Sign Language Translation from Video to Text

Report Final Project Recvis 2020: Topic F

Kodjo Mawuena Amekoe
École Normale Supérieure Paris-Saclay
kodjo.amekoe@ens-paris-saclay.fr

Souaidi Omar
CentraleSupelec
omar.souaidi@student.ecp.fr

Abstract

Sign Language Translation (SLT) first uses a Sign Language Recognition (SLR) system to extract sign language glosses from videos. Then, a translation system generates spoken language translations from the sign language glosses. In this project, we try to improve the performance of Sign Language Transformers [1] by explicitly or implicitly modelling the body pose (e.g. Hands, Face, Upper Body).

1. Introduction

Sign languages are languages that use the visual-manual modality to convey meaning and are expressed through manual articulations in combination with non-manual elements. The goal of sign language translation is to either convert written language into a video of sign or to extract an equivalent spoken language sentence from a video of someone performing continuous sign.

2. Problem Definition and Related Work

Neural Machine Translation (NMT) has made significant advances in the translation of sign languages. Recent work [1]introduce a novel transformer based architecture that joinly learns Continuous Sign Language Recognition and Translation (CSLRT) in an end-to-end manner. It performs the task of sign language translation on top of frame-level image features and outperforms previous NMT on the PHOENIX14T dataset. Recently, a first and new method to detect and estimate whole-body 2D/3D human poses, including bodies, hands and face, in the wild was introduced : DOPE [4]

One question is whether there is any benefit from explicitly modelling the body pose. Hence in this project we will first reproduce the method of [1] and set the performance as our baseline. Hence, we will try to see if adding features (Keypoints or Hidden Features from the CNN of DOPE) obtained with DOPE can improve the performance of our baseline.

3. Baseline

We re-implemented the code of [1] as a starting point. The results are shown in Table 1. We didn't get the best score with recognition loss weight λ_R equal to 5.0 as it is mentioned in the paper but with loss recognition weight set to 3.0 (We took this value for the rest of our project, and set the translation loss weight to 1.0). The results are not as good as those on paper, however the difference is small. We set our baseline performance on what we have been able to get by reimplementing the code. This work was carried out by both of us (Omar & Kodjo) to see if any of us could get better results.

Table 1: Reproduction of the results of [1]

λ_R	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
1.0	41.53	21.08	41.11	20.41
3.0	**29.78**	**21.36**	**29.40**	**21.02**
5.0	28.82	19.39	28.55	20.49
7.5	**28.18**	20.24	**28.11**	20.89
5.0 [1]	24.61	22.12	24.49	21.80

4. Additional Inputs

4.1. DOPE

We used a compressed version of the Phoenix2014 dataset. It contains about 8000 videos [Train $\approx$ 7000, Dev $\approx$ 500, Test $\approx$ 500]. For each video, we decompose it into several frames (with OpenCV) and for each frame we applied DOPE.

4.1.1 Keypoints

We modified the DOPE's code to lower the minimum score required to extract hands keypoints. The keypoints ex-

tracted are :

- Hands 2D & 3D
- Body 2D & 3D
- Face 2D & 3D

4.1.2 DOPE Hidden Features

We modified DOPE's code so that it can return not only the keypoints but also the features (that we are going to refer to by the DOPE Hidden Features) extracted by the ResNet50 NN. The architecture of DOPE is represented in figure 1, we got the the DOPE Hidden Features at the end of the Resnet50 block and just before the ROI Align. We could not just flatten the features (very high dimensionality) because it won't fit in memory, instead of doing that, we use an Adaptive Average Pooling to get for each frame a vector of dimension = 1024.
Note: We also tried to deblur the image using [2] & [3] to help DOPE extract the hand keypoints. Unfortunately, this attempt was unsuccessful.

4.1.3 Paragraphs

We also tried inputting a paragraph instead of a single sentence at a time. To generate paragraph, we have collected data from n successive videos such that the total number of frames does not exceed a certain threshold (threshold = 440, otherwise we have some memory's issues). The format of each file (video) is "DDMONTHS_YEAR_DAY_heute-XXXX.mp4" (XXXX is a number, 3067 for instance). A fixed DD, MONTHS, YEAR, DAY corresponds to one long video, and all the videos of the same reference are just splits of this one long video. For instance we can find in the train set [V-3067, V-3068, V-3069, V-3072, V-3073] (V stands for a specific DDMONTHS_YEAR_DAY_heute). As you can see the number 3070 and 3071 are missing, but they are not, they are in dev set or the test set, so in order to make a good comparison with our baseline, we didn't modify the dev set nor the test set.

