Detecting Temporal Boundaries in Sign Language Videos

Loubna Ben Allal
ENS Paris Saclay
loubna.ben_allal@ens-paris-saclay.fr

Kamil Akesbi
ENS Paris Saclay
kamil.akesbi@ens-paris-saclay.fr

Abstract

This project is part of the Object Recognition and Computer Vision course of the MVA master. It falls in the field of sign languages. We want to perform automatic sign segmentation to detect boundaries between signs using Computer Vision techniques.

1. Introduction

Sign language automatic indexing is an important challenge to develop better communication tools for the deaf community. However, annotated datasets for sign language are limited, and there are few people with skills to anotate such data, which makes it hard to train performant machine learning models. An important challenge is therefore to :

- Increase available training datasets.
- Make labeling easier for professionnals to reduce risks of bad annotations.

In this context, techniques have emerged to perform automatic sign segmentation in videos, by marking the boundaries between individual signs in sign language videos. The development of such tools offers the potential to alleviate the limited supply of labelled dataset currently available for sign research.

In this project, we reproduce the results of paper [3] on the task of sign segmentation on the BSL dataset, and then try to obtain better performances by testing various transformer-based architectures. We first show the limited results obtained with a Vanilla Transformer encoder adapted and trained from scratch, which motivated us to test a new architecture named ASFormer, (2021) [6], that allowed us to outperform the results of paper [3].

The code of the project is available here : `https://github.com/kamilakesbi/MVARecVisProject`

2. Related work

Various algorithms have been developped for automatic sign segmentation. In [3], the authors use an Inflated Convolutional Neural Network (I3D) coupled with a Multi Stage-Temporal Convolutional Network [1].

The I3D model is used to extract features at a frame level for each video sequence. In this network, the filters and pooling kernels of 2D CNNS are expanded using an additional temporal dimension to get 3D CNNs. The features are then fed into a Multi-stage Temporal CNN (MS-TCN) [1], that consists of different stages of 1D dilated CNNs. The dilated CNNs improve each other's predictions sequentially, expanding the size of the temporal receptive field and thus allowing the model to capture long range dependencies between the video frames.

In order to improve the results, Transformers [5] seem to be a great next approach to test. Indeed, these models have shown good promise when dealing with sequential data, thanks to their attention mechanism that improves the modeling of relationships between data inputs. Recent studies have shown the great potential of Transformers for vision tasks, thus questioning the need for convolutional neural networks. The authors of [6] replace the MS-TCN architecture with a hybrid Transformer combining both the strengths of temporal dilated convolutions and self-attention mechanism. This model consists of an encoder and several decoders that aim to refine each other's predictions.

3. Data Preprocessing

3.1. Dataset

We use the **BSLCorpus** linguistic data, which is a subset of 72K gloss annotations (i.e signs with their categories and temporal boundaries). Each video consists of a sequence of frames. We used the same transformations as well as train and test splits used by the authors of [3] to compare the results. The training and test sets respectively consist of 5413 and 703 videos with an average of 80 frames per video.

3.2. The I3D features

In this project, we attempt to improve the results of sign segmentation on the BSLCorpus by changing the MS-TCN architecture. Therefore, we don't use raw video sequences,

but rather pre-extracted features, which consist of a 1024 dimensional vector representation for each frame obtained with the I3D model. This I3D model uses 3D CNNs, is pre-trained on Kinetics [2] data and finetuned on the BSLCorpus. The features are available on the Github repository of the original project [4].

4. Reproducing the MS-TCN results

4.1. MS-TCN architecture

We first used the code of [3] to reproduce the classification results obtained on the **BSL corpus**: the authors used an MS-TCN architecture trained in a supervised manner with the pre-extracted I3D features and boundary labels.

The model consists of several stages : each stage generates an initial prediction that is refined by the next stage. At each stage, several dilated 1D convolutions are applied on the activations of the previous layer. The model uses dilated convolutions because they increase the receptive field exponentially without the need to increase the number of layers.

During training, we minimize the sum of the losses over all stages.

4.2. Loss function

We use the same loss as in [3] in all our experiments. This loss is a combination of a classification loss and a smoothing loss (to reduce over-segmentation errors), and is classic for action segmentation problems :

- For the classification loss, we use a cross-entropy loss $\mathcal{L}_{cls}$.
- For the smoothing loss, we use a truncated mean squared error over the frame-wise log probabilities $\mathcal{L}_{T-MSE}$, where T is the video length.

