Word Level Translation of American Sign Language using Video Vision Transformers

JONATHAN FERRARI, University of California, Berkeley, USA

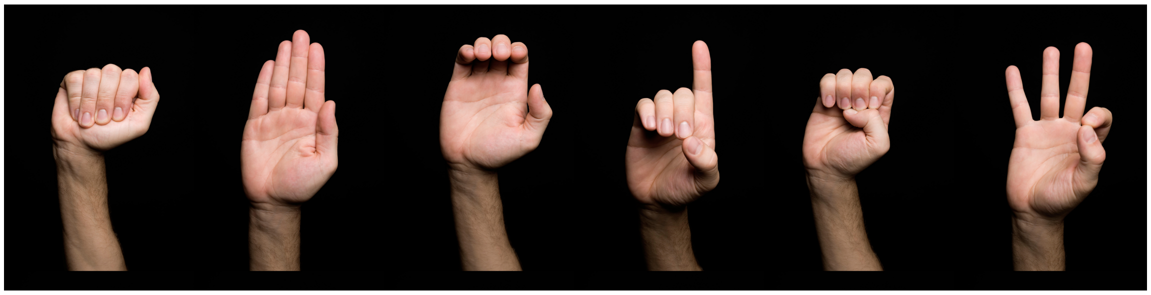

Fig. 1. American Sign Language Signs

In this research, I present a novel approach to enhancing digital inclusivity for the deaf and hard-of-hearing communities through the development of an advanced real-time American Sign Language (ASL) translation tool. Leveraging cutting-edge computer vision technologies and a newly created, comprehensive ASL dataset, our work addresses the critical limitations of current translation tools, which often neglect the cultural and contextual richness of ASL. By integrating diverse video data and signers from various backgrounds, we ensure our model's sensitivity to the nuanced, dynamic nature of sign language. Our results demonstrate significant improvements in the accuracy and inclusivity of ASL translations, paving the way for more effective communication technologies that cater to the needs of ASL users. This paper details the methodologies, challenges, and breakthroughs encountered in creating a translation tool that not only bridges communication gaps but also respects the cultural integrity of the ASL community.

ACM Reference Format:
Jonathan Ferrari. 2025. Word Level Translation of American Sign Language using Video Vision Transformers. 1, 1 (February 2025), 23 pages.

1 INTRODUCTION

In the journey toward digital inclusivity, this research stands at the intersection of technology and linguistics, offering a groundbreaking contribution through the development of an advanced real-time American Sign Language (ASL) translation tool. ASL, a rich and complex visual-gestural language, serves as the primary mode of communication for many in the deaf and hard-of-hearing communities across the United States and Canada. Unlike spoken languages, ASL employs movements of the hands and face to convey meaning, boasting its own unique grammar that diverges significantly from English. Its origins, deeply rooted in the early 19th century at the American School for the Deaf in Hartford, Connecticut, reflect a blend of local sign languages and French Sign Language (LSF), evolving over time into the distinct and comprehensive language it is today. ASL is not only a means of communication but also a pivotal

Author's address: Jonathan Ferrari, jonathanferrari@berkeley.edu, University of California, Berkeley, California, USA.

element of cultural identity, with variations in expression influenced by regional, age, and gender factors, similar to the dialects and accents found in spoken languages.

To address the challenges and limitations of existing ASL translation tools, this research leverages state-of-the-art computer vision technologies, including the innovative use of Visual Transformers. Visual Transformers represent a cutting-edge approach in machine learning, particularly adept at handling complex visual data by segmenting images into a series of components or 'tokens' and analyzing these both individually and in their broader context. This method's application to ASL translation is pioneering, offering the potential to capture the nuanced and dynamic aspects of sign language more accurately than ever before. By integrating a comprehensive ASL dataset and employing Visual Transformers, this research aims to bridge communication gaps more effectively, ensuring the translation tool is sensitive to the cultural and contextual richness of ASL.

Through meticulous methodology, this paper elucidates the process of developing a translation tool that promises not only to enhance communication for ASL users but also to uphold and respect the cultural integrity of the ASL community. The endeavor to improve ASL translation technology reflects a broader commitment to fostering a more inclusive society, where the deaf and hard of hearing can communicate seamlessly and without compromise. The implications of this research extend beyond practical communication tools, touching on the importance of linguistic diversity, cultural recognition, and the potential for technology to enrich human connection.

2 LITERATURE REVIEW

2.1 American Sign Language

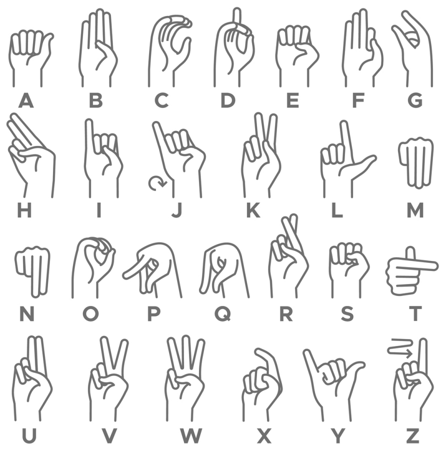

Fig. 2. ASL Alphabet Finger Spelling[15]

American Sign Language (ASL) stands as a significant component of the cultural and linguistic tapestry of the Deaf community in the United States and parts of Canada. Recognized not merely as a means of communication, ASL is an intricate language with its own syntax, grammar, and nuances, reflecting the rich cultural heritage of its users. The emergence and evolution of ASL, as detailed by Kusters and Lucas, offer a profound insight into the sociolinguistic dynamics that underpin the language's development and its role within Deaf culture[9]. These authors underscore the importance of understanding ASL's complexities and the sociocultural contexts from which it arises, advocating for a comprehensive approach to studying sign languages that encompasses both linguistic and sociolinguistic dimensions[9].

Furthermore, the historical context and foundational aspects of ASL are crucial for comprehending its significance and widespread adoption. Originating in the early 19th century with the establishment of the first school for the Deaf in the United States, ASL has since evolved to become a central mode of communication for the Deaf community, facilitating not only everyday communication but also the expression of identity and belonging[1]. This evolution highlights the adaptability and resilience of ASL, characteristics that underscore its role in fostering community and inclusion among its users. The Encyclopedia Britannica provides an overview of ASL, acknowledging its status as a fully developed language that supports a wide range of communication needs, from the personal and educational to the professional[1]. This recognition further emphasizes the importance of initiatives aimed at bridging the gap between the Deaf community and the hearing world, such as the development of technologies that translate ASL to spoken languages and vice versa.

In this context, the application of transformer models to recognize and translate ASL from video feeds into English text represents a groundbreaking step toward enhancing communication accessibility. By leveraging the power of advanced machine learning techniques, I hope to create tools that not only facilitate real-time translation of ASL but also deepen our understanding of its linguistic features. This endeavor not only has the potential to significantly improve the lives of Deaf and hard-of-hearing individuals by providing them with greater access to information and services but also contributes to the broader goal of promoting inclusivity and understanding across linguistic and cultural barriers.

