHIARASAN, 2022). That's a lot of people who use a language that is based on signs rather than syntax, wording, and even tone. The goal here is to create a model that can analyze those signs and display the meaning in text. This could help bridge the communication gap between many different communities.

Model Methodology

After reviewing the problem at hand and exploring what a solution means and entails, the group studied the data and researched for any possible model solutions. The research then led to a transformer model that caught the team's eye. This transformer model approach was designed by Wijkhuizen, M., in the Kaggle competition (2023). The project team decided to follow Wijkhuizen, M.'s approach to create a transformer model as one of the models to test for this project. The goal with this approach was to get a better understanding of the transformer model since Wijkhuizen, M.'s approach was to build a transformer model from scratch and not fine-turn a base model. Now as the methodology is further discussed for how this model approach was adopted, a deeper dive into the EDA, data preprocessing techniques and model architecture will be done to discuss how a model was designed to get the end results that were obtained. Along with the final solution, a handful of failed approaches will also be mentioned to give better insight on the journey that was taken and what didn't work and why.

EDA and Preprocessing Techniques

With this problem and the data used, there are multiple approaches to the methodology to get an ASL translating model. However, the approach for EDA and preprocessing is very strict. If not done correctly, regardless of how well the model is designed, it will perform poorly.

Now, first things first, having a training set to build the model around is key. A dataset was identified on Kaggle[1] that the team utilized, which contains multiple parquet files of various image frame data of ASL signs and a library with the corresponding meaning (Ashley, 2023). Now the goal of the EDA and preprocessing is to be able to upload the parquet files and get them into a format to use for the training and validation of the model. Now most of this is all embedded into the model architecture, but for the transformer model approach, it was identified to only use landmark points from hands, lips, and arm pose. The team found out there were multiple benefits to doing this regarding three different topics: clarity and context from lip movement, dominant hand significance, and computational efficiency. The next three subsections will briefly go over their importance in using only landmark points in the preprocessing. Another key item before this is expanded on was the criticality of sequence padding with the frame count.

In Wijkhuizen, M.'s transformer approach, all the data has been reformatted to 4D tensor (Number, Frame, KeyPoint, LandMark), where "Number" is the number of data file which also link to a label y for the meaning of signs. "Frame" is the frame of the video recording; if the frame is larger than 64, it will be downsampled, and if the frame is shorter than 64 frames, then it will be padded (2023). "KeyPoint" is the LandMark keypoint from the Mediapipe tracking result; this approach has limited only 66 key points: Lips has 40 key points, the dominant hand has 21 key points, and the dominant side pose has 5 key points, a total of 66 key points. The "LandMark" is 3 LandMark values of [x, y, z] from MediaPipe tracking. This has significantly reduced the size of data that need to be processed by the model and kept the most possible features that are important to ASL recognition. However, even with these approaches and the reasoning behind it, they weren't the only ones attempted. There were failed approaches that helped identify these realizations and gave better insight to how crucial this part of modeling actually is. More insight into the failed approaches in preprocessing will be further discussed in the "Failed Approaches" section later in this report.

[1] https://kaggle.com/

Clarity and Context from Lip Movement

In sign language, lip patterns are crucial as they provide additional context and clarity to the signs being made. Lip movements can change the meaning completely. Therefore, accurately tracking lip movements can significantly enhance the accuracy of sign language recognition. For example, in American Sign Language (ASL), the signs for “mother” and “father” differ only in lip shape. Mouth morphemes are also non-manual markers that convey additional grammatical information. “A well-known difference between two signs that contrast in non-manual sign (NMS) is LATE and NOT-YET ("TH" mouth morpheme)” (Handspeak., n.d.). Accurate lip tracking helps capture these nuances.

Dominant Hand Significance

In sign languages, the dominant hand plays a primary role in forming signs. It is typically more active and precise in its movements compared to the non-dominant hand. Focusing on the dominant hand helps accurately capture the nuances of sign language gestures. As an article in What's The Sign shows, consistently using a dominant hand is crucial for clear communication in sign language. Switching hand dominance while signing can be distracting, similar to how a strong accent might affect spoken language comprehension. Although Deaf individuals may still understand the signs, consistent use of a dominant hand ensures clearer communication (n.d.).

Computational Efficiency

Tracking all points provided by a tool like MediaPipe, which includes detailed mapping of the entire body, can be computationally intensive and may not always add significant value to sign language recognition. By focusing on key elements like the lips, dominant hand, and body pose, the system can be more efficient and faster, making it more practical for real-time applications. This transformer approach reduces tracking 486 full landmarks 468 to just 40 for the lips.

