Sign Language Video Segmentation Using Temporal Boundary Identification

Kavu Maithri Rao and **Yasser Hamidullah** and **Eleftherios Avramidis**
{maithri.rao,yasser.hamidullah,eleftherios.avramidis}@dfki.de
German Research Center for Artificial Intelligence (DFKI GmbH)
Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Sign language segmentation focuses on identifying temporal boundaries within sign language videos. As compared to previous segmentation techniques that have depended on frame-level and phrase-level segmentation, our study emphasizes on subtitle-level segmentation, using synchronized subtitle data to facilitate temporal boundary recognition. Based on Beginning-Inside-Outside (BIO) tagging for subtitle unit delineation, we train a sequence-to-sequence (Seq2Seq) model with and without attention for subtitle boundary identification. Training on optical flow data and aligned subtitles from BOBSL and YouTube-ASL, we show that the Seq2Seq model with attention outperforms baseline models, achieving improved percentage of segments, F1 and IoU score. An additional contribution is the development of a method for subtitle temporal resolution, which automates the generation of time-stamped SubRip Subtitle (.srt) files. Our code and links to the datasets used in this research are publicly available at https://github.com/MaithriRao/Thesis.

1 Introduction

Sign languages are the primary means of communication among both hard-of-hearing and deaf individuals globally. Sign languages are gestural natural languages incorporating facial expressions, body movements and hand gestures to communicate and express meaning (Davis and Zajdo, 2010).

In Sign Language (SL) research, obtaining high-quality annotations that can be used for text-SL parallel corpora is a persistent challenge. In our study, we focus on the annotations that involve precise marking of the temporal boundaries of subtitle units within video recordings, which entails identifying exactly where one subtitle unit ends and another begins. Such annotations typically also include translations for the visual content. This entire process is demanding, time-consuming, and labor-intensive (Dreuw and Ney, 2008), significantly hindering the development and evaluation of robust SL recognition and segmentation systems.

In addition to the challenges of manual annotation, a key challenge in SL segmentation is precise temporal localization, which involves accurately identifying when linguistic components occur. This is particularly difficult because consecutive sentences can be signed with minimal or no pauses, making their boundary detection challenging.

In this work, we propose to segment SL video streams into subtitle units. A subtitle unit is formally defined as a contiguous temporal segment of video that precisely corresponds to a single, complete textual subtitle as provided in synchronized caption data, e.g. SubRip Subtitle (.srt) or Web Video Text Tracks (.vtt) files. This choice of segmentation offers several key advantages. Subtitle units are highly suitable for downstream applications such as machine translation and information retrieval. This is particularly beneficial for machine translation, where current systems often struggle with isolated short phrases and require longer, complete sentences to capture whole meaning and context. Secondly, automating SL video segmentation into subtitle units using human-curated data significantly alleviates the manual annotation bottleneck. This focus also crucially addresses the challenge of subtle transitions between linguistic components, by inherently providing clear boundaries for continuous signing. Furthermore, subtitle units provide an better-suited intermediate granularity, balancing the fine-grained, potentially noisy frame-level segmentation with the broader, often inconsistent, phrase-level segmentation.

Previous SL recognition studies focused on sign or word-level segmentation, isolating individual signs from pre-segmented clips (Chaaban et al., 2021; Renz et al., 2021a). However, continuous SL integrates sentences and phrases, making word-level methods insufficient for capturing full linguistic context. Segmenting into subtitle-like units is crucial for capturing complete linguistic context necessary for translation and interpretation.

Focusing on subtitle-level segmentation, we investigate the effectiveness of sequence-to-sequence (Seq2Seq) models with and without attention mechanisms for automated boundary detection, using optical flow features to integrate motion information, which has demonstrated efficacy in shallow models and action recognition tasks. Following state-of-the-art research (Moryossef et al., 2023), we adopt BIO (beginning-inside-outside) rather than IO tagging used in previous work. This choice allows us to better capture the precise start and end points of subtitle units, accommodating the smooth transitions often present in continuous signing, mirroring its benefits for sign and phrase segmentation. Our model is based on an a Seq2Seq encoder-decoder model with an attention mechanism, employing a bidirectional LSTM (BiLSTM) in the encoder, which analyzes the frame features in both forward and backward directions, enabling the model to capture both past and future context. Moreover, integrating an attention mechanism enables the model to focus on the most pertinent segments of the input sequence at each phase.

We evaluate our model on the BOBSL (Albanie et al., 2021) and YouTube-ASL (Uthus et al., 2023) datasets, demonstrating the effectiveness of our approach for subtitle-level SL segmentation. Our results show that the Seq2Seq model with attention outperforms baseline models, achieving improved percentage of segments, F1 and IoU scores. Furthermore, we find that the integration of BIO tagging is crucial for modeling subtitle boundaries, and that the Seq2Seq encoder-decoder architecture with attention mechanisms significantly enhances segmentation quality.

As part of our research, we also present an automatic method for subtitle temporal resolution, able to generate .srt files from model predictions including time-stamped segmentation. This method contributes to significantly facilitating and automating the annotation process for SL datasets.

2 Related work

In this section we are focusing on previous work seeking to determine boundaries between separate signs or linguistic parts. Farag and Brock (2019) address word boundary detection in Japanese Sign Language (JSL) by employing a binary random forest classifier on 3D joint positions. This frame-by-frame approach, evaluated on JSL and human activity datasets, achieves an F1 score of 0.89, effectively distinguishing between motion transitions and genuine gestures.