2.2 Computer Vision

Computer vision is a multifaceted field of artificial intelligence that aims to mimic the capabilities of the human visual system, allowing machines to perceive, understand, and interpret visual data from the world around them. At its core, computer vision seeks to translate the vast amounts of visual information available in our environment into a format that computers can process and analyze, enabling a wide range of applications, from automated image labeling to complex scene understanding[4].

The journey of computer vision began with simple tasks such as edge detection and has evolved to tackle more complex challenges like object recognition, image segmentation, and motion analysis[4]. The advent of deep learning, particularly the development of convolutional neural networks (CNNs), has been a pivotal breakthrough in the field, significantly enhancing the accuracy and efficiency of visual data interpretation. These networks mimic the way the human brain processes visual information, leading to remarkable advancements in tasks such as facial recognition, autonomous vehicle navigation, and even medical image analysis[13].

As the field progresses, the integration of computer vision with other AI technologies, like natural language processing and machine learning, opens new avenues for innovation. This interdisciplinary approach enables the development of sophisticated systems that can not only see but also understand and interact with their surroundings in meaningful ways. For instance, computer vision techniques are being applied in augmented reality (AR) to create immersive user experiences, and in robotics, to enable machines to navigate and manipulate objects in their environment autonomously[3].

The potential applications of computer vision are vast, impacting numerous sectors including security, healthcare, automotive, and entertainment. As technology continues to advance, the scope of computer vision is expected to expand further, leading to more intelligent systems capable of performing complex visual tasks with high levels of accuracy and autonomy. This ongoing evolution underscores the importance of computer vision in the modern technological landscape, marking it as a key area of research and development in artificial intelligence.

2.3 Transformers and Previous Work

Transformers, which revolutionized the field of natural language processing (NLP), have since been applied to the realm of computer vision, offering innovative approaches for visual recognition and analysis. Initially designed for understanding sequences in text, transformers' architecture has been readily adapted for visual tasks, introducing models capable of handling the intricacies of image and video data with great efficiency. The pivotal work by Li et al. introduces Contextual Transformer Networks, a novel approach that leverages the contextual relationships within visual data, enhancing recognition tasks across a spectrum of applications[12]. This adaptation signifies a shift from traditional convolutional neural networks (CNNs) towards more flexible and powerful transformer-based models for visual recognition.

The evolution of transformers in visual tasks has led to the development of Video Visual Transformers (VVTs) and various other transformer variants tailored for specific applications, including sign language recognition. For instance, research by Zuo, et al. explores the intersection of natural language processing and sign language recognition, demonstrating the potential of transformers to bridge communication gaps for the Deaf community[20]. Furthermore, a comprehensive survey by Liu et al. provides an in-depth overview of visual transformers, highlighting their application in tasks ranging from image classification to complex video analysis, marking a significant advancement in the field[14].

The revolutionary impact of transformers is further exemplified in the domain of American Sign Language (ASL) recognition and translation. The How2Sign dataset, as discussed by Duarte et al., represents a significant step forward, offering a large-scale multimodal resource that facilitates the development of sophisticated ASL recognition systems[7]. Such resources are crucial for training and evaluating transformer-based models, ensuring their effectiveness and applicability in real-world scenarios. Innovations in this area not only contribute to the technological empowerment of the Deaf community but also offer insights into the adaptability of transformer models for interpreting the nuanced gestures and expressions inherent in sign language.

Moreover, the exploration of Deep Vision Transformers (DeepViT) and Temporally Efficient Vision Transformers for video instance segmentation underscores the ongoing efforts to refine and enhance transformer models for video data[19][18]. These variants illustrate the potential for transformers to not only recognize static images but also to interpret dynamic sequences, a capability of immense value in video-based applications, including sign language translation.

The advent of transformers and their application to computer vision tasks represent a paradigm shift in how machines understand and interpret visual information. The development of Video Visual Transformers and related variants has opened new avenues for research and application, particularly in areas requiring a nuanced understanding of visual sequences, such as sign language recognition[16]. As these technologies continue to evolve, they hold the promise of bridging communication barriers and enhancing accessibility, marking a significant milestone in the journey towards more inclusive and empowered digital communities.

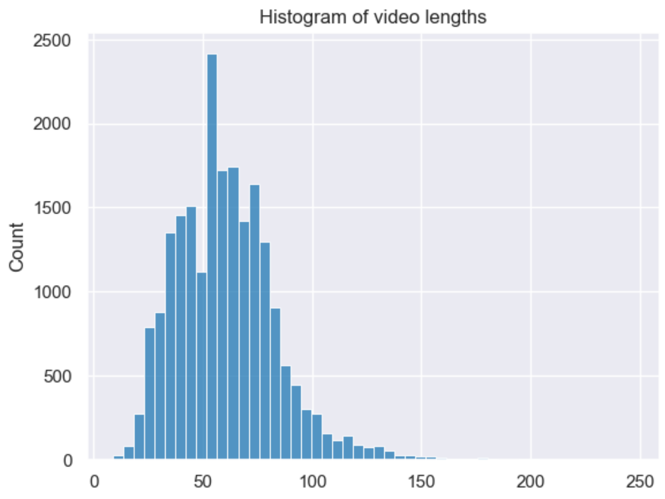

Fig. 3. Histogram of Video Lengths

3 METHODOLOGY

3.1 Data

All of the data used in this report are of the same form: videos represented as tensors. For video i (V_i), let C_i be the number of color channels, H_i be the height of each frame of video (pixels), W_i be the width of each frame of video (pixels), and F_i be the number of frames per video.

Thus, we can describe the shape of V_i as follows:

$$\text{shape}(V_i) = \mathbb{R}^{H_i \times W_i \times C_i \times F_i}$$

In particular, for the data I am working with, $\forall i,\ H_i = W_i = 256, C_i = 3$. There are a total of 21,083 unique videos with 2,000 different classes, where each class is one unique sign/word.

The number of frames varies across the different videos. The distribution of video frame lengths is as follows:

Table 1. Summary Statistics of Length

Measure	**Value**
Count	21083
Mean	60.45
Standard Deviation	23.16
Minimum	9
25th Percentile	43
Median (50th Percentile)	58
75th Percentile	74
Maximum	247

Each video is of one individual performing the ASL gesture of a word or a phrase widely accepted as part of the official language. One such example can be seen below, where a signer makes the work *book*. Clicking the image will bring you to the full video.

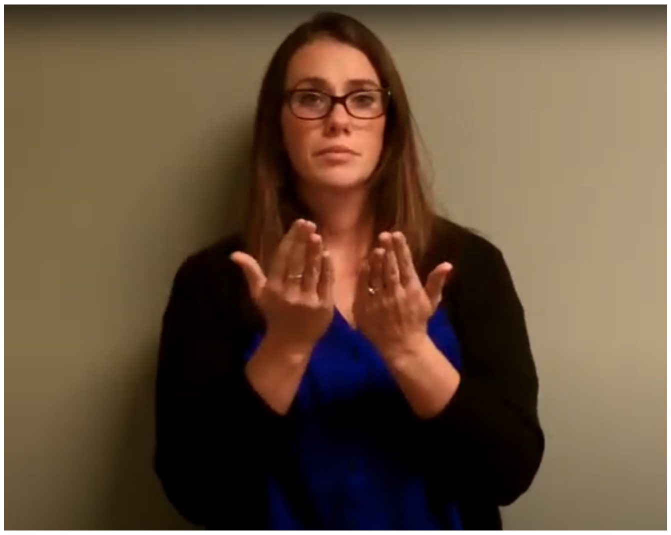

Fig. 4. Sign: Book

For this proof of concept, I have chosen to start with a small vocabulary, with large numbers of examples per class, to ensure accuracy over scalability.

