Bridging Vision and Language for Sign Language Understanding

Ali Sartaz Khan
Stanford University
askhan1@stanford.edu

Dat Tran
Stanford University
dattran@stanford.edu

Sufen Fong
Stanford University
sffong@stanford.edu

Abstract

Despite recent advances in computer vision, real-time sign language translation remains a challenging problem, hindered by the lack of standardized methodologies and large-scale benchmark datasets. In this work, we evaluate the effectiveness of different feature representations and model architectures on two core tasks: isolated sign recognition using the WLASL dataset and continuous sentence-level translation using the How2Sign dataset.

Our experiments demonstrate that full-frame RGB inputs consistently outperform skeleton-based and keypoint-only representations. For isolated sign recognition, a model combining I3D100 features with a Transformer encoder achieved the highest accuracy of 77%. For sentence-level translation, integrating the I3D100 encoder with a GPT-2 decoder produced the best results, reaching a BLEU score of 7.25% on a subset of How2Sign. These findings underscore the importance of rich spatiotemporal features and pretrained language models for effective sign language understanding.

1. Introduction

American Sign Language (ASL) serves as the primary means of communication for millions of deaf and hard-of-hearing individuals worldwide. In the United States alone, over 11 million people identify as deaf or have significant hearing loss, resulting in substantial social and economic barriers when communicating with the hearing population. Traditional sign language interpretation requires skilled human interpreters, which are costly and often unavailable in many real-time scenarios such as medical consultations, educational settings, and public services. Consequently, automatic sign language recognition and translation systems have the potential to bridge this communication gap, offering real-time, scalable solutions that improve accessibility and inclusion.

In this project, we focus on two complementary tasks: (1) isolated sign recognition using the Word-Level American Sign Language (WLASL) dataset [14], and (2) continuous sentence-level ASL translation using the How2Sign dataset [5]. For the WLASL task, the input to our algorithm is a video clip of a single ASL sign, uniformly sampled to 50 frames (224×224 pixels each), from which we extract either raw RGB frames or skeletal keypoints. We then employ deep architectures - ranging from Convolutional Neural Network (CNN) combined with Long Short-Term Memory (LSTM) layers baselines to a pretrained Inflated 3D ConvNet (I3D) [3] coupled with a Transformer encoder - to output a predicted sign label among 100 classes. For the How2Sign task, the input is a continuous ASL video clip sampled to 64 frames (1280x720 pixels each), along with optional optical flow or skeletal features, and our goal is to generate the corresponding English sentence. Here, we again leverage a pretrained I3D backbone to encode spatiotemporal features, followed by either a Transformer encoder-decoder or a large language model (e.g., GPT-2 [18] or BART [13]) to produce fluent English translations.

Our motivation is twofold. First, while many prior works address isolated word recognition [14, 12], few have directly compared keypoint-based versus frame-based representations under a unified architecture pool. Second, continuous translation from video to text remains a challenging, underexplored problem due to the complexities of fast hand movements, occlusions, and sentence-level context. By systematically evaluating a spectrum of architectures on both WLASL and How2Sign, we aim to identify which combinations of feature representation and model architecture yield the best performance for American Sign Language recognition and translation.

2. Related Work

Existing research on sign language recognition and translation can be broadly grouped into three categories: (1) feature representation and pre-processing methods, (2) publicly available datasets and their characteristics, and (3) model architectures for isolated recognition and continuous translation.

Feature Representation. Accurate sign recognition hinges on extracting informative spatial and temporal features that capture rapid hand movements, body posture, and facial expressions. Early approaches relied on hand-crafted features or raw RGB frames fed into 2D CNNs [12, 8]. With advances in pose estimation, skeleton-based representations have gained popularity, as they distill high-level joint trajectories, reducing background noise and computational complexity. Jiang *et al*. [10] proposed a skeleton-aware multi-modal network that fuses keypoints with CNN features to improve recognition robustness. Similarly, Gan *et al*. [7] demonstrated that concatenating skeletal keypoints with full-frame RGB channels helps the network distinguish subtle gesture variations - e.g., identical handshapes at different body locations - crucial for ASL semantics.