Renz et al. (2021a) explore automatic sign segmentation through two primary approaches. Initially, they propose a frame-level binary labeling method using I3D (Carreira and Zisserman, 2017) and MS-TCN (Farha and Gall, 2019), trained to minimize over-segmentation and reduce annotation costs. Building upon this, they introduce Changepoint-Modulated Pseudo Labelling for source-free domain adaptation, leveraging pseudo-labelling (Lee et al., 2013) to reduce model uncertainty in unlabelled data (Renz et al. (2021b)). Bull et al. (2020b) explore SL segmentation through spatio-temporal modeling and transformer-based approaches. Initially, they propose a method to automatically identify temporal boundaries using an ST-GCN (Yan et al., 2018) combined with a BiLSTM, trained on 2D skeleton data from French SL (LSF) videos (Bull et al., 2020a). Subsequently, Bull et al. (2021) introduce a system that uses Transformers to simultaneously segment SL videos and align them with subtitles, employing BERT (Devlin et al., 2019) for subtitle encoding and CNNs for video representation.

Moryossef et al. (2023) address the limitations of binary frame classification in SL segmentation by integrating linguistic cues and adopting the BIO tagging scheme (Ramshaw and Marcus, 1999), inspired by Named Entity Recognition, to better define segment boundaries. Their task is to perform segmentation of signs and phrases, for which they also utilize optical flow and 3D hand normalization. Evaluated on the DGS Corpus (Hanke et al., 2020), their model demonstrates improved cross-lingual generalization. Contrary to this work, that focuses on phrase-level segmentation, our work focuses on sentence-level and subtitle-level segmentation. We find this granularity (a) more appropriate for capturing complete meaning units, accounting for long-distance reording and other linguistic phenomena that require long context (b) better fit to real-world use-cases (e.g. captioning) and NLP tasks (parallel corpus creation, machine translation).

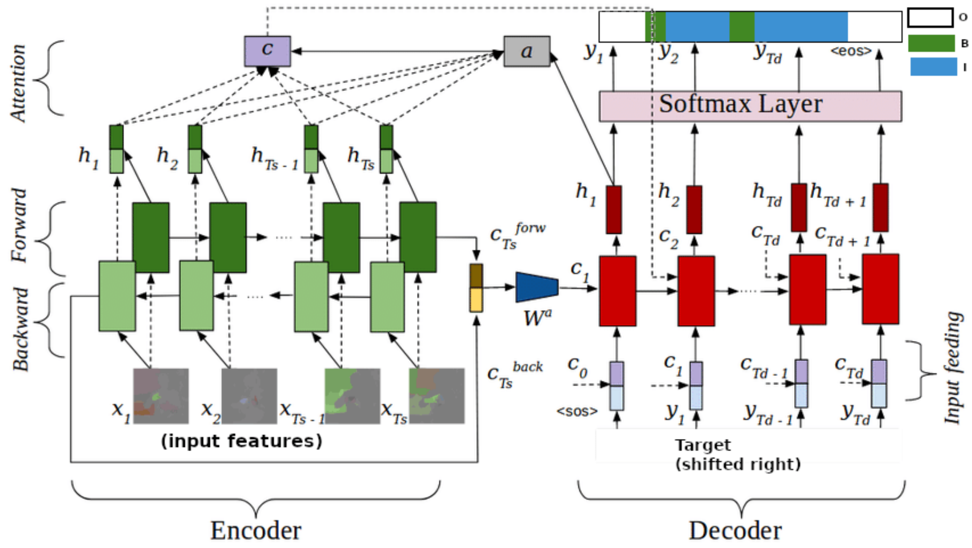

Figure 1: Seq2Seq Encoder-Decoder with Attention mechanism (Based on: Chowdhury and Vig, 2018)

3 Methods

3.1 Sequence-to-Sequence modelling

Our proposed approach for subtitle-level SL segmentation is based on a sequence-to-sequence model, which receives a sequence of input features derived from the SL video and outputs a sequence of respective subtitle tags.

Input features: Optical Flow We use the RAFT method (Teed and Deng, 2020) to estimate optical flow calculating pixel displacement between frames of a certain distance (in our case, 10 frames apart). This captures the detailed motion patterns which is provided as features to the Seq2Seq model for the boundary detection.

Output: BIO tags *Beginning-Inside-Outside* (BIO) tagging, is used to define and label segment boundaries (similar to Moryossef et al., 2023; Ramshaw and Marcus, 1999). The sentence boundary labels serve as target labels on the output of the decoder.

Consequently, we consider the following model variations:

Sequence Encoder and Autoregressive Encoder We adopt two encoder architectures to analyze feature sequences and capture temporal dependencies. A BiLSTM (Hochreiter and Schmidhuber, 1997) is employed to integrate preceding and subsequent context, capturing long-range dependencies. We integrate an autoregressive mechanism (Jiang et al., 2023; Moryossef et al., 2023), using two stacked encoders with sequential logit input for temporal coherence. Both encoder architectures serve as baselines.