Model Architecture

To solve this model, the end decision for the team was to use a transformer model similar to Wijkhuizen's approach. Wijkhuizen, M. 's transformer model approach is a custom-modified transformer model similar to the classic transformer model. There are custom changes to fit the ASL recognition tasks. This custom transformer only used a classic transformer encoder part for classification tasks, which is a normal approach to only use the transformer encoder or decoder part alone. Besides the architecture part, there are two major customizations that Wijkhuizen, M. created that differ from a classic transformer model; the attention mechanism and the embedding layer.

Attention Mechanism

The attention mechanism is a key component in Transformer models, enabling the model to focus on different parts of the input sequence for each step of the output sequence. Attention enables the model to concentrate selectively on various segments of the input sequence for making predictions, rather than interpreting the entire sequence as a uniform-length vector. This feature has been crucial in the triumph of the transformer model, sparking extensive subsequent research and the development of numerous new models (Kumar, A. 2023). Wijkhuizen, M.'s custom transformer model deployed attention_mask in the Scaled Dot-Product function in a different way compared to the classic transformer model in that this mask is applied in the Softmax step to selectively ignore or pay less attention to certain parts of the input, such as padding or irrelevant frames in a video sequence. Also, the Softmax layer was used instead of the Softmax function. The attention mechanism allows the model to focus on different parts of the input sequence dynamically, which is crucial for tasks like ASL recognition. In ASL, the importance of different landmarks can vary significantly across different signs. The multi-head attention mechanism is particularly well-suited to capture these varied dependencies.

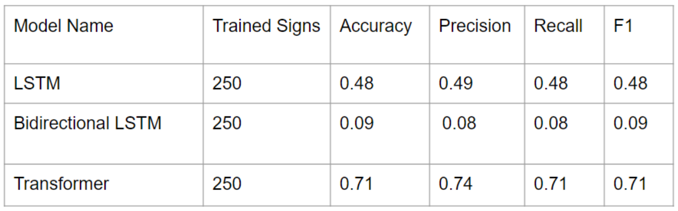
Embedded Layer

Another customized part is the embedding layer. In the context of Transformer models, positional embeddings are crucial for providing information about the order or position of the elements in the input sequence (Huang, Z. et al., 2020). Compared to the classic transformer model, this ASL transformer model approach has an additional LandmarkEmbedding class that the Embedding class uses to deal with the embedding of individual landmarks. Each landmark type was embedded separately from lips, hand, and pose. The Embedding class is still handling the positional embedding.

Results/Evaluation

The overall ASL Transformer designed by Wijkhuizen, M. is shown in APPENDIX 1. After training and evaluation, the model performance is shown in APPENDIX 2. Wijkhuizen, M. 's transformer model has an overall 0.71 F1 score with a weighted precision of 0.74 and a weighted recall of 0.71. It has outperformed any other model types that was attempted. Some ASL word predictions perform better than others; for example, airplane, apple, owl etc., have F1 scores higher than 0.90. and other words like kitty, yucky, and suffer under the F1 score lower than 0.40. However, most words' F1 scores are higher than 0.60, so we could use this model for a real-life application with some limitations. Now three overall models were attempted with comparison accuracy metrics shown in table 1 below. The table highlights the main reason the team moved to the final model over the other two, due to the results given during evaluation. However, the other two models will be briefly discussed in the next section.

Figure 1



Model Name	Trained Signs	Accuracy	Precision	Recall	F1
LSTM	250	0.48	0.49	0.48	0.48
Bidirectional LSTM	250	0.09	0.08	0.08	0.09
Transformer	250	0.71	0.74	0.71	0.71

Note: Comparison table for metrics on all three models attempted.

Failed Approaches

With all model design there are multiple approaches that are looked at with some that work, some that are better than others, and some that don’t work at all. Now with this problem, earlier the final solution was discussed but not the approaches attempted to lead to that end path. Now the failures are important because they give better insight into the journey and the learnings faced that assisted in the decisions made in the design.

EDA That Did Not Work

For instance, with the preprocessing of the data, due to the shear volume of parquet files used, if not done correctly, when importing all the files, it's possible that only the file as a whole average could be imported instead of each frame instance which will go from the 10-64 data points to a single input per frame. Now it is possible to miss it from first glance just due to the sheer volume of data regardless of approach. Now if this happens and then a basic LTSM model is used just to test the data, you will get almost 0% accuracy from your results. This is due to the multiple data points per frame turning into one. With an average you cannot replicate the image with the coordinates to get the sign intended, which makes it not possible to train to as essentially you will have a single point in space acting as a full sign instead of the multiple points in space, shaping the actual sign. This just shows how critical preprocessing is for this situation and for modeling in general.