3.2 Data Collection

Data has been collected from WLASL[11]. The initial dataset was scraped and collected over the course of a week using the process outline in the WLASL repository. The WLASL is a large dataset of videos from different signers, containing all of the videos I used in this report.

The dataset is a few years old, so unfortunately some of the links to the original videos no longer work. Fortunately, I was able to acquire the rest of the original data, with the help of one of the authors of the WLASL paper, Dongxu Li, who was able to send me a zip file with all of the videos which I was unable to scrape/download.

3.3 Data Preparation

To fit a model to the data, the videos all need to be of the same shape, size, and format. To accomplish this there are many necessary steps. Namely, there are a few important transformations.

First of all, the data has been be reduced to monochromatic video; this means that all of the channels (e.g. R, G, B), have been reduced to one (e.g. gray-scale). This effectively ensured that $\forall i : C_i = 1$. This transformation was taken care of by using the built-in method `cv2.cvtColor`, which easily allowed me to change the color space of each frame as it was read.

After the monochromatic transformation was applied, all of the data was scaled. Generally in video data, a pixel value at row i and column j can take on values

$$P_{ij} \in \{x : x \in \mathbb{N} \wedge 0 \leq x \leq 255\}$$

where 0 implies the pixel is pitch black, and 255 implies it is pure white.

In the data preparation, the values were scaled by dividing each pixel by 255, to ensure that pixel values are normalized such that all pixel values fall from 0 to 1. More formally, the new values P_{ij} satisfy

$$P_{ij} \in \{x : x \in \mathbb{R} \wedge 0 \leq x \leq 1\}$$

where 0 is pitch black, and 1 is pure white.

Next, I ensured that all data has the same dimensions for height and width. This ensured

$$\forall i, j \quad C_i = C_j \wedge H_i = H_j$$

I attempted to scale the images to 256x256, however, this proved to be much too computationally intensive. Instead, I iteratively reduced the resolution until it was feasible. I ended up reducing the resolution to 64x64, which created 4,096 pixels per frame rather than 65,536 with the original dimensions. I accomplished this by using Bi-linear interpolation to reduce the images to my desired size before eventually stitching them back together into a tensor which represents the video. This allowed the model to create far fewer weights for classification, which was thus more computationally tractable.

Finally, for the F dimension, I scaled each video in the temporal dimension, by ensuring each video only has 20 frames.

I chose 20 frames for a few reasons. First, after some ad-hoc analysis, I found that less than 1% of the videos had fewer than 20 frames. This meant that I would not have to deal with the edge case of less than 20 frames very much. It also ensured that my data was not too complex, in that there were only 81,920 total pixels per video, rather than a higher temporal resolution leading to greater complexity.

In the case where the video had less than 20 frames, I added all black frames to the beginning and the end until the video reached 20 frames in length. This is done through an iterative process using a `while` loop, and at each iteration, add a frame to the end, check if we are at the desired length, then, add a frame to the front, until we reach 20 frames.

In the case of the video being longer than 20 frames, which is far more common, this is when temporal sampling was used. This was done using the `np.linspace` function to create evenly spaced jumps between frames.

Finally, I expanded the dimensions of the video, so that it would still have a channel dimension, which is the common practice in computer vision.

Thus, after all of the transformations, each video (V_i) satisfied the following:

$$H_i = W_i = 64, C_i = 1, F_i = 20$$

This is a form that is necessary for the model, as each video needs to be the same shape, and beneficial so that the model is not overly complex and intractable.

3.4 Synthetic Data

One of the limitations of using computer vision for such a task is you need a large number of examples per class for the model to be able to learn the features used in prediction. Specifically for classifying hand signs, it is hard to find a large number of each sign repeated by multiple signers.

Below is the distribution of the number of examples per class:

Table 2. Summary Statistics of Length

Measure	**Value**
Count	2000
Mean	10.54
Standard Deviation	3.55
Minimum	6
25th Percentile	8
Median (50th Percentile)	10
75th Percentile	12
Maximum	40

As we can see in the histogram below and the table above, there isn't much data per class. This introduces an issue in how we build our classifier, as we may lack enough for the model to learn enough complexity from such a small example.

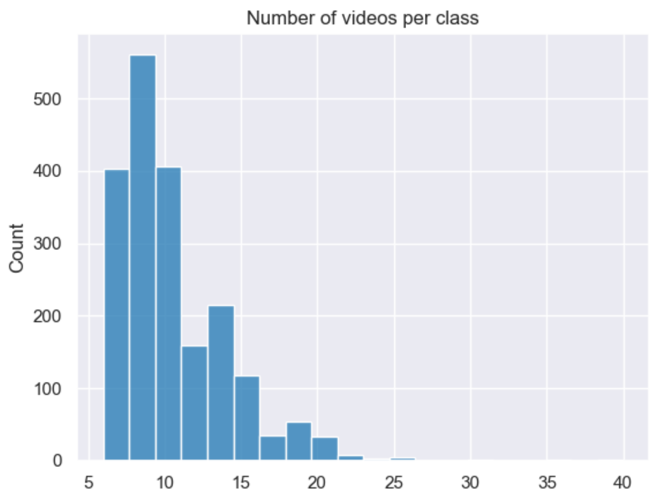

Fig. 5. Class Counts

Hence, we need a way to "create" more data, as unfortunately for this research, funding was not available to create a more comprehensive dataset.

This is where we use synthetic data, the practice of creating “artificial” data from real-world data which is still useful in training a model. For this research, I implemented a few different methods of generating synthetic data, which is outlined below. I used methods pertaining to flips/rotations, blurs/noising, and spatial/temporal cropping. These methods are outlined below.

The `rotate_video` function manipulates the orientation of a video by rotating it by a specified angle, typically 90 or 270 degrees. This adjustment is beneficial for generating synthetic data that simulates different viewing angles, thus aiding in the development of models that recognize objects from various perspectives, for instance, if the camera is angled, or viewing a signer from different angles.

The `flip_video` function enhances dataset variability by flipping video frames horizontally, vertically, or both. This method is straightforward yet effective for augmenting data, especially in scenarios where increased orientation variance is needed, for similar reasons as `rotate_video`.

In the `crop_video` function, video frames are cropped to a specified region, allowing focus on particular areas of interest. This is particularly useful as by using random crops, we are allowing the model to not over-train on certain backgrounds or positioning of the hands, such as having the hands always be in the center of the video.

The `noise_video` function introduces random noise and brightness variations to video frames, simulating real-world imperfections such as sensor noise and lighting fluctuations. This method prepares the model to perform well even under suboptimal conditions, enhancing its robustness.

Using the `blur_video` function, Gaussian blur is applied to video frames to simulate out-of-focus effects or to desensitize data to minor variations. This approach helps in training the model to focus on significant features such as arms or hands, rather than overfitting to noise.