Seq2Seq Encoder-Decoder without Attention We utilize a BiLSTM encoder and an LSTM decoder. The encoder analyzes the input sequence, producing context vectors (final hidden and cell states) that are transmitted to the decoder. The decoder subsequently generates output tokens derived from the preceding output and the encoder's final hidden state. However, this architecture depends on a static context vector, which may restrict its capacity to capture long-range dependencies.

Seq2Seq Encoder-Decoder with Attention A primary constraint of conventional Seq2Seq encoder-decoder systems is their difficulty in effectively handling long input sequences. This is due to the model's dependence on a single context vector of a predetermined length to store and transmit the information from the input sequence to the decoder. For long input sequences, the fixed-size context vector may have difficulty preserving all the required details, particularly those related to long-range dependencies, leading to a decline in output quality. To overcome this constraint, the attention mechanism (Bahdanau, 2014) is incorporated into Seq2Seq models, specifically designed for RNN-based architectures (Figure 1).

3.2 Subtitle Temporal Resolution

For subtitle file generation, where accurately identifying BIO tags is crucial, we employ sequence prediction methods. We find that beam search decoding with a beam width of 4 yields more precise and accurate model predictions compared to greedy search, after evaluating both methodologies. This process generates temporal interval tokens, indicating subtitle categories: no subtitle(O), start of subtitle(B), or continuation of subtitle(I). The key steps include:

a) The process starts by inputting a start token into the model, hence commencing the prediction sequence.

b) At each time step, we retain a collection of the leading sequences with the highest cumulative probability scores, limited to a certain beam width. In our experiments, we evaluated the beam widths 3, 4, 5 and 6, and determined that the beam width of 4 yielded optimal results for our purpose.

c) For every candidate sequence in the beam, the model predicts potential subsequent tokens, producing a probability value for each. The cumulative score of each sequence is updated, indicating the probability of that sequence.

d) Among all expanded sequences, the highest-scoring sequences (up to the beam width) are retained, while the others are eliminated.

e) The search continues until the end-of-sequence (EOS) token is reached.

f) Upon reaching the end of the sequence, the optimal sequence is determined by the highest cumulative probability.

Algorithm 1 is a post-processing algorithm that maps model predictions obtained earlier to frame boundaries, which can subsequently be converted into subtitle timing generation. The detailed steps are provided in the Appendix A.2.

3.3 Evaluation Metrics

F1 Score We compute the macro-averaged per-class F1 score at the segment level, using argmax to determine segment labels. This is our primary metric for validation, early stopping, and model selection.

Percentage of Segments (%) Following (Moryossef et al., 2023), we assess segment alignment accuracy by calculating the ratio of predicted segments to ground truth segments (1), with 100% indicating perfect alignment.

Input: $all\_predictions, all\_softmax\_outputs, sequence\_frames$
Output: $combined\_preds$: List of predictions with frame boundaries
Initialize $combined\_preds \leftarrow []$;
$current\_frame \leftarrow 0$;
foreach $(preds\_chunk, softmax\_chunk)$ *in* $(all\_predictions, all\_softmax\_outputs)$ **do**
  Initialize $probabilities \leftarrow []$;
  foreach $(pred, soft)$ *in* $(preds\_chunk, softmax\_chunk)$ **do**
    $probability \leftarrow soft[pred]$;
    Append $probability$ to $probabilities$;
  end
  $total\_prob \leftarrow \text{sum}(probabilities)$;
  $frame\_lengths \leftarrow \left[\frac{d}{total\_prob} \cdot sequence\_frames \ \forall\ d \in probabilities\right]$;
  foreach $(pred, length)$ *in* $(preds\_chunk, frame\_lengths)$ **do**
    Append $(current\_frame, current\_frame + length, pred)$ to $combined\_preds$;
    $current\_frame \leftarrow current\_frame + length$;
  end
end
return $combined\_preds$;

Algorithm 1: Probabilities to Subtitle boundaries

$$\% = \left(\frac{\text{Predicted Segments}}{\text{Ground Truth Segments}}\right) \times 100\% \quad (1)$$

Intersection over Union (IoU) IoU, as described in (Moryossef et al., 2023), measures segment overlap (2), indicating the model's ability to capture precise segment boundaries. A score of 1 signifies perfect overlap.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \quad (2)$$

Efficiency We evaluate the efficiency of each model based on parameter count and training time (55 epochs) using NVIDIA Tesla V100 and NVIDIA RTX A6000 GPUs.

4 Experimental Setup

4.1 Dataset

For our research, we employ the BOBSL and YouTube-ASL datasets. BOBSL comprises British Sign Language (BSL) interpreted footage from various BBC broadcasts, paired with English subtitles (Albanie et al., 2021), while the YouTube-ASL dataset provides a comprehensive collection of American Sign Language (ASL) videos with corresponding annotations (Uthus et al., 2023).

Model	Dataset	F1	IoU	%	# Params	Time
Sequence Encoder	**BOBSL**	**0.58**	**0.60**	**2.50**	1.38M	~ 14h
	YouTube-ASL	**0.56**	**0.58**	**0.70**	1.18M	~ 15h
Autoregressive Encoder	BOBSL	0.55	0.51	1.74	1.42M	~ 1d
	YouTube-ASL	0.47	0.50	0.55	1.26M	~ 1d

Table 1: Test evaluation metrics for our BOBSL and YouTube-ASL dataset using Sequence Encoder and Autoregressive Encoder model. A Comparative Analysis of F1, IoU and % of segments across Sequence Encoder and Autoregressive Encoder.