Datasets. Publicly available datasets have been pivotal to progress in this field. The WLASL dataset [14] contains over 2,000 sign classes and tens of thousands of video instances collected from a diverse signer pool. Most works focus on a subset of the top 100 or 200 frequent signs to balance class distribution [14]. How2Sign [5] offers over 80 hours of continuous ASL video sourced from instructional videos, annotated with English translations and 2D keypoints. Prior to WLASL and How2Sign, earlier datasets such as RWTH-PHOENIX-Weather 2014T were domain-specific (weather forecasting) and limited in vocabulary [2]. The release of How2Sign and WLASL has enabled end-to-end deep learning pipelines that jointly learn spatiotemporal features and language modeling [1, 22].

Model Architectures for Isolated Recognition. Traditional frame-based pipelines process each RGB frame through a backbone CNN (e.g., ResNet50 [9] or MobileNetV2 [19]) to extract spatial features, followed by sequence modeling via LSTMs and temporal attention [4, 12]. Dima *et al*. [4] fine-tuned YOLOv5, which was already pretrained on COCO dataset, to recognize finger-spelled vocabulary, indicating that object-detection backbones can be repurposed for sign recognition. More recent works leverage 3D ConvNets to directly capture spatiotemporal information. Carreira and Zisserman's Inflated 3D ConvNet (I3D) [3], pretrained on Kinetics-400 [11], has become a standard for video understanding and has been applied to ASL with Transformer to model long-range dependencies [22]. Budria *et al*. [1] compared LSTM-based, PerceiverIO, and Transformer-based encoders on How2Sign, finding that pure Transformer models outperform recurrent approaches for continuous sign recognition.

Model Architectures for Continuous Translation. ASL video translation to English text is inherently more complex due to sentence-level structure and linguistic nuances. Early efforts used CNN+LSTM pipelines with attention mechanisms to generate gloss sequences or English translations [8, 17]. Tarres *et al*. [22] introduced a baseline that combines I3D with Transformers encoder-decoder, achieving a BLEU score of 8.0 on How2Sign. Topic-level detection methods [1] segment continuous videos into semantically coherent units before translation, facilitating alignment between video frames and text. Despite these advances, achieving both high accuracy and real-time inference remains challenging due to model complexity and limited annotated data.

In summary, while keypoint-based methods excel in computational efficiency and noise robustness [10, 7], frame-based and multi-stream approaches leveraging pre-trained backbones (e.g., I3D+Transformer) have demonstrated superior accuracy on both isolated and continuous sign recognition tasks [22, 1]. Our work builds upon these insights by systematically evaluating skeleton-only, frame-only, and I3D+Transformer pipelines, as well as exploring large language model (e.g., GPT-2 [18] or BART [13]) decoders, to determine the optimal architecture for ASL recognition and translation on WLASL and How2Sign.

3. Datasets

3.1. WLASL

We use the Word-Level American Sign Language (WLASL) dataset [14], a large-scale collection of isolated sign language videos curated for recognition tasks. The full dataset includes over 2,000 unique sign classes and tens of thousands of video samples captured from a diverse group of signers. For this project, we focus on a subset comprising the 100 most frequent signs by video count, enabling a balanced yet tractable classification task. Each selected sign is represented by approximately 15–20 video samples, recorded at 25 frames per second. Our final split includes 1,249 videos in total: 816 for training, 117 for validation, and 316 for testing.

3.1.1 Features

Each video in our subset is reshaped to 224x224 pixels, normalized by dividing the pixels by 255, and preprocessed to contain 50 frames that were uniformly sampled across its duration. For each frame, we extract skeletal keypoints using MediaPipe's holistic model [16], which provides 3D coordinates for the hands. These keypoints serve as the primary features for our model which encodes spatial and gestural information crucial for sign language recognition.

This preprocessing pipeline allows the model to oper-

Figure 1. Keypoint-extracted frames from the WLASL dataset

ate on structured pose representations instead of raw pixel data, improving efficiency and enabling better generalization. Figure 1 shows an example of keypoint annotations extracted from a WLASL video frame.

3.2. How2Sign

How2Sign [5] is a large-scale, multimodal, and multi-domain dataset for continuous American Sign Language (ASL) recognition. It comprises over 80 hours of sign language video data and 35,000 validated sentence-level annotations. Derived from the How2 instructional video series, the dataset includes 31,000 training, 2,300 validation, and 1,700 test clips, spanning a vocabulary of approximately 16,000 English words. The videos are recorded in 720p high definition (1280×720 pixels) at 30 frames per second. On average, each clip contains 162 frames (approx 5.4 seconds) and 17 words per sentence.