Lastly, the `temporal_crop_video` function randomly omits frames based on a specified probability, effectively simulating missing data or variable frame rates. This technique is crucial for training video processing models to handle irregular inputs efficiently, ensuring robust performance in real-world applications.

Each of these methods plays a significant role in creating a diversified and challenging dataset, essential for developing advanced machine learning models.

By using these methods, I was able to generate a large amount of usable data to train and test the model on. For the proof of concept, I train on 1,400 videos of shape $20 \times 64 \times 64$, as opposed to roughly only 100-200 examples, thus allowing the model to both learn more complex patterns and be trained on messy data, so that it performs well in the real world, not just under simulation with clean data.

3.5 Splitting Data and Handling

I randomly split the data to create a training and test set for the data. To ensure that there is enough data for classification, I completed a stratified split of the data. This means that for each class Y_i, I split it such that 80% of the data for each class is in the training set, and 20% is in the test set. The overall training set is thus:

$$\text{Test} = \bigcup_{i=1}^{n} Y_i^{\text{Test}}$$

The analogous case is true for the training set. This will ensure that there are no examples of classes in the test set that have only been trained on a few examples of the class.

For handling the data, I have defined a `VideoDataGenerator` class, which will allow me to not only batch the data but efficiently process the data.

The `VideoDataGenerator` class is a custom implementation that extends the `Sequence` class from Keras, designed to efficiently manage video data loading and preprocessing for training deep learning models on video classification tasks. This generator handles the loading of video files, conversion to grayscale (if needed), resizing to a uniform frame size, and normalizing the number of frames per video. By inheriting from `Sequence`, it ensures that the data generation is thread-safe, which is critical when using multi-threading in training to avoid data consistency issues. This class effectively handles all necessary data transformations mentioned earlier.

The class is particularly beneficial for handling video, as it can load in batches, meaning the files don't all need to be stored in memory at once, which significantly reduces the memory overhead needed for processing a large number of videos which each are storage intensive.

3.6 Model Architecture

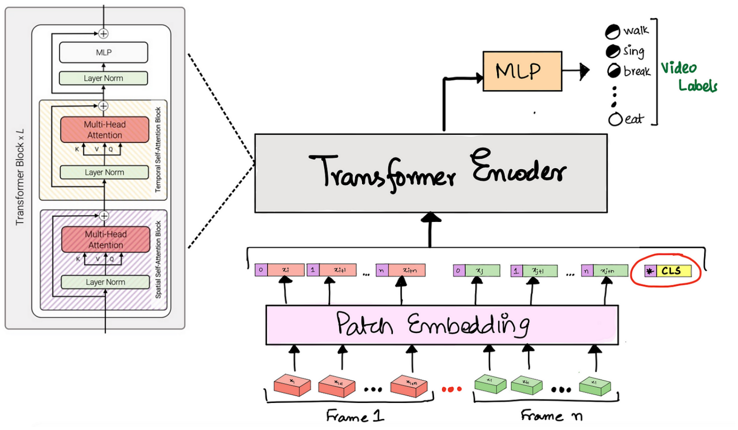

Fig. 6. ViVit Model Architecture[8]

The model that I am using to classify these videos is a Video Vision Transformer (ViViT). The ViViT model, introduced in a research paper by Arnab, et al., represents a significant advancement in video processing deep learning architectures[2]. ViViT extends the Vision Transformer (ViT) architecture, which was originally designed for image processing, to handle video inputs by incorporating spatial and temporal information effectively. Unlike traditional architectures that rely heavily on convolutions, the ViViT model separates the spatial and temporal components of videos, processing them through distinct transformer mechanisms. This allows ViViT to better learn the intricate dynamics and spatial details within video data. Developed to address the inefficiencies of convolutional approaches in capturing long-range dependencies and complex temporal patterns in videos, ViViT leverages the self-attention mechanism of transformers, providing a more flexible and comprehensive approach to understanding video content[17].

ViViT offers several advantages over traditional 3D Convolutional Neural Networks (3D CNNs), which have been the standard for video and volumetric data processing. 3D CNNs extend the 2D convolutions by adding an additional temporal dimension to the kernels, allowing them to extract features from both space and time dimensions simultaneously[10][5]. However, this approach can lead to significantly increased computational costs and model parameters, as it involves convolving through three dimensions. ViViT, by contrast, utilizes the transformer's ability to handle sequences, applying self-attention mechanisms that capture relationships within data across long ranges, both spatially and temporally. This ability allows ViViT to focus on relevant parts of the video more effectively than 3D CNNs, leading to better performance, especially on complex tasks involving intricate temporal dynamics. Additionally, ViViT's separation of spatial and temporal processing allows for more efficient training and potentially better scalability, handling various video lengths and complexities with greater ease.

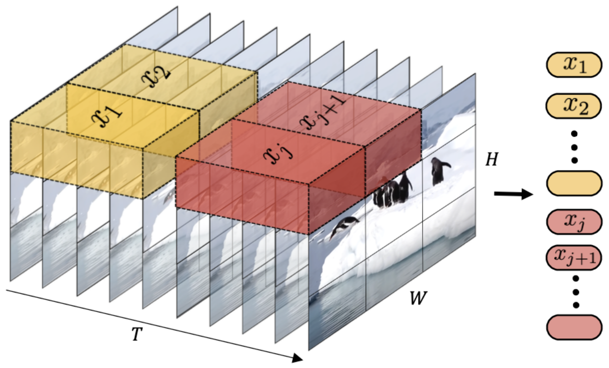

Fig. 7. Tublet Embedding[8]

The ViViT model employs a unique approach called "tublet embedding" to effectively manage video data. This technique involves segmenting video inputs into small spatiotemporal blocks, or "tublets," before processing them through the transformer architecture. By breaking down the video into these manageable chunks, ViViT can focus on both spatial details and temporal dynamics within each segment, facilitating a more detailed and nuanced understanding of the video content. Tublet embedding enables the model to efficiently capture local motion and appearance features while maintaining the global context, which is crucial for accurately interpreting complex activities and interactions in videos. This strategy not only enhances the model's ability to process and analyze videos but also improves its performance by allowing it to focus computational resources on critical features within the video, thus optimizing both learning and inference processes.

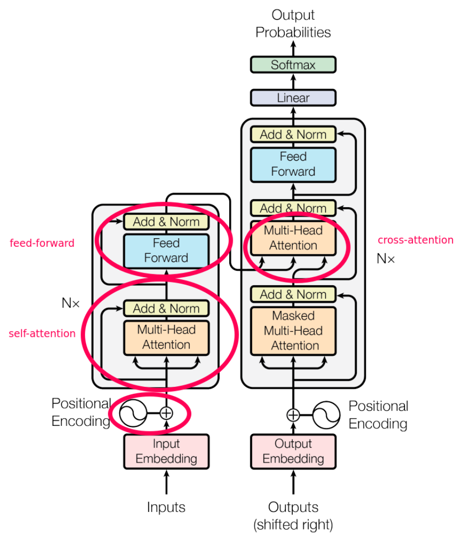

Fig. 8. Transformer Architecture[6]

In the second stage of the ViViT model, the self-attention mechanism plays a pivotal role. After tublet embeddings are enhanced with positional information, they are processed through multiple layers of self-attention. This mechanism allows the model to evaluate and understand the relationships and interactions between each tublet and all others across the entire video sequence. By focusing attention variably on different parts of the video, the model can prioritize areas with significant information and diminish less relevant data. This dynamic focusing capability is crucial for comprehensively analyzing videos, as it enables the model to capture complex patterns and temporal dynamics effectively. The self-attention stage is particularly adept at extracting contextual relationships in data, which is essential for tasks that involve understanding sequences and spatial hierarchies, such as in detailed video analysis and classification.