We use the manually-aligned subset of the BOBSL dataset, consisting of 60 videos, as other subsets exhibit inconsistencies. The videos, with a frame rate of 25 fps, are pre-divided into training (40 videos), validation (10 videos), and test (10 videos) sets. Most videos are either 30 or 60 minutes long, with an average duration of 45 minutes. This dataset features diverse genres, including comedy, drama, and entertainment, captures co-articulated signs, and offers a natural signing style. For the YouTube-ASL dataset, we use 70% of the dataset for training, 20% for validation, and 10% for testing. The videos in this dataset vary in duration, ranging from 40 seconds to 40 minutes, providing a diverse collection of lengths that supports effective model training and evaluation.

For our segmentation task, we preprocess video frames by resizing, normalizing, and grouping them into 375-feature segments based on annotations. This segmentation enables the model to learn temporal context and transitions, essential for accurate results.

4.2 Experiments

Our experiments are organized into 4 stages: feature extraction, baseline temporal modeling, and two variations of Seq2Seq encoder-decoder architectures. We first establish a robust feature representation using ResNet-101, then explore temporal modeling with BiLSTM and autoregressive encoders, and finally evaluate the segmentation accuracy of Seq2Seq models with and without attention.

Feature Extraction Given the different nature of motion data compared to RGB, training 2DCNNs from scratch is often preferred. However, due to our limited data relative to ImageNet, we employ transfer learning with a ResNet-101 model pre-trained on ImageNet (motivated by Yosinski et al. (2014)) for feature extraction.

As our objective is exclusively feature extraction rather than classification, we remove the final fully connected layer from the ResNet-101 model. An Adaptive Average Pooling layer is set to produce a constant spatial dimension in the network output. This setting guarantees the model's output will be a compact feature vector, irrespective of the input image dimensions. This layer generates a feature vector with the shape (2048,). Employing Adaptive Average Pooling enables preserving the high-level features of the ResNet-101 model, while normalizing the output dimensions to a vector format. The input dimensions for each image are (224, 224, 3), where 224x224 denotes the spatial dimensions and 3 indicates the number of channels for RGB images.

For BOBSL we use their pre-computed optical flow features as input, which have been processed through a ResNet-101 model to extract relevant features. For the YouTube-ASL we use RAFT (Teed and Deng, 2020) to estimate optical flow, calculating pixel displacement between 10 frames apart.

Sequence Encoder and Autoregressive Encoder For temporal modeling, 2048-dimensional feature vectors extracted from ResNet-101 are fed into a BiLSTM encoder. Each batch has 375 feature vectors, extracted from a single frame of the video segment. The sequence length is determined after testing multiple different values to achieve an appropriate balance between collecting temporal patterns and guaranteeing efficient processing. The BiLSTM encoder predicts BIO tags for each frame, classifying them as B, I or O of the subtitle, effectively segmenting the video into SL segments.

Similarly, an autoregressive encoder processes the 375 feature vectors, incorporating logits from the current time step as input to the next, enhancing temporal coherence in the BIO tag predictions.

Seq2Seq Encoder-Decoder without Attention In the Seq2Seq model without attention, the input consists of 2048-dimensional features from ResNet-101, with a sequence length of 375 frames. To optimize efficiency, sequences are sorted by length, avoiding padding tokens. The BiLSTM encoder processes these sequence, generating a context vector that summarizes the input. The LSTM decoder then uses this context vector to predict segments corresponding to "B" (beginning), "I" (inside), or "O" (outside) within the SL sequence.

Seq2Seq Encoder-Decoder with Attention Here a BiLSTM encoder (2 layers, 128 hidden units, dropout 0.2) encodes 375x2048 input sequences from ResNet-101. The decoder (2 LSTM layers, 128 hidden units, dropout 0.1) uses an attention mechanism to compute a weighted sum of the encoder outputs, forming a context vector (256 dimensions) at each decoding step. This context vector, combined with the previous output embedding (128 dimensions), is used to generate logits via a fully connected layer. A softmax operation is used to normalize these logits into a probability distribution over the output segments.

Further comprehensive details regarding our model training procedures, including specific hyperparameters, training time analysis, and the implementation of techniques such as Teacher Forcing and Scheduled Sampling, are provided in Appendix A.1.

5 Results

This section presents our experimental results, addressing several key aspects of subtitle-level segmentation.

5.1 Performance differences between Sequence Encoder and Autoregressive Encoder models in SL segmentation

Analyzing the performance in Table 1, the Sequence Encoder generally demonstrates superior segmentation quality compared to the Autoregressive Encoder across both datasets.

A notable pattern emerges in the segmentation behavior: for both models, the BOBSL dataset consistently leads to over-segmentation (250% for Sequence Encoder, 174% for Autoregressive Encoder), indicating that models tend to predict more segments than the ground truth. Conversely, the YouTube-ASL dataset results in under-segmentation (70% for Sequence Encoder, 55% for Autoregressive Encoder), where fewer segments are predicted. This disparity in segmentation tendency highlights differences in the annotation granularity between the datasets. While the Autoregressive Encoder typically involves a slightly higher parameter count compared to the Sequence Encoder, its training time is considerably longer.