3.2.1 Features

We used two types of features: full-body poses and skeleton poses, that are both provided in the How2Sign dataset. We preprocessed each clip to contain 64 uniformly-spaced frames of size 224x224 and normalized by dividing the pixels by 255. Figures 2 and 3 show the full body and skeleton frames extracted from the clip respectively.

Figure 2. Extracted frames (full body) from the How2Sign dataset

4. Methods

4.1. WLASL

4.1.1 Keypoint-based Variant

LSTM Model. For a lightweight approach, we used 3D hand keypoints extracted from each frame via MediaPipe

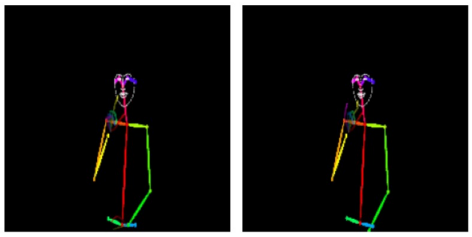

Figure 3. Extracted frames (skeleton) from the How2Sign dataset

(see Figure 1) [17]. These keypoints were concatenated and treated as a sequence, which was passed through a single-layer Long Short-Term Memory (LSTM) network to capture temporal dependencies across frames. The LSTM output was followed by a linear classification layer. This model operates entirely on pose-based features and avoids the computational overhead of processing raw images.

4.1.2 Frame-based Architecture Variants

CNN + LSTM. We implemented a baseline model using convolutional layers to process individual video frames, followed by an LSTM to model temporal relationships [17]. The CNN encoder learns spatial features from each frame, which are then fed sequentially to the LSTM. A linear output layer maps the final temporal embedding to class logits.

Pretrained CNN + LSTM. To evaluate alternative feature extractors, we introduced an alternative solution, using pretrained 2D CNNs (ResNet50 [9] and MobileNetV2 [19]), both initialized with weights trained on the ImageNet dataset. These models were applied frame-by-frame to extract high-level spatial features from each video frame, effectively encoding the visual content of each moment in the sequence. The resulting per-frame feature vectors were fed into a bidirectional LSTM to capture temporal dependencies across the sequence. A temporal attention mechanism was then applied to weigh the importance of each frame, enabling the model to focus on the most informative parts of the video. The attention-weighted sum of LSTM outputs was passed through a linear classifier to predict the corresponding sign. This architecture is designed to learn not only temporal patterns but also which specific moments are most relevant for accurate sign identification.

I3D100 + LSTM. The Inflated 3D ConvNet (I3D) backbone is initialized with weights pretrained on Kinetics-400 [11] for 100 classes, and used to extract spatiotemporal embeddings from ASL video clips. As an alternative to the Transformer decoder, we introduced a novel approach, feeding the resulting frame-wise feature sequence into a bidirectional LSTM to model temporal dynamics. A temporal attention mechanism then aggregates the LSTM out-

puts into a single context vector, which is passed through a linear classification head to predict the sign class. This architecture leverages powerful pretrained 3D features together with sequential encoding and attention.

I3D100 + Transformer. The same pretrained I3D model was also used here to extract spatiotemporal features from ASL videos. This was used to extract deep spatiotemporal features from short video clips. These features were treated as a sequence and passed into a Transformer encoder to capture long-range dependencies between temporal segments. The final representation was pooled and passed through a linear classification layer [20]. This architecture leverages strong pretrained spatiotemporal features and the expressivity of self-attention.

4.2. How2Sign

4.2.1 Input-Based Variants

We explored several input-based preprocessing techniques to evaluate how different video representations affect model performance. All variants in this section were trained using the I3D100-Transformer Encoder + Transformer Decoder architecture. We chose this setup as our baseline for How2Sign captioning, because the encoder showed the best classification performance on WLASL (see Table 1).

Skeleton-Based. The How2Sign dataset provides synthetic skeleton videos generated using the EDN (Everybody Dance Now) motion transfer and synthesis framework. EDN transfers pose and keypoint information from source videos onto a target identity, effectively synthesizing a simplified representation of body motion. As shown in Figure 2, these skeleton videos feature a blacked-out background to eliminate speaker identity and appearance variation. This reduces visual noise and prevents the model from learning irrelevant features such as background, lighting, or camera setup. The skeleton format emphasizes body movement and pose, though it may omit critical contextual cues like facial expressions and hand shape due to keypoint estimation limitations.