The final step in the process is using a simple modified Neural Network to aggregate the features and predictions. This network consists of Dense layers, as well as Layer Normalization and Global Average Pooling. This allows the output to be a probability distribution, which is essential for evaluating accuracy, and augmenting predictions with other known information. In my implementation, I use the `gelu` activation, as a more differentiable version of `relu`, which gave me a richer gradient.

3.7 Training

The training process for the ViViT model involves several key steps designed to optimize the performance and accuracy of video classification, particularly for American Sign Language. Initially, the process starts with the preparation of a diverse dataset through data augmentation techniques. Videos from the dataset are manipulated by flipping, rotating, and adding noise, thus generating synthetic variations that help the model generalize better to different conditions. These augmented datasets are then split into training and validation sets, ensuring a robust testing framework to evaluate the model's performance during training. These processes have been explained in detail in previous sections.

The core of the training involves the `run_experiment` function which orchestrates the creation, compilation, and fitting of the model. It starts by initializing the ViViT classifier with specific parameters like tubelet embedding dimension and patch size. The model is compiled using the Adam optimizer and is set to minimize the sparse categorical cross-entropy loss, a common choice for classification tasks. Key performance metrics such as accuracy and top-5 accuracy are tracked to assess the model during training.

The training process utilizes callbacks for logging training metrics to CSV files and saving the best model based on validation accuracy. If a validation set is provided, the model is trained using both training and validation data, which helps in tuning the hyperparameters more effectively and prevents overfitting.

After the training epochs are completed, the model's performance is evaluated on the training set to estimate its accuracy and top-5 accuracy, providing insights into how well the model has learned to classify the different categories accurately.

This systematic training process ensures that the ViViT model is well-optimized to perform high-quality video classification, capable of understanding and interpreting complex visual gestures.

3.8 Testing

After computing the training accuracy and saving the model, the test accuracy and test loss are computed. These are also computed alongside the training accuracy and loss at each epoch via the CSV log callback, implemented in the `run_experiment` function.

4 RESULTS

4.1 Accuracy

Unfortunately, I wasn't able to perform much hyperparameter tuning as a part of this research as the computational resources required to tun the model were very high. Instead, I focused on creating higher-quality data and making some slight changes to the learning rate and the optimizer to increase validation accuracy.

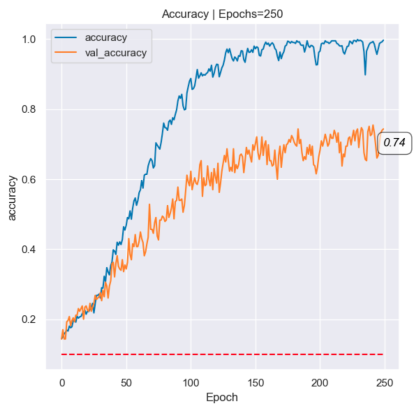

Fig. 9. Accuracy by Epoch

Above, we can see the accuracy of both the validation and training sets. It appears that while the rate of increasing accuracy does decrease, the plot is generally positive, meaning that as epochs continue to increase, the validation accuracy should not decrease. At 250 epochs, the model was able to achieve 99.6% training error and 77% accuracy on the validation set.

The dashed red line represents a chance model, where each prediction is made by randomly selecting one of the possible outputs. We see that even from the very first epoch, the model outperforms chance at about 15% validation accuracy.

4.2 ROC Curve

Following the discussion of model accuracy, we turn our attention to the Receiver Operating Characteristic (ROC) curve, which provides another dimension for evaluating the performance of the classification model. The ROC curve plots the true positive rate against the false positive rate, offering insights into the trade-offs between sensitivity and specificity across different thresholds. This visualization helps us understand the robustness of our model in distinguishing between classes under various conditions.

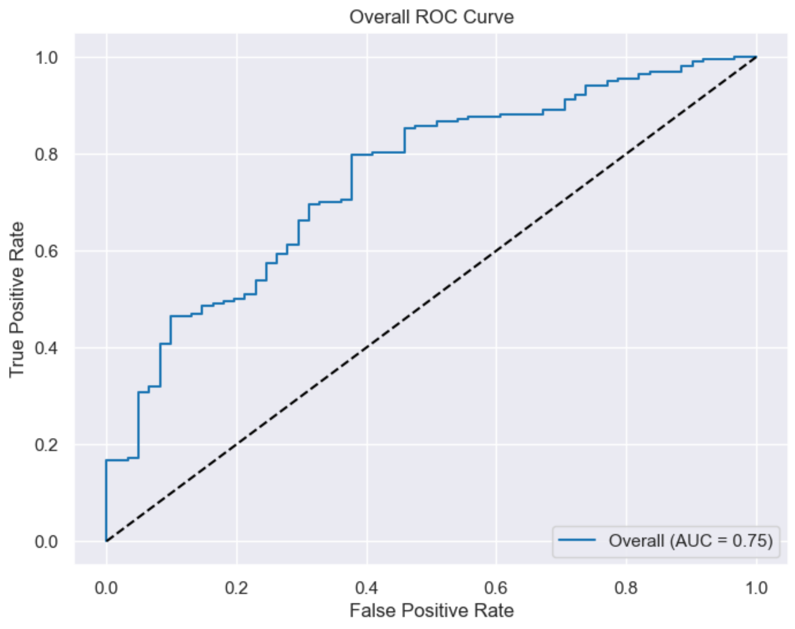

Fig. 10. ROC Curve

The ROC curve displayed above demonstrates the classification model's performance with an Area Under the Curve (AUC) of 0.75. This indicates a good ability of the model to distinguish between the classes, especially considering the multiclass nature of the problem. The curve stays well above the diagonal line of no-discrimination, which reflects random chance, particularly in the lower range of false positive rates. This suggests that the model achieves a high true positive rate without incurring a proportional increase in false positives, especially beneficial in applications where the cost of false positives is high. The performance flattens towards the higher end, indicating diminishing returns in sensitivity from increasing the model's threshold for predicting positive instances.

4.3 Top *N* Accuracy

Another metric used in multiclass problems which is sometimes more informative is Top 5 accuracy. As my model outputs probability distributions, I can calculate this metric for my model. The process for calculation involves sorting the predictions in descending order, and then taking the top 5. If the correct class is in that top 5 subset, it is counted as "correct".