The final loss function is thus :

$$\mathcal{L}_s = \mathcal{L}_{cls} + \mathcal{L}_{T-MSE}$$

4.3. Evaluation metrics

The **evaluation metrics** are the *mF1B* and the *mF1S*. These metrics are defined in [3] :

The *mF1B* measures the capacity to predict the position of the boundary. A boundary prediction is correct if its distance to a ground truth boundary is lower then a given threshold. By computing the average F1 scores obtained with different thresholds (integers in $[1, 4]$), we obtain the *mF1B*.

The *mF1S* measures the capacity to predict the extent of the sign segments using the F1 score, where sign segments with an IoU higher than a given threshold are defined as correct.

4.4. Results

We were able to reproduce the paper results, obtained on the **evaluation set** of the BSL corpus :

Dataset	mF1B	mF1S
BSLCorpus	68.68±0.6	47.71±0.8

5. Testing Transformer-based architectures

5.1. First model : the original Transformer encoder

5.1.1 Architecture

We implemented a transformer encoder to perform sign segmentation using the I3D frame features.

We used the **encoder architecture** from [5], (using adapted code from MVA's ALTEGRAD class) followed by a **linear classification layer** that labels boundary frames as 1 and interior frames as 0. We adapted the architecture's input format and padding masks.

We removed the first embedding layer of the encoder as the frames were already represented by features, and replaced it by a linear layer that reduces the input dimension from 1024 to either 512 or 256, otherwise the model had trouble converging.

5.1.2 Experiments

Regarding the implementation, we used the loss defined earlier similarly to [3]. We used ADAM optimizer with a small learning rate of $1e{-}5$, we tried using learning rate schedulers but they didn't improve the results. We also tried different architectures, by changing the number of encoder blocks, hidden dimensions d_{model}, feed-forward d_{fc} dimension as well as the number of attention heads. We first trained the model for 80 epochs, and by analyzing the mF1B curves of the training and test set, we noticed that the model overfits after a number of epochs, so we added Early-stopping (with patience 4). The table below shows the results of the architectures that gave us the best results:

layers	d_{model}	h	d_{fc}	params x10^6	mF1B eval	mF1S eval
2	512	8	1024	4.73	58.54	**41.76**
	256			1.84	59.75	39.32
4	512	8	1024	8.94	**60.28**	41.35
	256			3.42	59.63	40.31
8	512	8	1024	17.35	59.55	37.43
	256			6.58	59.65	39.67

Figure 2 shows the mF1B and mF1S of the model with 4 layers and d_{model} = 512, h = 8 and d_{fc} = 1024.

Note : With early stopping, the model training usually stopped around 10 to 15 epochs.

5.2. Second model : Transformer for action segmentation (ASFormer)

5.3. Architecture

Although transformers provide many benefits to sequential data modeling, they have some limitations when it comes to tasks such as action segmentation. Indeed, transformers lack inductive biases for small datasets compared to convolutional neural networks. Furthermore, the MS-TCN architecture benefits from prediction refinement since it involves multiple stages of CNNs, while the original transformer does not meet the refinement demand of action segmentation tasks. In ASFormer, the authors attempt to solve the previous problems by introducing the following modifications:

- They reduce the hypothesis space of the transformer model, by adding a dilated temporal convolutional layer that brings the local connectivity of the features.
- They use multiple decoders that refine each other's predictions, in a similar way to the MS-TCN stages.

Figure 1 shows the architecture of the encoder and decoders, for simplicity reasons we only show one decoder in the figure. This model works like MS-TCN, its inputs are the sequences of frame features.

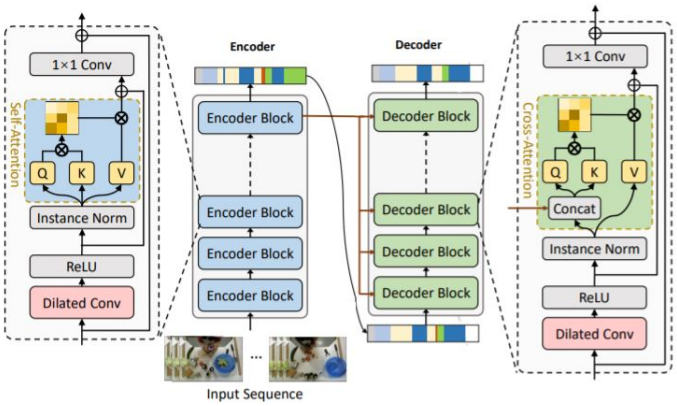

Figure 1. Simplified architecture of ASFormer showing the encoder and decoder formats, note that ASFormer uses more that one single decoder [6].