Optical Flow-Based. We computed optical flow representations from the full-body RGB videos to emphasize temporal motion features. Optical flow algorithms compute a 2D vector field representing motion between consecutive frames. We tested three flow algorithms, Farneback [6] , TV-L1 [21], and Lucas-Kanade [15], and selected the Lucas-Kanade method for its computational efficiency and emphasis on tracking salient keypoints rather than full-frame pixel motion. For each video, we calculated the mean motion across frames and selected the first 64 frames that exceeded a motion threshold of 1.0. Lucas-Kanade parameters included a window size of 7.0, maximum corners set to 100, and a quality level of 0.3. These settings were chosen to prioritize detection of small, fast motions common in signing.

First 64 Frames. Inspired by prior work [22], which used the first 16 consecutive frames for I3D-based sign language translation, we investigated the use of the first 64 consecutive frames from each How2Sign video. This simple heuristic requires no prior knowledge of motion or semantics and serves as a straightforward baseline for evaluating fixed-length temporal representations.

Linearly Sampled Frames. To improve temporal coverage while maintaining a fixed input length, we implemented uniform linear sampling. For each video, we sampled 64 frames spaced evenly across its entire duration, regardless of motion intensity. This method ensures that early, middle, and late segments of the video are all represented, helping the model attend to long-range dependencies in signing without bias toward the beginning of the clip.

4.2.2 Architecture-based Variants

I3D100-Transformer Encoder + Transformer Decoder. Since I3D-Transformer encoder was the best performing architecture for WLASL, we introduced a novel method by adding the Transformer decoder to translate How2Sign videos into sentences. The decoder uses masked multi-head self-attention to ensure token predictions depend on the previous token. The encoder-decoder cross-attention allows the decoder to attend to the encoder's output, facilitating the generation of target sequences based on encoded features. A fully connected linear layer projects the decoder's output to the vocabulary size (7000), producing logits for each token in the target sequence. The benefit of using I3D-Transformer-ASL model is that it is tailored for American Sign Language recognition and translation tasks so the model can be trained end-to-end.

I3D100 + GPT-2 Decoder. In this architecture, we use a pretrained I3D model trained on 100 classes to extract 1024-dimensional spatiotemporal features from input video clips. These features are then passed through a projection layer and then directly fed into the embedding space of a frozen GPT-2 language model - another method we proposed - which acts as a decoder to generate captions autoregressively. This model serves as a baseline for video-to-text generation using only visual features and a pretrained language model. Its simplicity makes it fast to train, and it provides insight into how well raw I3D features encode sign language semantics without additional temporal modeling.

I3D100/2000-Transformer Encoder + GPT-2 Decoder. To improve upon the baseline, we introduce a Transformer encoder between the I3D feature extractor and the GPT-2 decoder. The I3D model extracts 1024-dimensional spatiotemporal features for each frame, which are then processed by a 2-layer Transformer encoder to capture long-range temporal dependencies. The output of the encoder is projected as a learned prefix to condition GPT-2's text generation. We experiment with two variants of this architecture: one using an I3D backbone pretrained on 100 classes (I3D100), and another using an I3D model trained on 2000 classes (I3D2000). The latter benefits from greater signer diversity and vocabulary coverage, improving the model's ability to generalize to varied signing styles in How2Sign. This hybrid setup combines rich visual features, temporal modeling, and a powerful language decoder and is suited for generating sentence-level captions from continuous sign language videos.

I3D100-Transformer Encoder + BART Decoder. This model retains the I3D100 feature extractor and Transformer encoder, but replaces GPT-2 with BART as the decoder, as another original method of ours. Unlike GPT-2, BART is a sequence-to-sequence model with a bidirectional encoder and an autoregressive decoder, trained with denoising objectives. By using BART, we take advantage of its stronger generation capabilities and robustness to noisy inputs. This setup allows us to better handle the diverse and often noisy video recordings in How2Sign, making it a strong candidate for improving caption fluency and correctness in challenging real-world signing scenarios.

5. Experiments

5.1. Training Procedure

5.1.1 WLASL

All models were trained using the same procedure for consistency. We used the Adam optimizer with a learning rate of 1e-5 and weight decay of 1e-6. The learning rate was chosen to prevent overfitting due to the small dataset size and the weight decay was selected for mild regularization while preserving the pretrained model's feature representations. More details can be found in section 5.3.1. Adam optimizer was used due to its fast convergence with minimal tuning and its combinaion of AdaGrad and RMSProp benefits. Models were trained for 10 epochs using a standard cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

where C is the number of classes, y_i is the true label, and $\hat{y}_i$ is the predicted class probability.