To address the challenges posed by these segmentation tendencies and the dataset-specific behaviors, our subsequent work focuses on a refined subtitle-level segmentation strategy using Seq2Seq models. Due to inherent differences in dataset characteristics and the unique nature of our subtitle segmentation task, a direct quantitative comparison with previous SL segmentation work is not directly feasible.

Model	F1	IoU	%	# Params	Time
Seq2Seq Encoder-Decoder w/o attention	0.58	0.70	2.16	3.1M	$\sim$ 15h
Seq2Seq Encoder-Decoder w/ attention	**0.60**	**0.74**	**1.03**	7.8M	$\sim$ 2d

Table 2: Test evaluation metrics for our BOBSL dataset using the proposed Seq2Seq Encoder-Decoder model with and without attention. A Comparative Analysis of F1, IoU and % of segments across two models.

Model	F1	IoU	%	# Params	Time
Seq2Seq Encoder-Decoder w/o attention	0.55	0.58	0.87	3.1M	$\sim$ 19h
Seq2Seq Encoder-Decoder w/ attention	**0.60**	**0.62**	**0.95**	3.0M	$\sim$ 2d

Table 3: Test evaluation metrics for our YouTube-ASL dataset using the proposed Seq2Seq Encoder-Decoder model with and without attention. A Comparative Analysis of F1, IoU and % of segments across two models.

5.2 Seq2Seq Encoder-Decoder model with and without attention to improve segmentation of longer, multi-sentence videos

We evaluate the ability of Seq2Seq models, with and without attention, SL video segmentation. Using F1 score, IoU, and segment percentage on the BOBSL dataset, we compare model performance. The datasets' video lengths allow us to analyze each model's capacity to handle continuous SL sequences, focusing on performance differences and strengths.

For the BOBSL dataset as shown in Table 2, the Seq2Seq Encoder-Decoder without attention

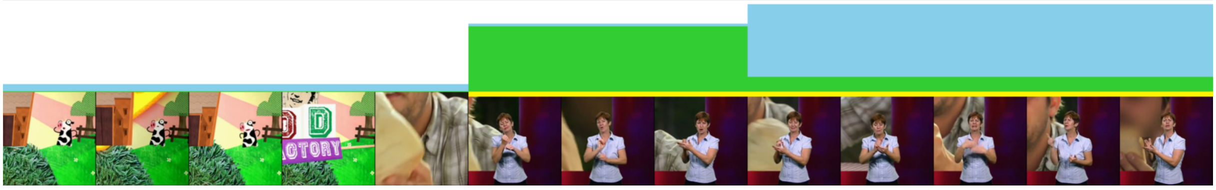

Figure 2: An illustration of subtitle-level segmentation approach, with a BOBSL test set, in **yellow, signing**: 'If you've ever baked your own bread, you probably prefer this to the supermarket bread.' Our attention based model effectively detects subtitle boundaries and segments with BIO tags. Here the **B tag (green)** represents the start of the subtitle, the **I tag (light blue)** for continuation, and the **O tag (white)** for outside of the subtitle segment. The model assigns these tags based on the predicted probability for each segment, effectively delineating the subtitle boundaries and segmenting the video.

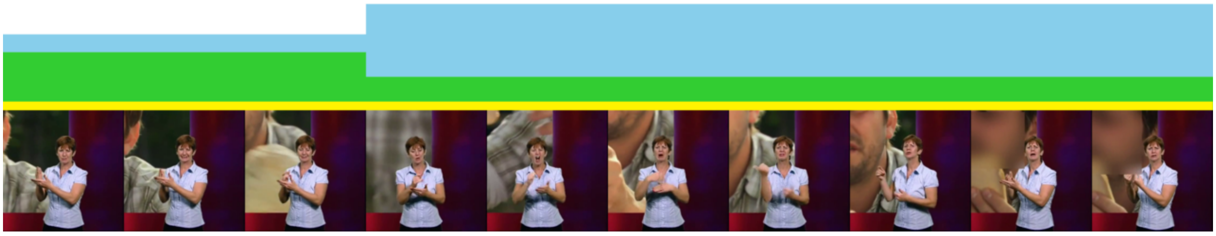

Figure 3: Continuation of the sequence from Figure 2, where the model correctly segments the new subtitle with the "B" and "I" tags as it moves smoothly between subtitles without pausing.

demonstrates moderate segmentation accuracy with an F1 score of 0.58 and reasonable overlap recognition with an IoU of 0.70, but exhibits significant over-segmentation, with a segment percentage of 216%. In contrast, the Seq2Seq model with attention attains an F1 score of 0.60, signifying moderate precision in identifying and segmenting relevant SL sequences. This is supported by an IoU of 0.74, highlighting the model's ability to identify overlapping regions between predicted and ground-truth segments. The model attains best segment percentage of 103%. The addition of attention increases the model's parameters to 7.8 million and training time to about 2 days, from 3.1 million parameters and 15 hours for the model without attention.

On the YouTube-ASL dataset as in Table 3, the Seq2Seq model without attention achieves an F1 score of 0.55 and an IoU of 0.58, indicating poor segmentation and overlap recognition. The model demonstrates under-segmentation, identifying only 87% of the segments. It has 3.1 million parameters and trains in 19 hours, suggesting optimization is needed. However, the Seq2Seq model with attention demonstrates a balanced performance with an F1 score of 0.60 and moderate overlap recognition (IoU: 0.62). The model identifies 95% of segments, indicating slight under-segmentation.