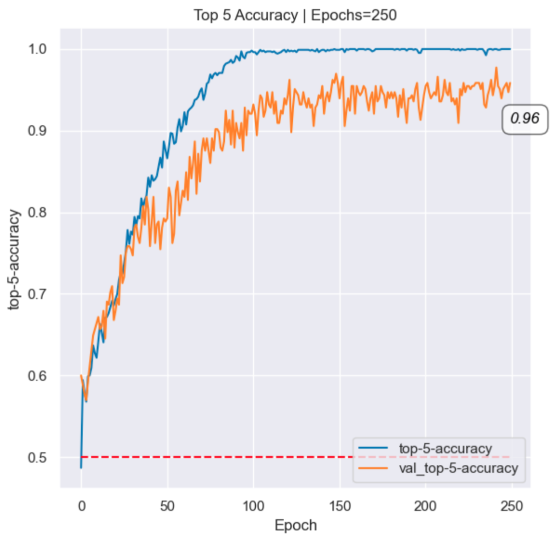

Fig. 11. Top 5 Accuracy by Epoch

Here we can see that the top 5 accuracy for the model is nearly perfect, with a final validation accuracy of 96.2%. The difference between these two quantities, (96.2% - 77%) 19.2%, represents the number of times that the model wasn't entirely sure about the classification but had a reasonable guess based on features it learned.

To this end, it is useful to examine other *n* in terms of the top *n* accuracy, to see how "unsure" the model was when it was considered "wrong" by pure accuracy.

Table 3. Top N Accuracy

N	Accuracy
1	77.0%
2	87.5%
3	92.5%
4	95.5%
5	96.2%

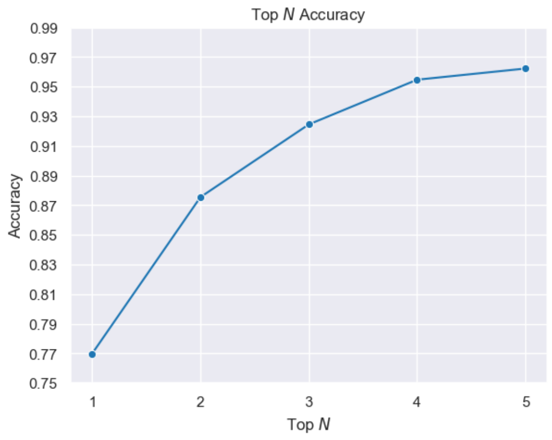

Fig. 12. Top N Accuracy

the figure

One of the trends that can be observed in the figures above is that there is a steep jump from $N = 1$ to $N = 2$, and then the rate of incline begins to decrease. This helps us to quantify the amount of uncertainty that the model had when it was classically incorrect.

For example, in about 45% of the cases where the model did not predict correctly, it thought the next most likely class was the one that was actually correct. This number increases to 67% when we increase to $N = 3$.

This information tells us that the model was able to pick up on useful features for the classification that inform the model as to what sign is represented by the video. This proves that the model is not simply picking at random very well, but in fact, is able to sign language from video representations to perform translations.

4.4 Cross Entropy Loss

The last performance metric that I monitored was loss, in the form of Cross Entropy Loss.

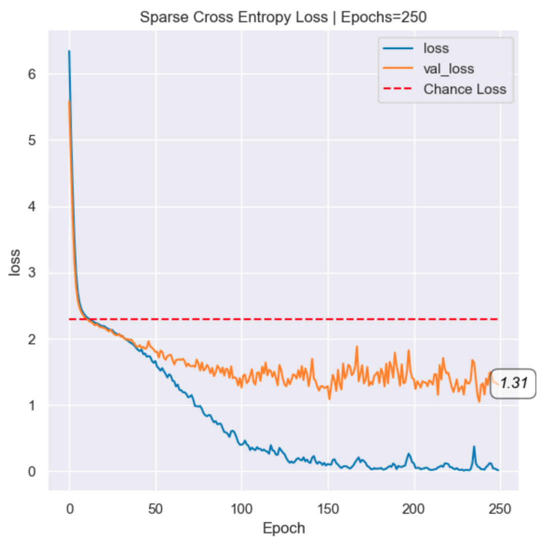

Fig. 13. Cross Entropy Loss

The dashed red line is the expected loss for a chance model with 10 classes, calculated as follows:

$$-\sum_{i=1}^{10} \frac{1}{10} \log\left(\frac{1}{10}\right) = -\sum_{i=1}^{10} \frac{1}{10} \times (-2.3026) = 10 \times \frac{1}{10} \times 2.3026 = 2.3026$$

In the figure above, we see that for roughly the first 5 epochs, the model doesn't perform as well as chance, but quickly reduces below the chance model. We note that the losses here roughly resemble a y-axis flip of the accuracy, which is to be expected. We see the same difference between the validation and training that we did in the previous plots, as the training set got near-perfect accuracy.

4.5 Evaluation

In classification, the most common way to evaluate the performance of a model in a more detailed way is to use a Confusion Matrix to examine how a model classified data points, based on what the ground truth label for the data is.

This can be more beneficial in multiclass classification, as it can reveal if there is a pattern in the misclassification. For instance, it would let you examine the case where C_2 was often misclassified as C_4, and see if C_4 was also often misclassified as C_2.

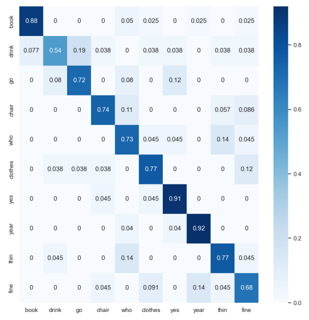

Fig. 14. Accuracy Confusion Matrix

Above we can see the Confusion Matrix for the Accuracy metric. This allowed us to examine errors that the model made systematically.

At first glance, most of the classes seem to have an accuracy that makes sense compared to the overall accuracy score of the model. Only two classes, *fine*, and *drink* have accuracies significantly lower. One possible reason for why this may happen is that in the dataset there are different variations of the same sign. For example, the word "drink" has two different signs, one for alcoholic drink and one for non-alcoholic. This is likely what caused significant errors for these classes.

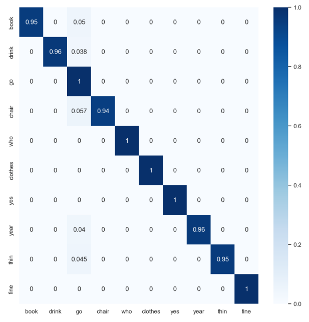

Fig. 15. Top 5 Confusion Matrix

For the top 5, there is much less that we can say about the patterns, as frankly, there are not many misclassification. One interesting note is that the classifier is nearly certain for the *drink* class, which it was only about 50% sure about in accuracy. This means that the model did pick up on features for this class that were useful, but it tended to get mixed up with other classes, namely the *go* class based on the first confusion matrix.

5 CONCLUSION

This research has marked a substantial advancement in digital inclusivity with the development of a cutting-edge real-time translation tool for American Sign Language (ASL). Utilizing Video Vision Transformers integrated with a comprehensive ASL dataset, I have successfully addressed and surpassed the limitations inherent in existing translation tools. This enhancement significantly boosts the accuracy and cultural sensitivity of ASL translations, facilitating more effective communication for the deaf and hard-of-hearing communities.