Input video clips were uniformly sampled and reshaped to match the expected input format for each model. During training, only the task-specific components (e.g., classifier head, LSTM, Transformer layers) were updated; pretrained encoders (when used) were frozen.

After each epoch, models were evaluated on a held-out validation set using both average loss and top-1 accuracy:

$$\text{Accuracy} = \frac{\#\text{ correct predictions}}{\#\text{ total samples}} \times 100\%$$

5.1.2 How2Sign

All models were trained using a consistent procedure to ensure fair comparison. We used the Adam optimizer with a learning rate of 1e-4 and a weight decay of 1e-6. The Adam optimizer and decay of 1e-6 were selected to be the same as the WLASL experiments. The learning rate of 1e-4 was chosen after generating the best BLEU scores on a subset of 1000 training samples. Each model was trained for 10 epochs using a cross-entropy loss function applied at each time step over the output vocabulary. The objective was to maximize the likelihood of the correct token sequence:

$$\mathcal{L}_{\text{Model}} = -\sum_{t=1}^{T} \log P(x_t \mid x_1, x_2, \ldots, x_{t-1}; \theta)$$

where T is the target sequence length, x_t is the token at time step t, and $P(x_t \mid x_1, \ldots, x_{t-1}; \theta)$ is the model's predicted probability of x_t given the preceding tokens and model parameters θ.

Input video clips were uniformly sampled and preprocessed to match the required input format of each architecture. During training, both the pretrained encoder and the large language model (LLM) decoder were kept frozen to isolate the effects of the learned prefix conditioning. After each epoch, models were evaluated on a held-out validation set using both the average validation loss and BLEU score. The average loss is computed as:

$$\text{Average Loss} = \frac{\text{Cumulative Validation Loss}}{\#\text{ Validation Samples}}$$

To evaluate caption quality, we used the BLEU score, a standard metric in machine translation that measures the overlap of n-grams between generated and reference sentences, with a penalty for brevity. It is defined as:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where N is the maximum n-gram order, w_n is the weight assigned to each n-gram (typically uniform), and p_n is the modified precision for n-grams of size n.

The brevity penalty (BP) is given by:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases}$$

where c is the length of the candidate sentence and r is the length of the reference. This penalty discourages the model from generating overly short translations that might superficially appear precise.

5.2. Quantitative Results

5.2.1 WLASL

We evaluated a range of architectures using different input representations, 3D keypoints extracted via MediaPipe or raw RGB video frames, to determine the most effective combination for word-level sign classification on the WLASL dataset. Each input type was paired with multiple model architectures, and performance was evaluated using classification accuracy.

Table 1. Model architecture performance on WLASL.

Model Architecture	Train Size	Accuracy (%)
Keypoint-Based Variants		
LSTM	816	6.60
Frame-Based Variants		
CNN + LSTM	816	0.50
ResNet50 + LSTM	816	2.40
MobileNetV2 + LSTM	816	1.80
I3D100 + LSTM	816	75.50
I3D100 + Transformer	**816**	**77.00**

The results indicate that raw frame inputs paired with pretrained spatiotemporal models significantly outperform keypoint-based or CNN-LSTM models trained from scratch. In particular, the combination of I3D and Transformer layers proved highly effective, suggesting that leveraging both local and global temporal context is critical for accurate gesture recognition. In 5.3.1, we conducted a hyperparameter importance analysis for the I3D + Transformer architecture to further refine performance and identify the most influential configurations.

5.2.2 How2Sign

Preliminary Results. Table 2 highlights clear performance differences across input and architecture variants on a subset of How2Sign. Input-based methods such as optical flow and fixed-frame extraction yielded low BLEU scores, indicating poor caption quality. Linearly sampling frames performed marginally better (0.07%), suggesting that temporal coverage helps even without motion-aware preprocessing. Hence, we used linearly sampled frames for all consequent experiments. Architecture-based models using pretrained I3D backbones significantly outperformed input baselines. The `I3D100 + GPT-2` model achieved the highest BLEU score (6.58%), with performance further improved by using a more expressive visual backbone in `I3D2000-Transformer + GPT-2` (6.70%). The addition of a Transformer encoder in `I3D100-Transformer + GPT-2` slightly decreased performance (6.10%), possibly due to overfitting on the small dataset. The `I3D100 + BART` variant underperformed (0.12%), likely due to misalignment between BART's pretraining objective and the prefix-only conditioning used in our setup.