This work was carried out by Omar.

5. Experimentation

All our experiments were performed on Google Colab using a TESLA V100. And we uploaded the code and results of all the experiments on github [1].
Before fusing, each additional input (keypoints or Dope Hidden Features) is passed to a Spatial Embedding Layer SEL (the same as the one used in [1]). The purpose of this NN is to normalize and embbed.

[1] https://github.com/omarsou/sign_language_project

5.1. Keypoints

5.1.1 Early Fusion

First, the fusion of the sign features (features obtained by the CNN of [1]) and the keypoints is made just before the encoder as shown in Figure 1. We used two methods for the fusion.

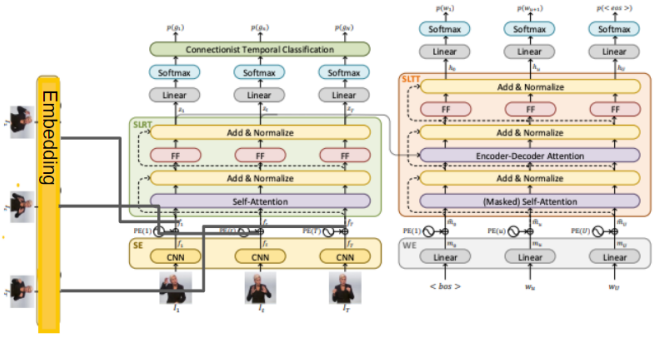

Figure 1: An overview of our early fusion architecture

Concatenation The first method is concatenation where we concatenated the keypoints embedding with sign features.
For instance, for one frame, a certain keypoints which is a vector of dimension (x, y) was flatten to obtain a vector of dimension (x*y,) and then passed to the SEL which embbed this vector in one vector of dimension x*y//n(x) (where n(x) = 2 if x = 2 and n(x) = 3 if x = 3), and then the concatenation with sign features result in a vector of dimension 512 + x*y//n(x). The results are shown in Table 2. No noticeable improvement.
This work was carried out by Kodjo.

Table 2: Early Fusion (Concatenation)

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
Body 2D	30.36	20.57	30.15	20.60
Hand 2D	30.54	20.42	30.88	19.88
Face 2D	30.91	19.28	30.80	18.63
Body 3D	29.65	**20.91**	30.15	**21.07**
Hand 3D	30.36	20.16	30.54	20.46
Face 3D	**29.15**	19.99	**29.19**	20.44
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

Addition The second method is a simple addition where the keypoint's vector of dimension (x*y,) is passed to the SEL which embed it to one vector of dimension 512 and

then we just add it to the sign features vector. The results are shown in Table 3. Overall the results are slightly better than one we get with the concatenation method. However, no noticeable improvement.
This work was carried out by Kodjo.

Furthermore, we take the best keypoints and try different combinations with the Addition Method. The table 4 represents the result we got.
This work was carried out by Omar.

Table 3: Early Fusion (Addition)

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
Body 2D	30.93	21.17	30.73	20.63
Hand 2D	30.21	20.71	29.73	20.58
Face 2D	31.33	20.95	30.97	**21.22**
Body 3D	32.27	20.60	32.40	19.04
Hand 3D	29.46	**21.56**	30.08	20.72
Face 3D	**29.38**	20.83	**29.56**	20.47
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

Table 4: Early Fusion Combination (Addition). H stands for Hand, B for Body and F for face.

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
H3D+ F2D+B2D	30.37	20.65	30.03	20.42
H3D + F2D	30.82	20.48	30.64	19.78
H3D+B2D	**28.10**	20.17	**29.09**	**20.69**
H3D+F3D+B3D	29.73	**21.00**	29.42	19.83
F3D+B3D	34.51	20.37	34.07	19.17
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

5.1.2 Late Fusion

Secondly, the fusion of the sign features and the keypoints is made after the encoder. We used only one method : the Addition. The results are shown in Table 5. No noticeable improvement.