The methodology, centered on state-of-the-art computer vision technologies and an expansive dataset, enables the nuanced capture of ASL's dynamic gestures and facial expressions. The performance of my model was rigorously tested and demonstrated substantial improvements over existing systems. Notably, the model achieved an impressive 99.6% training accuracy and a 77% validation accuracy at 250 epochs, as well as a top-5 accuracy rate of 96.2%. These metrics not only underscore the model's robustness but also its capacity to handle real-world variability in ASL communication.

The Receiver Operating Characteristic (ROC) curve further substantiates the efficacy of our approach, with an Area Under Curve (AUC) of 0.75, indicating a good discriminatory ability between correct and incorrect class predictions. This is particularly important in a multiclass classification problem like ASL translation, where distinguishing between numerous similar gestures is challenging.

Furthermore, this research delves into the importance of creating a scalable model that does not compromise the interpretability or cultural integrity of ASL. By developing and utilizing synthetic data augmentation techniques, I addressed the challenge of limited data availability, enhancing the model's exposure to varied ASL signing styles and conditions. This approach not only improved the robustness of this model but also ensured that it remained sensitive to the contextual nuances of ASL.

The implications of our findings extend well beyond the technical achievements. They contribute to a broader societal impact, enhancing the accessibility of technology for ASL users and promoting inclusivity. Continued advancements in this field can bridge communication gaps not only within the deaf community but also between deaf and hearing individuals, fostering a deeper understanding and integration across diverse communication landscapes.

This project establishes and serves as a practical reference for future research in communication tools. By combining advanced AI with a deep understanding of ASL's nuances, we have taken a significant step toward creating technologies that not only perform well but also respect and reflect the diversity of user needs. This progress promises more accessible and effective communication aids for the deaf and hard-of-hearing communities, contributing positively to broader efforts of inclusivity.

6 DISCUSSION

While this research opens up ideas in new spaces, there is still a significant amount of work to be done. This research has its limitations, and before it can become something useful in the real world there is a lot more work to be done.

6.1 Challenges

While I ultimately attained a resulting model that exceeded expectations, there were a few challenges I ran into on the way.

One significant challenge in this research was managing the size and storage of video data. With over 20,000 videos, each undergoing multiple transformations, the storage requirements quickly escalated, proving to be both space-inefficient and costly. The inherent large file size of video data compounds this issue, particularly when each frame is necessary to capture the detailed nuances of ASL. To address these concerns, future studies might explore more efficient data compression techniques or more sophisticated data sampling methods that reduce the volume of data required for effective training without compromising the quality and integrity of the models. Additionally, implementing more advanced data storage solutions that can dynamically adjust compression based on data access patterns may provide a balance between accessibility and efficiency.

The computational demands of training our model were substantial, with processing times exceeding five hours even after reducing the sample size and resolution of our video inputs. This challenge is primarily attributed to the high complexity of the Video Vision Transformers and the intricate nature of the video data itself, which includes capturing fine-grained temporal and spatial dynamics of ASL. To mitigate these demands, future research could investigate more efficient architectures or adaptive training methods that require less computational power. Techniques such as transfer learning, where a pre-trained model is fine-tuned rather than trained from scratch, or the development of lightweight transformer models specifically optimized for video processing, could significantly reduce the computational load and make the technology more scalable and feasible for broader applications.

The novelty of Video Vision Transformers (ViViTs) presents another layer of complexity, as there are no out-of-the-box software solutions or libraries specifically designed for them. This gap necessitated the development of custom coding solutions, which can be time-consuming and prone to errors, potentially slowing down the research progress. Addressing this challenge effectively requires a dual approach: first, contributing to the development of open-source libraries that support ViViT architectures, thereby fostering a community of practice that can continuously improve and iterate on these tools; second, encouraging collaborations between academia and industry to accelerate the development of robust, user-friendly software that can leverage the strengths of ViViTs in various applications. As these tools become more standardized and accessible, researchers will be able to focus more on innovation and less on resolving technical constraints, pushing the boundaries of what can be achieved with ASL translation technologies.

6.2 Limitations

Additionally, due to my resources, this research has a few notable limitations that should be acknowledged.

Training our model on low-resolution data and reducing the videos to black and white were necessary compromises to manage computational demands and storage efficiency. However, these compromises potentially limited the model's performance. Higher resolution data with full color and more frames per video could capture finer details of ASL signs, which are often subtle yet critical for accurate translation. The color information, for instance, could enhance the model's ability to distinguish between similar signs where slight variations in skin tone or clothing contrast provide important contextual cues. Future research should explore the feasibility of incorporating higher resolution and full-color data, possibly through more efficient processing techniques or more powerful computational resources. This could lead to significant improvements in the model's accuracy and its ability to handle the nuanced aspects of sign language.

The necessity to limit the vocabulary for this project arose from the transformations required to make the dataset manageable and computationally tractable. While this approach was effective for a proof of concept, it inherently restricts the breadth of the model's applicability and might overlook unique errors or challenges associated with a more extensive vocabulary. Signs that are rare or complex could behave differently in translation models and might introduce new types of errors not observed with a limited vocabulary set. Expanding the vocabulary in future studies would not only test the model's robustness but also enhance its practical utility, providing a more comprehensive tool that can support a wider range of communication needs within the ASL community.

Although the dataset included signers from diverse backgrounds, the challenge of sourcing a sufficiently varied and extensive dataset remains a significant hurdle. Greater diversity in the dataset, especially with signers from a broader array of backgrounds, would enhance the model's generalizability and accuracy across different demographic groups. This includes not only ethnic and regional diversity but also variations in age, gender, and signing style. Future efforts to expand the dataset should focus on including signers who represent the full spectrum of the ASL community. Such an enriched dataset would train the model to be more adaptable and sensitive to the wide variations in sign language usage, ultimately leading to a translation tool that is more inclusive and effective for all users.

6.3 Future Work

As I look forward to continuing this project over the next year, we have outlined several strategic enhancements and expansions to elevate the effectiveness and accessibility of our ASL translation tool. First, we plan to substantially increase the vocabulary size. This will involve curating a larger dataset that includes a broader array of signs, particularly those less commonly used. By expanding the dataset and incorporating more diverse signers, we aim to refine the model's accuracy and its capability to handle complex translation tasks.

The next phase of development will extend the model's capabilities beyond word-level translation to include sentence translation. This enhancement will involve integrating more complex linguistic models to accurately translate full sentences, considering syntax and grammar to maintain contextual and semantic coherence. Additionally, we plan to augment the prediction probabilities by incorporating advanced language models such as BERT and GPT, which will improve the model's understanding of contextual nuances.

A significant practical extension of this project will be the development of a webcam-based application for real-time use. This application will allow users to translate ASL live using their computer's webcam, which will significantly enhance the tool's accessibility and ease of use with a user-friendly interface.

Finally, in collaboration with my academic advisor, I plan to publish the results of my research. This publication will detail the advancements made in the project, discuss the technological and methodological innovations, and outline the impacts of this work on the deaf and hard-of-hearing communities. Through these steps, we aim to advance the state of technology in ASL translation and make a meaningful contribution to enhancing communication accessibility for the deaf and hard-of-hearing communities.

7 REPRODUCABILITY

All information about the code and data for this project can be found on the GitHub repository. Some additional visualizations can be found on the thesis defense slides.