Table 2. BLEU scores of different architectures on How2Sign.

Model Architecture	Train Size	BLEU (%)
Input-Based Variants		
Skeleton Videos	1.5k	0.04
Optical Flow	1.5k	0.01
First 64 Frames	1.5k	0.01
Linearly Sampled 64 Frames	**1.5k**	**0.07**
Architecture-Based Variants		
I3D100-Trans. Enc. + Trans. Dec.	1.5k	0.07
I3D100 Enc. + GPT-2 Dec.	**1.5k**	**6.58**
I3D100-Trans. Enc. + BART Dec.	1.5k	0.12
I3D100-Trans. Enc. + GPT-2 Dec.	1.5k	6.10
I3D2000-Trans. Enc. + GPT-2 Dec.	**1.5k**	**6.70**
Best Performing Architecture		
I3D2000-Trans. Enc. + GPT-2 Dec.	9.0k	6.92
I3D100 Enc. + GPT-2 Dec.	**9.0k**	**7.25**
I3D100 Enc. + GPT-2 Dec.	31k	7.16

Best Performing Architectures. To further investigate how data scale affects captioning performance, we trained our two best-performing architectures on an expanded subset of 9k How2Sign training samples (see Table 2). The I3D100 encoder combined with a GPT-2 decoder achieved the highest BLEU score of 7.25%, while the I3D2000-Transformer encoder with GPT-2 reached 6.92%. These results indicate that increasing the training data improves generation quality. However, when we scaled the I3D100 + GPT-2 model to the full dataset of approximately 30k training samples, the BLEU score slightly dropped to 7.16%. This suggests that simply increasing the dataset size does not always yield better performance. One possible explanation is that the larger dataset introduced greater variability in signer appearance, signing speed, and video quality,

which may have acted as noise without corresponding architectural changes or regularization strategies. Another explanation is overfitting, as we saw BLEU score drop after early gains in the first two epochs. Some strategies that could help are increasing the weight decay, applying dropout to more parts of the model, and label smoothing.

5.3. Qualitative Results

We conducted a qualitative analysis on the best-performing architectures for each dataset. For WLASL, this was the `I3D + Transformer` model, while for How2Sign, it was the `I3D100 Encoder + GPT-2 Decoder` model.

5.3.1 WLASL

Hyper-parameters. We performed a hyperparameter analysis while varying hidden state size, dropout, learning rate, number of Transformer layers, and weight decay. Figure 4 shows the results, with each line representing a unique configuration colored by accuracy. Table 3 summarizes the importance and correlation of each hyperparameter with final model performance.

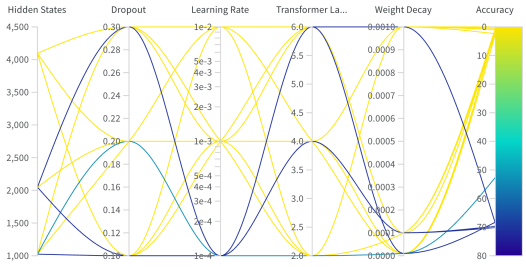

Figure 4. Results from the hyperparameter analysis

Among all parameters, the learning rate had the strongest influence on accuracy, with a negative correlation of -0.592, indicating that values above 1e-4 consistently degraded performance. Hidden state size also showed a moderate negative correlation (-0.619), suggesting that larger sizes may lead to overfitting or instability. In contrast, the number of Transformer layers correlated positively with accuracy (0.609), with optimal results achieved using 4–6 layers which highlights the value of deeper temporal modeling. Dropout and weight decay were less impactful, both showing mild negative correlations. This suggests that when using a frozen backbone and a moderately sized dataset, strong regularization is less critical. Overall, the results underscore the importance of carefully tuning learning rate, model depth, and hidden dimensionality.

I3D Saliency Maps. Figure 5 illustrates the most prominent heatmap saliency visualizations from the I3D backbone for the sign corresponding to the word "doctor." The highlighted regions represent the spatial areas most influential in the model's prediction at each time step. Across consecutive frames, we observe consistent activation over the signer's upper body and hands, indicating that the model is attending to the hand movements and posture, which are critical for interpreting this sign. This supports the effectiveness of the I3D feature extractor in capturing semantically rich spatiotemporal cues necessary for accurate sign classification.