The observed performance differences between the datasets can be attributed to their distinct structural characteristics. For example, the BOBSL dataset consists of full sentences, where interpreters typically make clear pauses between them, aiding the model's segmentation task. In contrast, the YouTube-ASL dataset contains subtitles that may span across multiple sentences or include two sentences within a single subtitle, which may cause greater challenges for segmentation. This difference in structure could explain the model's superior performance on the BOBSL dataset, and it may be assumed that this structural difference affects the segmentation task on the YouTube-ASL dataset.

5.3 Effect of the subtitle temporal resolution to the quality of generated SL subtitle files

To assess the quality of generated subtitle files, we manually evaluate the model's accuracy in capturing subtitle timing and segmentation, as shown in Table 4. This table compares the actual and model-generated subtitle start and end times. This case study illustrates the model's overall performance on the BOBSL dataset, revealing its strengths and limitations in boundary detection, segmentation accuracy, and alignment with natural speech flow. The model demonstrates promising capabilities, achieving closer boundaries in specific segments, though perfect matches remain challenging.

The model effectively delineates subtitle boundaries in segments like [Subtitle 8, 9, 13, 14], closely

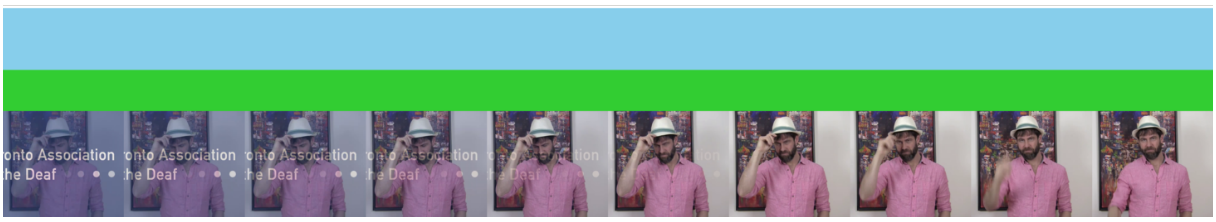

Figure 4: Failure instance in which the model incorrectly assigns a high probability to the "I" tag, indicating that signing activity is occurring.

Figure 5: Failure instance where the model incorrectly under-segments the subtitles, predicting a single subtitle instead of two distinct ones, thereby assigning a high probability to the "I" tag and indicating continuous signing activity.

aligning generated timings with actual subtitles. For example, Subtitle [8] and [9] correctly separate paused segments, while [13] and [14] accurately capture continuous signing. This demonstrates the model's ability to perceive subtle subtitle transitions beyond simple pauses. However, achieving exact timing matches is difficult due to our segment-level analysis, resulting in minor discrepancies. Furthermore, the model introduces temporal discrepancies in other segments, notably subtitles that succeed [10] and those before [13], leading to artificial interruptions and fragmented subtitles. This inconsistency in segmentation accuracy highlights the challenge of achieving frame-level precision without frame-level segmentation, and disrupting the natural flow.

5.4 Analysis of Model Performance and Error Categories

To gain a deeper understanding of our model's performance and limitations, we analyze its predictions, deriving insights into common patterns of success and distinct categories of errors. These findings are illustrated through representative examples.

Our Seq2Seq model generates probability scores for Beginning (B, green), Inside (I, light blue), and Outside (O, white) tags at every temporal step, visualized as distinct colored lines overlaid across the video's duration. The thin yellow bar above the video frame represents the ground truth temporal span of the subtitle unit where signing occurs. For the final segmented output, shown as the large, solid background color of each segment, the tag with the highest predicted probability is selected as the dominant label, indicating the model's most confident classification for that duration.

In Figure 2, the model accurately segments the BOBSL dataset video, correctly identifying subtitle boundaries. It accurately predicts non-signing periods (white "O" tag), the start of subtitle segments (green "B" tag), and the continuation of segments (light blue "I" tag). This demonstrates the model's ability to label the beginning and continuation of signing subtitles without false boundaries. Similarly, in Figure 3, the model effectively detects transitions between subtitles, even without pauses, using high probability scores for "B" and "I" tags. This highlights the model's ability to identify boundaries based on natural signing structure rather than just pauses.

Despite general efficiency, the model occasionally misidentifies subtitle boundaries, failing to consistently distinguish signing from non-signing activity. In Figure 4, the model incorrectly assigns a high probability to the "I" tag, indicating signing when there is none. This error may stem from feature ambiguity, where subtle motion in non-signing segments, such as raising and removing a hat, is misconstrued as signing. Additionally, an imbalance in training data may bias the model towards the "I" tag, particularly with minimal or unintentional movements. In Figure 5, the model under-segments, failing to recognize transitions between distinct signing periods, further highlighting the difficulty in distinguishing between signing and non-signing behaviors.