7.1 Data

The dataset for this project was the WLASL dataset[11]. Instructions for downloading part of the data can be found on their website. However, many of the links (about 60-70%) no longer work, and as such the data itself has to be requested from the publishers of the work.

The dataset in full is about 3 GB and increases for each transformation you perform. Data is available by request from me, as long as terms set forth by the original publisher are agreed to.

7.2 Code

As mentioned previously, all of the code is stored at https://github.com/jonathanferrari/thesis. The repository is organized in the following structure.

At the top level, the standard files such as `.gitignore`, `LICENSE`, and `README.md` are stored. There are also 4 folders with information about the project stored in them.

The first folder is the `write-ups` folder. This contains some of the assignments from the year, and will eventually contain this paper. These files are stored in both markdown and HTML.

The second folder is the `plots` folder, which contains many of the visualizations relevant to the research. It contains all of the visualizations used in this paper, as well as evaluative plots for a few other iterations of the model.

The third folder is the `models` folder. This contains some of the models stored as `.keras` files. It also contains some of the evaluative CSVs that contain the metrics of each epoch for the models.

The fourth folder contains, `data`, which is not in the public repository due to the size of the data but is available by request. This folder contains all of the original videos, one sub-folder for each transformation of the videos, and data files such as the labels for the videos, and metadata about each video.

Finally, the `code` folder contains the relevant code base.

The important Python files are `cnn.py, data.py, synthetic.py, vivit.py`. The `cnn` file contains a working 3D CNN, which was my first model attempt before I decided to use a ViViT. I decided not to use this as a baseline as it was too inefficient to retrain multiple times. The `synthetic` file contains functions such as `rotate_video`, which allowed me to generate more data for my research. The `data` file contains a few constant variables, such as `files`, which is an array of file names that lead to the videos of the dataset. It also contains helpful utility functions such as `get_video_label` and `load_video`. However, the main part of this file is the `VideoDataGenerator` class, (mentioned previously) which allowed me to efficiently train my models while not using too much overhead or complex code. Finally, the `vivit` file contains actual classes for the model, as well as helpful utility functions which allowed me to thoroughly test and train each of my models.

I was able to easily use the code from all of my files in my notebook with this simple block of code:

```python
from vivit import run_experiment, test_files, classes
from data import get_video_label, VideoDataGenerator, labels_df, files, labels
import synthetic
```

As for the notebooks, there are a few relevant ones: `classify.ipynb, data.ipynb, eda.ipynb, eval.ipynb`

The `data` and `eda` notebooks were used to explore the format and shape of the data and were used to create many of the figures and visualizations in this paper. The `eval` notebook was used to evaluate and compare models, as well as monitor them during the training process, with plots that updated after each epoch of training. The `classify` notebook is the most important, as this is the environment that I used to train and save all of the models that I used.

Together the data pipeline and code repository worked efficiently and allowed me to streamline the process of writing code and allowed me to focus more on exploration.

8 ACKNOWLEDGEMENTS

Special thanks to Kevin Miao, who offered his extensive expertise and aid on this project. I would also like to acknowledge professors Eric Van Dusen and Narges Norouzi for their support in allowing me the freedom to work on new research, and the support to guide me when I needed help.

REFERENCES

[1] 2024. *American Sign Language (ASL) | Britannica*. https://www.britannica.com/topic/American-Sign-Language

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. ViViT: A Video Vision Transformer. https://arxiv.org/abs/2103.15691

[3] Ronald T Azuma. 1997. A Survey of Augmented Reality. *Presence: Teleoperators & Virtual Environments* 6 (08 1997), 355–385. https://doi.org/10.1162/pres.1997.6.4.355

[4] Dulari Bhatt, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. 2021. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics* 10 (10 2021), 2470–2470. https://doi.org/10.3390/electronics10202470

[5] Vivek Bheda and Dianna Radpour. 2017. Using Deep Convolutional Networks for Gesture Recognition in American Sign Language. https://arxiv.org/abs/1710.06836v3

[6] Stefania Cristina. 2022. The Transformer Model - MachineLearningMastery.com. https://machinelearningmastery.com/the-transformer-model/

[7] Amanda Duarte, Shruti Palaskar, Lucas Ventura, Deepti Ghadiyaram, Kenneth DeHaan, Florian Metze, Jordi Torres, and Xavier Giro-i Nieto. 2021. How2Sign: A Large-Scale Multimodal Dataset for Continuous American Sign Language. *Thecvf.com* (2021), 2735–2744. https://openaccess.thecvf.com/content/CVPR2021/html/Duarte_How2Sign_A_Large-Scale_Multimodal_Dataset_for_Continuous_American_Sign_Language_CVPR_2021_paper.html?ref=https://githubhelp.com

[8] Momal Ijaz. 2022. ViViT Video Vision Transformer - AIGuys - Medium. https://medium.com/aiguys/vivit-video-vision-transformer-648a5fff68a4

[9] Annelies Kusters and Ceil Lucas. 2022. Emergence and evolutions: Introducing sign language sociolinguistics. *Journal of Sociolinguistics* 26 (02 2022), 84–98. https://doi.org/10.1111/josl.12522

[10] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. 2019. Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks. https://arxiv.org/abs/1901.10323v3

[11] Dongxu Li, Cristian Rodriguez, Xin Yu, and Hongdong Li. 2020. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *The IEEE Winter Conference on Applications of Computer Vision*. 1459–1469.

[12] Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. 2023. Contextual Transformer Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (02 2023), 1489–1500. https://doi.org/10.1109/TPAMI.2022.3164083

[13] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. 2022. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE transactions on neural networks and learning systems* 33 (12 2022), 6999–7019. https://doi.org/10.1109/tnnls.2021.3084827

[14] Yang Liu, Yao Zhang, Yixin Wang, Feng Hou, Jin Yuan, Jiang Tian, Yang Zhang, Zhongchao Shi, Jianping Fan, and Zhiqiang He. 2023. A Survey of Visual Transformers. *IEEE transactions on neural networks and learning systems* (01 2023), 1–21. https://doi.org/10.1109/tnnls.2022.3227717

[15] Tim McCue. 2022. Learn American Sign Language! - DDOmbuds.org. https://ddombuds.org/2022/04/20/learn-american-sign-language-2/

[16] Munir Oudah, Ali Al-Naji, and Javaan S Chahl. 2020. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging* 6 (07 2020), 73–73. https://doi.org/10.3390/jimaging6080073

[17] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. 2020. Visual Transformers: Token-based Image Representation and Processing for Computer Vision. https://arxiv.org/abs/2006.03677

[18] Shusheng Yang, Xinggang Wang, Yu Li, Yuxin Fang, Jiemin Fang, Wenyu Liu, Xun Zhao, and Ying Shan. 2022. Temporally Efficient Vision Transformer for Video Instance Segmentation. https://arxiv.org/abs/2204.08412v1

[19] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. 2021. DeepViT: Towards Deeper Vision Transformer. https://arxiv.org/abs/2103.11886

[20] Ronglai Zuo, Fangyun Wei, and Brian Mak. 2023. Natural Language-Assisted Sign Language Recognition. https://arxiv.org/abs/2303.12080v1