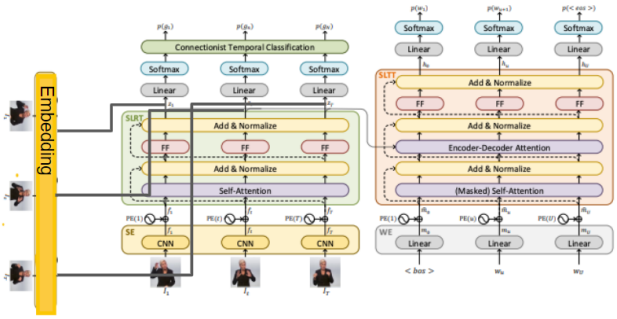

Figure 2: An overview of our late fusion architecture

Furthermore, we also tried different combinations between best keypoints. The results are shown in table 6.
This work was carried out by Omar.

Table 5: Late Fusion (Addition)

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
Body 2D	**28.69**	20.25	**28.36**	20.79
Hand 2D	29.76	**21.31**	29.28	20.49
Face 2D	30.80	20.31	30.59	20.20
Body 3D	33.76	20.60	32.82	**20.81**
Hand 3D	30.74	20.41	30.57	20.30
Face 3D	32.10	20.32	31.01	20.25
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

Table 6: Late Fusion Combination (Addition)

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
H2D+B2D	**28.08**	**21.40**	**28.36**	18.88
H2D + B3D	30.66	20.15	30.62	20.52
H2D+B2D+B3D	**30.50**	20.03	30.36	**20.46**
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

5.1.3 Conclusion

According to the results we got, we think that adding keypoints was like adding noise to the Data. Hence we could not get better results.

5.2. DOPE Hidden Features

5.2.1 Early Fusion

First, the fusion of the sign features and the DOPE Hidden Features is made just before the encoder. We used two methods for the fusion.

Concatenation The first method is concatenation.
For instance, for one frame, the DOPE Hidden Features which is a vector of dimension 1024 is passed to the SEL which embbed this vector in one vector of dimension 512 and then the concatenation with sign features result in a vector of dimension 1024.

Hidden Encoder The second method was to give to the encoder the DOPE Hidden Features as a Hidden states. Hence we modify the encoder so that we can apply the MultiHead Attention mechanism to the source with respect to the target (Here the source is DOPE Hidden Features and the target is the Sign Features).
The results are shown in Table 7. No noticeable improvements.
This work was done by Omar.

Table 7: Early Fusion (Feature). "Avg" stands for DOPE features that we obtain by applying an Adaptive Average Pooling and "Max" for features that we obtain by applying an Adaptive Max Pooling

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
Concat Avg (1)	**29.90**	20.17	**29.70**	19.07
Concat Max (1)	31.20	18.50	31.01	19.37
Hidden Avg (2)	35.75	**20.21**	35.40	**19.88**
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

5.2.2 Late Fusion

For the late fusion, we only used the Addition Method after the encoder. Except that this one, we weighted the addition. By taking the output of the SLE (The DOPE Hidden Features post SLE) and the output of the Encoder (Sign Features Post Encoder), the total output was : Output = α * sign_features_post_encoder + γ * DOPE_hidden_features_post_SLE . We tried different value of α & γ for a fixed loss recognition weight (LRW = 1.0). The results are shown in the table 8.

Then we took the one that performs best on the test set α = 1.0 and γ = 0.1 (thing that we shouldn't have done, as in real life, we don't have access to it) and we tried different loss recognition weight. The results are shown in table 9.
This work was done by Omar.

5.3. Paragraphs

Furthermore, we also tried inputting a paragraph instead of a single sentence at a time. The results are shown in table 10, as we can see, the results are much worse.
This work was done by Omar.

Table 8: Late Fusion (Feature)

	DEV		TEST	
λ_R = 1.0	WER	BLEU-4	WER	BLEU-4
$\alpha = 1.0$ $\gamma = 1.0$	**38.72**	**20.67**	**38.01**	20.19
$\alpha = 1.0$ $\gamma = 0.1$	40.43	20.57	40.39	**20.77**
$\alpha = 1.0$ $\gamma = 0.01$	**46.73**	20.12	44.82	20.21
$\alpha = 0.1$ $\gamma = 1.0$	71.92	17.94	69.92	17.55
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

Table 9: Late Fusion (Feature)

$\alpha = 1.0$ $\gamma = 0.1$	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
$\lambda_R = 1.0$	40.43	20.57	40.39	20.77
$\lambda_R = 3.0$	31.84	20.36	30.85	20.29
$\lambda_R = 5.0$	28.40	**21.16**	28.76	20.83
$\lambda_R = 7.5$	28.34	20.96	**27.71**	**21.67**
$\lambda_R = 10$	**27.70**	20.36	27.99	20.69
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