LSTM Approaches

Other models were also attempted where architectures like GRU and Bi-directional LSTM were utilized, aiming to capture the data's dynamics more comprehensively. These alternatives were considered to enhance the model's ability to understand the sequential patterns within the sign language gestures more effectively.

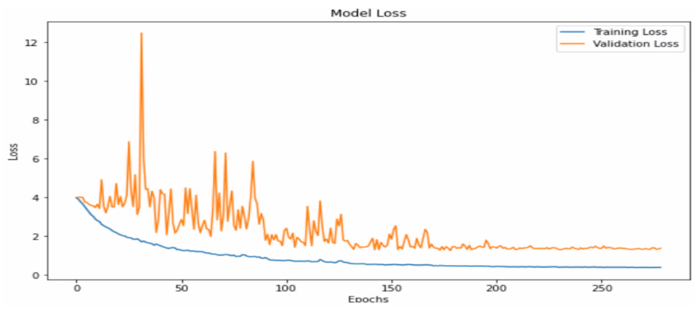
Main LTSM Attempt. The first approach to solve this problem involved using a LTSM model which is common for sequence modeling. In the early attempts, difficulties were faced with accurately representing the ASL data. The model needed to comprehend the fluid and dynamic nature of sign language, where nuances in movement are crucial. The LSTM's inability to capture these subtle movements led to a significant loss in translation from gesture to meaning. This led to the model displaying promising accuracy during training but faltered significantly in validation. This discrepancy was a classic case of overfitting, where the model memorized the training data but failed to generalize its learnings to new, unseen data.Our dataset had inherent class imbalances, which skewed the learning process. Despite efforts to counter this with class weights, the LSTM model struggled to learn equally from all categories of signs, leading to biased predictions. This was partially due to setting a frame cap of ten which was limiting to signs using larger frame counts. Now if you refer below to table 2 or figure 1 you see these results and where they falter.

Table 2

Model Name	Trained Signs	Accuracy	Precision	Recall	F1
All Signs	250	0.48	0.49	0.48	0.48
Removed Low Performing Signs	65	0.76	0.77	0.76	0.76
Top 15	15	0.88	0.89	0.88	0.88

Note: Performance breakdown for Main LTSM model.

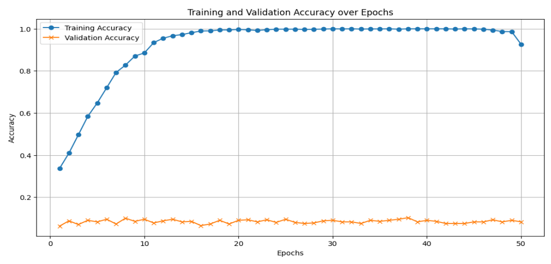
Figure 1



Note: Loss vs epoch for LTSM results.

Alternative LTSM Approaches. *I*n pursuit of a more effective solution, LTSM models like bi-directional or gated recurrent unit style architectures were also experimented with. Aiming to capture the dual nature of sign language gestures, a Bi-directional LSTM was attempted. While this model provided a more nuanced understanding of sequential data, it still fell short in accurately predicting the wide array of ASL signs, partly due to its complex structure leading to longer training times and computational inefficiency.Then with the GRU model, it was found to be a more efficient alternative to LSTM. However, it too struggled with similar issues of data representation and overfitting, failing to capture the intricate details necessary for accurate ASL recognition. However, the BiLTSM model was the third model closest to having any results. The model showed promise during training, with an initial rise in accuracy, indicating its capability to learn from the temporal features (Appendix 3A). This positive aspect was a testament to the effectiveness of the preprocessing steps, particularly in the creation of temporal features and sequence padding. However, the model's performance on the validation set painted a different picture. The low validation accuracy, which did not show significant improvement, pointed towards issues such as overfitting as can be seen in figure 2. This discrepancy between training and validation performance suggested that while the model could learn the training data, it struggled to generalize these learnings to new, unseen data. This plateau in model performance hinted at possible limitations in the model's capacity, the feature representation, or the dataset itself. The persistent low validation accuracy was indicative of overfitting, a common challenge in machine learning where the model learns the specifics of the training data too well but fails to apply this learning effectively to new data.