6 Conclusion

SL segmentation presents unique challenges due to its temporal and spatial complexity, including subtle transitions and variability across users. This study addresses subtitle-level SL segmentation using Seq2Seq models. A key contribution is an automated system for generating .srt subtitle files with accurate temporal boundaries. We adapt and improve the Encoder-Decoder model with attention specifically for subtitle-level segmentation. Utilizing optical flow and ResNet-101 features, our model enhances temporal alignment and transition management. Our focus on subtitle boundaries distinguishes our approach from frame-level studies. Our study conclusively demonstrates the efficacy of automated and precise subtitle-level SL segmentation, achieving strong F1, IoU, and segmentation accuracy. This marks a critical advancement for understanding and processing continuous sign language.

Future research could explore incorporating diverse input features like OpenPose, joint modelling of RGB videos and optical flow data, applying the model to synchronize subtitles with continuous signing, and testing on more varied sign language datasets to enhance generalizability.

Limitations

Our proposed approach, while effective, has several limitations. We haven't directly compared to the phrase-based SoTA but this is due to limitations of the available annotated datasets, and we are strong on our opinion that subtitle-level segmentation has clear advantages. Our evaluation is restricted to BOBSL and YouTube-ASL datasets with English subtitles, which may not adequately capture the linguistic diversity and intricacies of global sign languages, potentially limiting the model's generalizability as it has not been evaluated on datasets with greater variation. Furthermore, the model's primary reliance on optical flow makes it susceptible to noisy or inadequate motion data, such as during occlusions or subtle movements. Achieving a perfect one-to-one mapping between predicted and actual subtitle timing also remains a challenge. Finally, the study's reliance on manually labeled subtitle boundaries introduces potential noise and imprecision due to the inherent difficulty in their exact delineation.

Ethical Considerations

In our work, we present experiments on the British Sign Language and American Sign Language which should be seen and respected as the primary languages of the respective language communities. Although we perform this research aiming to provide equal access to language technology for sign language users, the fact that the majority of the researchers in NLP are hearing people entails the risk of developments that are not in accordance with the will of the respective communities, and therefore it is required that every research step takes them in constant consideration. In order to mitigate this, in our broader research we have included members of the Deaf/deaf and hard-of-hearing communities as part of the research team, consultants and participants in user studies and workshops and we have been in co-operation with related unions and communication centers. It should also be noted, that our experiments are part of a broader series of research projects, and the results presented here should be by no means considered ready for production nor used as final products without the agreement of the communities. The use of datasets follows their respective licenses and limitations and every follow-up work should adhere to those.

Acknowledgments

The research reported in this paper was supported by BMBF (German Federal Ministry of Education and Research) via the projects BIGEKO (grant no. 16SV9093) and SocialWear (grant no. 01IW20002). We would like to thank Prof. Dr. Josef van Genabith and Dr. Cristina España-Bonet for their support.

References

Samuel Albanie, Gül Varol, Liliane Momeni, Hannah Bull, Triantafyllos Afouras, Himel Chowdhury, Neil Fox, Bencie Woll, Rob Cooper, Andrew McParland, et al. 2021. BBC-Oxford British sign language dataset. *arXiv preprint arXiv:2111.03635*.

Dzmitry Bahdanau. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Hannah Bull, Triantafyllos Afouras, Gül Varol, Samuel Albanie, Liliane Momeni, and Andrew Zisserman. 2021. Aligning subtitles in sign language videos. In

Proceedings of the IEEE/CVF international conference on computer vision, pages 11552–11561.

Hannah Bull, Annelies Braffort, and Michèle Gouiffès. 2020a. Mediapi-skel-a 2d-skeleton video database of french sign language with aligned french subtitles. In *12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 6063–6068.

Hannah Bull, Michèle Gouiffès, and Annelies Braffort. 2020b. Automatic segmentation of sign language into subtitle-units. In *European Conference on Computer Vision*, pages 186–198. Springer.

Joao Carreira and Andrew Zisserman. 2017. 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.

Hussein Chaaban, Michèle Gouiffès, and Annelies Braffort. 2021. Automatic annotation and segmentation of sign language videos: Base-level features and lexical signs classification. In *16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021)*, volume 5, pages 484–491.

Arindam Chowdhury and Lovekesh Vig. 2018. An efficient end-to-end neural model for handwritten text recognition. *arXiv preprint arXiv:1807.07965*.

Barbara L. Davis and Krisztina Zajdo. 2010. The syllable in sign language: Considering theother natural language modality. In *The Syllable in Speech Production: Perspectives on the Frame Content Theory*. Taylor & Francis.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Philippe Dreuw and Hermann Ney. 2008. Towards automatic sign language annotation for the ELAN tool. In *Proceedings of the LREC2008 3rd Workshop on the Representation and Processing of Sign Languages: Construction and Exploitation of Sign Language Corpora*, pages 50–53, Marrakech, Morocco. European Language Resources Association (ELRA).

Iva Farag and Heike Brock. 2019. Learning motion disfluencies for automatic sign language segmentation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7360–7364. IEEE.

Yazan Abu Farha and Jurgen Gall. 2019. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584.

Thomas Hanke, Marc Schulder, Reiner Konrad, and Elena Jahn. 2020. Extending the Public DGS Corpus in size and depth. In *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*, pages 75–82, Marseille, France. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Zifan Jiang, Adrian Soldati, Isaac Schamberg, Adriano R Lameira, and Steven Moran. 2023. Automatic sound event detection and classification of great ape calls using neural networks. *arXiv preprint arXiv:2301.02214*.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta.