5.3.1 Experiments

Regarding the optimization, we kept the same scheme as [6], adapting the original code from the paper. The loss is a combination of Cross-Entropy with a smoothing loss slightly different from $\mathcal{L}_{T-MSE}$: it is the mean squared error over the frame-wise probabilities. We used a channel dropout rate of 0.3 and a learning rate of 0.0005 with Adam optimizer. We also used early stopping with patience 4.

ASFormer uses one encoder that outputs initial predictions, which are then refined by a sequence of decoders. The number of blocks inside the encoder and decoders as well as the number of decoders are crucial. Therefore we tuned these two parameters, and the table below shows the results of the best models we found, *I* stands for the number of decoders and *J* for the number of blocks:

I	J	mF1B eval	mF1S eval
3	5	68.89	47.80
	7	**69.79**	**49.23**
	9	68.14	45.72
4	7	69.62	45.11
	9	69.35	44.21

The model with 3 decoders and 7 blocks gives the best results. Figure 3 shows the training and validation curves for this model.

6. Final results and discussion

The table below shows the performances of our final models on the BSLCorpus. For the Transformer and ASFormer models we kept the architectures with the best performances on the evaluation set. We ran the ASFormer model with three different seeds and found a confidence interval 0.2 for meanF1B.

Model	mF1B eval	mF1S eval
MS-TCN	68.68 ± 0.6	47.71 ± 0.8
Vanilla Transformer	60.28 ± 0.3	42.70 ± 0.2
ASFormer	$\mathbf{69.79 \pm 0.2}$	$\mathbf{49.23 \pm 1.2}$

We can see that ASFormer performs significantly better than MS-TCN : on average, it outperforms the MS-TCN by a score of 1.11 in meanF1B and 1.52 in mean F1S. The pretrained weights of our best model can be downloaded from our public code repository (link in readme).

7. Conclusion

In this project, we worked on the problematic of sign language automatic indexing. It was a very collaborative work where both of us worked on each part. We reproduced the results of [3] on the BSL corpus using pre-extracted I3D features and the MS-TCN architecture. We further replaced the MS-TCN architecture with transformer based approaches, and showed that the AS-Former model outperformed the MS-TCN architecture by +1.11 meanF1B and + 1.52 meanF1B on average. We believe that this score could be further improved with more fine-tuning on the ASFormer model.

However, we think that the improvements made possible with Transformer based approaches are still small : it seems hard to increase the mean F1B by a huge margin. Further work could consider training both features and transformer based models at the same time.

8. Figures

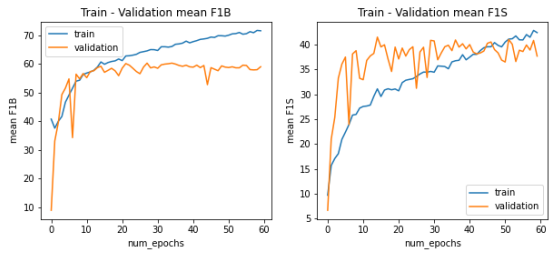

Figure 2. mF1B and mF1S training curves of Transformer encoder.

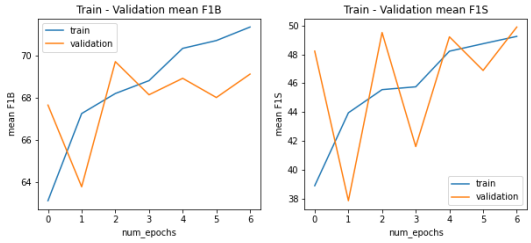

Figure 3. mF1B and mF1S training curves of ASFormer.

References

[1] Yazan Abu Farha and Jurgen Gall. 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, 2019.

[2] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[3] Katrin Renz, Nicolaj C Stache, Samuel Albanie, and Gül Varol. 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, 2021.

[4] Katrin Renz, Nicolaj C Stache, Samuel Albanie, and Gül Varol. Sign language segmentation with temporal convolutional networks. `https://github.com/RenzKa/sign-segmentation`, 2021.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[6] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. *arXiv preprint arXiv:2110.08568*, 2021.