Figure 2



Note: BiLTSM Model Results.

Learnings and Recommendation

The following section will go over challenges faced during this project and recommended future improvements for the next iteration continued from this first attempt. Now, this is important to understand because not every model will be perfect on the first approach but knowing how to do better on the second is what's important.

Handling Large Dataset Challenges

A significant obstacle encountered in the development of GestureAIde was the management of an extensive dataset exceeding 50 gigabytes. This dataset's size presented logistical challenges, particularly in uploading and processing within cloud-based platforms such as Google Colab. The sheer volume of data necessitated a more efficient approach to mitigate bandwidth and storage issues. To address this, two solutions were implemented. First, Google Drive was used to upload the dataset once, avoiding repetitive data transfers. Secondly, a local PC with a high-performance GPU was utilized to handle the computational demands. This approach streamlined the data handling process and optimized the computational efficiency required for processing such a large dataset.

Challenges with Inference and Testing with Real-Time Data

Another challenge faced was in the realm of real-time data processing during the inference and testing phases. Utilizing OpenCV for this purpose, especially within the Google Colab environment, revealed significant performance limitations. The primary issue was the integration of JavaScript to enable webcam access, which led to a drastically reduced frame processing rate, hovering around one frame per second. This severely hampered the model's ability to function effectively in real-time applications. The testing was transitioned to a local computing environment to overcome this hurdle. Running the model locally on a machine equipped with an adequate GPU resolved these issues, substantially improving the speed and effectiveness of GestureAIde's real-time data processing capabilities.

Future Improvements

Now with this project there were several failed approaches and challenges faced when trying to design a final working model. Often it came down to computing power and preprocessing the data. Now due to that if someone were to build off of this model and perform a second attempt, the advice that would be available would be first, if limitations were not desired, to get as much computing power as possible. Due to the restrictions existing for this team, this did affect the capability of the modeling attempted. Limited computing led to limiting what data was used such as the 64 frame max set, which limited the viewed data to key landmarks. Now adding those limited portions of data may improve accuracy on some of the words seen to perform at a lower metric as there may be key features that better differentiate the sign to a similar one. Now also finding a way to train all signs without limiting or padding the frame count would also be helpful. Right now the model is padding the frame count if there is not enough, so if there are ten frames, 54 blank frames would be added. Now an analysis was not done to see commonality between low performing words and sequence count, but adding blank data could be altering the training of certain words that could be impacting this. Being able to accept the frame count as is without padding or limiting would be key.

Conclusion

With this project the team was successfully able to create a model that could identify various signs trained to. There were several approaches with a transformer model identified as the best approach to follow. Now throughout this paper, the approaches for all attempts were discussed with the methodology of the transformer model being the main focus. Now with all modeling, some challenges were also faced such as computing power restrictions with the data loading and javascripting. Now this was discussed in the paper with some recommendations for future attempts given. Hopefully this paper helps give a basis for anyone to review and start where this team left off with creating a full proof model to translate ASL.

References

Ashley Chow, Glenn Cameron, Mark Sherwood, Phil Culliton, Sam Sepah, Sohier Dane, Thad Starner. (2023). Google - Isolated Sign Language Recognition. Kaggle. https://kaggle.com/competitions/asl-signs

Handspeak. (n.d.). Minimal pairs in sign language (ASL). From https://www.handspeak.com/learn/109/

Huang, Z., Liang, D., Xu, P., & Xiang, B. (2020). Improve Transformer Models with Better Relative Position Embeddings. arXiv preprint arXiv:2009.13658

Kumar, A. (2023, February 1). The transformer model and its applications: Understanding attention mechanism. Medium. Retrieved from https://medium.com/@mittal.atul06/the-transformer-model-and-its-applications-understanding-attention-mechanism-c37e6e3a76dc

MADHIARASAN, M. M., & ROY, P. P. (2022, April). *A comprehensive review of sign language recognition: Different types, modalities, and datasets*. ar5iv. https://ar5iv.labs.arxiv.org/html/2204.03328

What's The Sign. (n.d.). ASL: Which Hand Do I Use? Unveiling the Secret. Retrieved [date you accessed the article], from https://www.whatsthesign.com/asl-which-hand-do-i-use

Wijkhuizen, M. (2023, April 04). GISLR TF Data Processing & Transformer Training. Kaggle. https://www.kaggle.com/code/markwijkhuizen/gislr-tf-data-processing-transformer-training