Amit Moryossef, Zifan Jiang, Mathias Müller, Sarah Ebling, and Yoav Goldberg. 2023. Linguistically motivated sign language segmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12703–12724, Singapore. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1999. Text Chunking Using Transformation-Based Learning. In Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, pages 157–176. Springer Netherlands, Dordrecht.

Katrin Renz, Nicolaj C Stache, Samuel Albanie, and Gül Varol. 2021a. Sign language segmentation with temporal convolutional networks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2135–2139. IEEE.

Katrin Renz, Nicolaj C Stache, Neil Fox, Gul Varol, and Samuel Albanie. 2021b. Sign segmentation with changepoint-modulated pseudo-labelling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3403–3412.

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer.

David Uthus, Garrett Tanzer, and Manfred Georg. 2023. YouTube-ASL: A Large-Scale, Open-Domain American Sign Language-English Parallel Corpus. *Preprint*, arXiv:2306.15162.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

A Appendix

A.1 Model Training

1. **Training Details:** We train the BiLSTM and autoregressive encoders using the Adam optimizer with a learning rate of 1e-4 and a batch size of 16. Gradient clipping with a clip value of 1 is applied to overcome the exploding gradient. We use the ReduceLROnPlateau, and an early stopping with patience=10 using both validation loss and the F1 score.

   We train Seq2Seq encoder-decoder models, both with and without attention mechanisms, for segmenting SL into subtitle units. Preliminary tests using cross-entropy loss resulted in overfitting, adopting the transition to Negative Log-Likelihood Loss (NLLLoss) for improved management of class imbalance. Our preliminary hyperparameter search involves testing a range of LSTM layers (2, 4, 6, 8), fully connected layers (1, 2), hidden sizes (128, 256, 512, 1024), dropout rates (0, 0.1, 0.2, 0.3), optimizers (SGD, Adam), learning rates (1e-3, 1e-4, 1e-5), and batch sizes (9, 12, 16), we conclude hidden size 128, 4 LSTM layers, 1 FC layer, encoder dropout 0.2, and decoder dropout 0.1, optimal to both YouTube-ASL and BOBSL datasets.

2. **Training Time:** To optimize training efficiency, we employ a two-stage process: pre-extracting ResNet-101 features from optical flow images and storing them for direct loading during training, thus reducing computational overhead. The Seq2Seq Encoder-Decoder without attention trains in 14-16 hours, whereas the attention-based model requires around one day. Training on the BOBSL dataset is faster due to its limited size, whereas the extensive YouTube-ASL dataset requires longer training times to achieve adequate convergence.

3. **Teacher Forcing and Scheduled Sampling:** Teacher Forcing, where the decoder receives actual target outputs during training, can result in over-dependence on ground truth labels and instability during inference. To mitigate this, we employ Scheduled Sampling. This method randomly alternates between using actual labels (teacher forcing) and model predictions as decoder inputs during training, enabling the model to adapt to prediction errors.

A.2 Algorithm to Map Probabilities to Subtitle Boundaries

1. **Model Predictions:** Collect raw predictions and their corresponding confidence scores (softmax probabilities) for each segment.

2. **Normalize Probabilities:** Compute the proportion of each prediction by dividing its probability by the total probability of all predictions in the sequence.

$$\text{Normalized Probability}_i = \frac{\text{Probability}_i}{\text{Total Probability}}$$

3. **Frame Allocation:** Assign frames to each segment using the normalized probability and the total number of frames in the sequence.

$$\begin{aligned}\text{Frames}_i = {} & \text{Normalized Probability}_i \\ & \times \text{Sequence Frames}\end{aligned}$$

4. **Frame Mapping:** Calculate the start and end frame for each segment iteratively.

$$\text{End Frame}_i = \text{Start Frame}_i + \text{Frames}_i$$

   Start the first segment at frame 0, and for subsequent segments, the start frame is the end frame of the previous segment.

5. **Convert to Time:** Map the calculated start and end frames to time using the frame rate (FPS).

$$\text{Time} = \frac{\text{Frame}}{\text{Frames per Second}}$$

Actual subtitle	Model generated subtitle
00:00:20.410 --> 00:00:21.813 Bug free?	00:00:20,930 --> 00:00:21,054 [Subtitle 8]
00:00:21.816 --> 00:00:22.676 No.	00:00:22,657 --> 00:00:24,055 [Subtitle 9]
00:00:22.774 --> 00:00:24.748 Insect free?	00:00:24,055 --> 00:00:26,047 [Subtitle 10]
00:00:24.748 --> 00:00:25.722 Brilliant.	00:00:26,047 --> 00:00:28,027 [Subtitle 11]
00:00:25.883 --> 00:00:31.710 Well, I'm going to reveal the secrets behind supermarket food, by making the ingredients that go into a sandwich.	00:00:28,027 --> 00:00:30,000 [Subtitle 12]
00:00:48.453 --> 00:00:55.707 If you've ever baked your own bread, you probably prefer this to the supermarket bread.	00:00:48,646 --> 00:00:55,638 [Subtitle 13]
00:00:55.707 --> 00:01:01.220 But the problem with this stuff is that it goes rock hard in a day or so, while the supermarket bread...	00:00:55,638 --> 00:01:01,140 [Subtitle 14]

Table 4: Comparison of Actual Subtitles with Model-Generated Subtitles for BOBSL dataset