Table 10: Paragraphs

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
Paragraph	46.61	16.41	46.65	15.72
[1]	24.61	22.12	24.49	21.80
Baseline	**29.78**	**21.36**	**29.40**	**21.02**

5.3.1 Ensemble Model

Finally, we did an Ensemble Learning. First, we train a model using the Sign Language Transformers Architecture with only the DOPE Hidden features. We investigated the effects of having different numbers of transformers layers (same as in the official [1]). As it is shown in table 11 our best performance was obtained with 2 transformers layers. (Even if we had access to the result on the test set, we took only the one that performed best on the dev set)
In our next experiment, we examine the performance with various recognition loss weights with setting the number of transformers layers to 2. As it is shown in table 12, our best result comes with setting loss recognition weight to 7.5.
This work was done by Omar.

Ensemble We had two outputs, the gloss probabilities at the end of the encoder (after the linear layer and before

Table 11: DOPE Hidden Features SLT : Impact of different numbers of layers)

	DEV		TEST	
Layers	WER	BLEU-4	WER	BLEU-4
2	**72.94**	**14.16**	**70.44**	**13.15**
3	76.78	13.57	74.17	13.13
4	76.41	13.91	73.87	12.63

Table 12: DOPE Hidden Features SLT : Impact of different loss recognition weight

	DEV		TEST	
LRW	WER	BLEU-4	WER	BLEU-4
$\lambda_R = 1$	72.94	14.16	70.44	13.15
$\lambda_R = 3$	58.63	15.42	56.47	14.86
$\lambda_R = 5$	**49.75**	**16.12**	**48.84**	15.60
$\lambda_R = 7.5$	50.73	15.87	50.48	**16.35**

the softmax activation) and the decoder output for the translation (after the linear layer and before the softmax activation). We took the best SLT trained on sign features (our baseline) and the best SLT trained on DOPE Hidden Features and we merge them so that :

$$GPE = \frac{GPB + \gamma \times GPD}{1 + \gamma} \quad (1)$$

with GPE the gloss probabilities of the ensemble, GPB the gloss probabilities of the baseline. and GPD is the gloss probabilities of DOPE.

$$DOE = \frac{DOB + \gamma \times DOD}{1 + \gamma} \quad (2)$$

where:

- DOE is the the decoder output of the ensemble
- DOB the decoder output the baseline
- DOD is the decoder output of DOPE

We try different value of γ, the results are shown in table 13.

Since DOPE has never seen sign video before, the gloss recognition is bad and maybe we should have set the recognition loss weight much higher. So for our final result and comparison, we choose the set the γ to zero for the gloss probabilities so that for the gloss prediction the model takes into account only the gloss probabilities given by the baseline. Hence, we have that :

$$GPE = GPB$$

Table 13: Ensemble SLT

	DEV		TEST	
gamma	WER	BLEU-4	WER	BLEU-4
$\gamma = 1.0$	59.27	21.86	59.17	20.84
$\gamma = 0.5$	52.60	**22.50**	51.96	**21.81**
$\gamma = 0.1$	32.91	22.06	33.20	21.12
$\gamma = 0.05$	**30.80**	21.86	**30.78**	21.59

Table 14: Final Results

	DEV		TEST	
	WER	BLEU-4	WER	BLEU-4
Final SLT Ensemble	29.78	**22.50**	29.40	**21.81**
[1]	**24.61**	22.12	**24.49**	21.80
Baseline	29.78	21.36	29.40	21.02

The final results and comparison are shown in Table 14. Our ensemble model outperformed our Baseline Model and slightly outperformed the original SLT model.
This work was done by Omar.

5.3.2 Conclusion

By doing an ensemble learning (SLT with DOPE Hidden Features and SLT with sign features), we were able to outperform both our baseline's model and the paper's original model. As future work, we could extract the DOPE Hidden Features from the original dataset [1] (the one that is not compressed) and take the weights of the original model that gave the best result as mentioned in[1], and do the same experiment as in 5.3.1.

References

[1] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation, 2020. 1, 2, 3, 4, 5

[2] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. *ArXiv e-prints*, 2017. 2

[3] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. 2

[4] Philippe Weinzaepfel, Romain Brégier, Hadrien Combaluzier, Vincent Leroy, and Grégory Rogez. Dope: Distillation of part experts for whole-body 3d pose estimation in the wild, 2020. 1