Table 3. Hyperparameter correlation with accuracy

Hyperparameter	Importance (↑)	Correlation
Learning Rate (lr)	High	-0.592
Hidden States	Moderate	-0.619
Transformer Layers	Moderate	0.609
Weight Decay	Low	-0.187
Dropout	Low	-0.400

Figure 5. I3D Saliency Maps from WLASL

Transformer's Temporal Attention. Figure 6 presents the temporal attention matrix from the Transformer encoder, averaged across attention heads, for the same example shown in the saliency maps. Each cell denotes how much a query time step attends to another time step, with frame ranges mapped back to their original positions in the 50-frame sequence. The model exhibits a sharp focus on the frame range 16–23 across nearly all query steps, suggesting that this temporal segment contains the most discriminative motion features for recognizing the word "doctor." This aligns with the saliency maps in Figure 5, where the signer's hand movement is most visually prominent within that interval. Together, these results demonstrate how the spatial and temporal components of the model cooperate to localize and attend to the most informative parts of a sign.

5.3.2 How2Sign

I3D Saliency Maps. Figure 7 presents gradient-based saliency maps for two consecutive frames, highlighting regions of the video that contribute most to the model's loss during caption generation. The visualizations reveal that the model primarily focuses on the hands and face, which are essential regions for recognizing ASL. This is encouraging

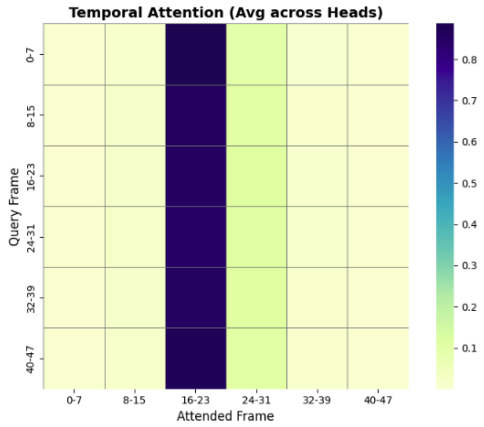

Figure 6. Transformer Temporal Attention for WLASL

because it suggests that the I3D backbone is extracting relevant spatial information aligned with the signing activity. However, the spread of saliency across both hands and the upper torso, with some noisy activation on the background, indicates that the model may still lack precise spatial localization. This could be due to the use of mean pooling across the entire temporal sequence, which flattens motion dynamics. Future improvements could involve incorporating more temporally-aware attention mechanisms or spatial masking to better isolate salient regions over time.

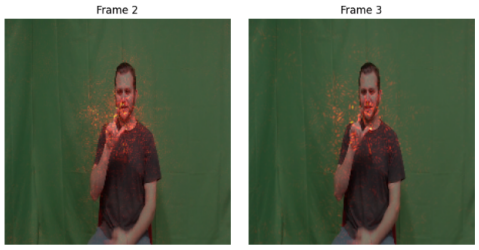

Figure 7. I3D saliency maps for How2Sign

GPT-2 Decoder Attention Maps. To better understand how the decoder utilizes visual information, we visualize the attention weights from caption tokens to the video-derived prefix embeddings in the final GPT-2 layer (Figure 8). The attention map reveals that most caption tokens assign low weights to the prefix tokens, indicating limited reliance on the visual input during generation. While a few tokens exhibit slightly elevated attention to specific prefixes, there is no clear structure or alignment to suggest meaningful temporal grounding. This supports our observation that the model often generates plausible but semantically incorrect captions, likely due to insufficient conditioning on the visual modality. These findings highlight a key limitation of prefix-only architectures for temporally complex tasks like sign language transcription.

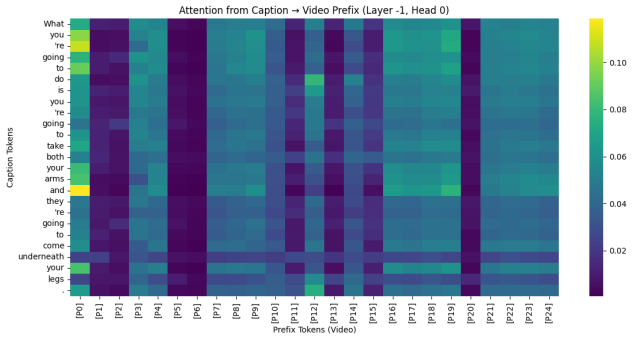

Figure 8. Token-to-Prefix Attention Map from GPT-2 Decoder

6. Conclusion and Future Work

In this project, we explored a range of feature representations and model architectures for isolated sign recognition on WLASL and full-sentence translation on How2Sign. Our results show that frame-based inputs consistently outperform skeleton and optical flow representations. On WLASL, the best performance (77% accuracy) came from combining a pretrained I3D100 backbone with a Transformer encoder, leveraging I3D's spatiotemporal features and the Transformer's ability to model long-range dependencies. For How2Sign, the highest BLEU score (7.25%) was achieved by integrating I3D features with a Transformer encoder and GPT-2 decoder, highlighting the benefit of combining strong visual encoders with pretrained language models for sentence-level translation.

Underperforming models, such as keypoint-only or optical flow variants, lacked the visual fidelity or temporal coverage needed for accurate recognition. Adding a Transformer encoder before GPT-2 sometimes reduced performance on smaller datasets due to overfitting, suggesting that deeper temporal modeling requires larger-scale training. Overall, our findings emphasize that (1) pretrained 3D visual backbones are critical for gesture understanding, (2) temporal attention mechanisms outperform vanilla LSTMs, and (3) sequence-to-sequence language models can effectively leverage high-level visual features for translation.

Looking ahead, future work could include training on larger, more diverse ASL datasets to reduce overfitting and allow end-to-end fine-tuning of the I3D and Transformer components. Incorporating multi-modal inputs (e.g., RGB, skeleton, and optical flow) into a unified architecture may further improve robustness. Additionally, aligning visual encoders with ASL-tuned language models and optimizing for real-time inference through pruning or distillation could pave the way for deployable ASL translation systems.

References

[1] A. Budria, L. Tarres, G. I. Gallego, F. Moreno-Noguer, J. Torres, and X. Giro-i Nieto. Topic detection in continuous sign language videos. *arXiv preprint arXiv:2209.02402*, 2022. 2

[2] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. Neural sign language translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7784–7793, 2018. 2

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1, 2

[4] T. F. Dima and M. E. Ahmed. Using yolov5 algorithm to detect and recognize american sign language. In *2021 International Conference on information technology (ICIT)*, pages 603–607. IEEE, 2021. 2

[5] A. Duarte, S. Palaskar, L. Ventura, D. Ghadiyaram, K. DeHaan, F. Metze, J. Torres, and X. Giro-i Nieto. How2sign: a large-scale multimodal dataset for continuous american sign language. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2735–2744, 2021. 1, 2, 3

[6] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, page 363–370, Berlin, Heidelberg, 2003. Springer-Verlag. 4

[7] S.-W. Gan, Y.-F. Yin, Z.-W. Jiang, L. Xie, and S.-L. Lu. Vision-based sign language translation via a skeleton-aware neural network. *Journal of Computer Science and Technology*, 40(2):378–396, 2025. 2

[8] M. Gupta, G. Singh, and A. Yadav. Cnn based speech and text translation using sign language. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 433–437. IEEE, 2021. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 3

[10] S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li, and Y. Fu. Skeleton aware multi-modal sign language recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3413–3423, 2021. 2

[11] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2, 3

[12] D.-H. Lee and J.-H. Yoo. Cnn learning strategy for recognizing facial expressions. *IEEE Access*, 11:70865–70872, 2023. 1, 2

[13] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 1, 2

[14] D. Li, C. Rodriguez, X. Yu, and H. Li. 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*, pages 1459–1469, 2020. 1, 2

[15] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). volume 81, 04 1981. 4

[16] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 2

[17] @NicholasRenotte. Action detection tutorial for sign language (youtube video). https://www.youtube.com/watch?v=doDUihpj6ro, 2021. Accessed: 2025-04-28. 2, 3

[18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1, 2

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2, 3

[20] sumedhsp. Sign language recognition (sign-language-recognition). https://github.com/sumedhsp/Sign-Language-Recognition, 2021. Accessed: 2025-05-05. 4

[21] J. Sánchez, E. Llopis, and G. Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 3:137–150, 07 2013. 4

[22] L. Tarrés, G. I. Gállego, A. Duarte, J. Torres, and X. Giró-i Nieto. Sign language translation from instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5625–5635, 2023. 2, 4