Object recognition and computer vision - Project Report
Sign Language Translation from Video to Text

25 January 2020

Samuel DIAI
Ecole nationale des ponts et chaussées
samuel.diai@eleves.enpc.fr

1. Introduction

Translating sign language is not an easy task. Indeed, sign language uses multiple asynchronous information channels. There is two different types of channels : Manual features (shape, movement and pose) and Non-manuals features (facial expression, mouth and movement of the head, shoulders and torso).

The latter is often neglected, and previous state of the art models used to focus only on manual features. But this isn't enough to capture all the information, and the meaning can be completely different if one omit non-manual features. Thus a viable sign language translation machine must take both manual and non-manual features into consideration.

2. Subject

For this project, I will work with a transformer based architecture that jointly learns Continuous Sign Language Recognition and Translation [2]. This article introduces intermediate representation : "sign glosses" which are minimal lexical items.

The architecture is composed of a Sign Language Recognition Transformer (SLRT), an encoder transformer model predicting sign gloss sequences from spatial embeddings extracted from sign videos. It allows us to learn spatio-temporal representations from the videos. Then, these representations are used in a Sign Language Translation Transformer (SLTT), an autoregressive transformer decoder model, which is trained to predict the text translation.

If this model estimates some body pose (hands, face, upper body) using the sequences of images. It does not take the complete body pose (2D and 3D representations) as an input.

Thus, we will try to see if adding the complete body pose estimation improves the translation and recognition performance.

3. Reproducing the results

In this part, I tried to understand the SLT architecture and to create a baseline that will be useful for the model with the pose estimations. I used the code provided by the authors of [2] : source code

3.1. SLT Architecture

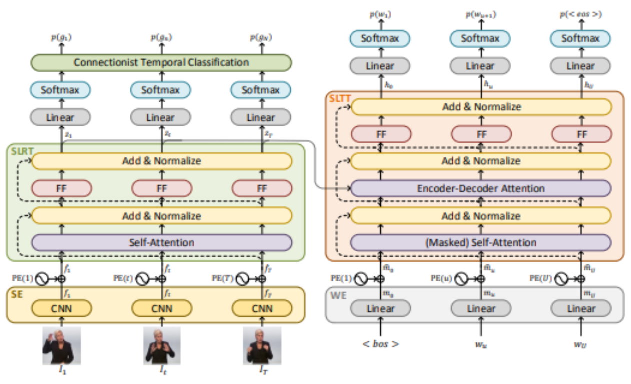

Figure 1: Architecture of the SLT [2] (with a single encoder-decoder layer)

The model is trainable in an end to end manner : it first recognizes and understands the information contained in the sequences of sign language videos. It learns the glosses conveyed by the sign langage with a Transformer Encoder (SLRT). It then computes the loss of the SLRT $\mathcal{L}_R$ using a Connectionist Temporal Classification (CTC) loss (it basically learns the probability of the glosses given the video inputs). Then the second part of the architecture does the translation task. It uses a Transformer Decoder (SLTT) (with multi head attention), which tries to learn the predict text sentences using the real inputs text sentences and the gloss outputs from the encoder. It finally compute the SLTT loss $\mathcal{L}_T$, which is a cross entropy loss for each word. Thus, the global loss of the model can be computed as a weighted sum of the above losses : $\mathcal{L} = \lambda_R \mathcal{L}_R + \lambda_T \mathcal{L}_T$.

	DEV					TEST				
	WER	BLEU-1	BLEU-2	BLEU-3	BLEU-4	WER	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Sign2Gloss	**30,18**	-	-	-	-	**30,11**	-	-	-	-
Sign2Text	**-**	**43,82**	**30,75**	**23,66**	**19,29**	**-**	**42,88**	**30,02**	**22,81**	**18,43**
Recog Sign2(Gloss+Text)	**29,09**	**44,88**	**32,47**	**25,29**	**20,72**	**28,9**	**44,11**	**31,72**	**24,52**	**20,05**
Trans Sign2(Gloss+Text)	**30,22**	**44,4**	**32,36**	**25,37**	**20,8**	**29,75**	**45,86**	**33,32**	**25,85**	**21,14**

Table 1: Overall performances of the SLT architecture

3.2. Tuning the weights of the loss

In order to set my baseline results, my first experiment was to tune the weights of loss:

Loss Weights		DEV		TEST	
λ_R	λ_T	WER	BLEU-4	WER	BLEU-4
1	0	30,18	-	30,01	-
0	1	-	19,29	-	18,43
1	1	38,32	20,43	38,13	20,69
3	1	30,48	20,46	30,36	20,94
5	1	30,22	**20,8**	**28,9**	**21,14**
10	1	**29,09**	20,72	29,75	20,05

As we can see, the best results are shared between the two configurations $\lambda_R = 5, \lambda_T = 1$ or $\lambda_R = 10, \lambda_T = 1$, but the first configuration seems to be better overall : it gives the best results on the DEV and TEST sets for the translation, and gives the best result on the TEST set for the recognition. In the paper [2] from camgoz and al, they find the same configuration, but with better results overall. In the following sets or results, I will use this configuration : $\lambda_R = 5, \lambda_T = 1$.

3.3. Tuning the number of encoder/decoder layers

In the architecture and the code provided by [2], it is possible to stack several encoder layers for the SLRT and similarly to stack several decoder layers for the SLTT. Increasing the number of layers will allow the network to have more weights and thus to be able to learn more complex features. However, a too wide network might suffer from vanishing gradients. There is a trade-off to optimize. As they did on the paper, I trained a SLT network with $\lambda_R = 1, \lambda_T = 0$ to see the impact on the recognition task.

Layers	DEV		TEST	
	WER	del / ins	WER	del / ins
1	34,37	23,51 / 2,78	33,76	22,09 / 3,08
2	29,86	16,89 / 2,38	29,56	15,43 / 2,65
3	**28,18**	**14,28 / 2,24**	**28,41**	**15.23 / 2,45**
4	30,58	16,73 / 2,99	29,87	15,38 / 2,89
5	29.77	15,10// 2.44	28,9	14,06 / 2,25
6	33,6	22,6 / 2,56	32,87	21,18 / 2,42

I noticed that the optimal number of layers was 3 with the lowest WER score. They also made the same observation. Then for the rest of report, my configuration will be with this setting.

3.4. Overall performance

Finally, I tried to obtain the best results possible and compare them with their results. With the provided code, I was able to run the Sign2Gloss models ($\lambda_T = 0$), the Sign2Text models ($\lambda_T = 0$) and the sign2(Gloss + Text) models ($\lambda_T \neq 0, \lambda_R \neq 0$).
The results are showed on the table 1 above. We can notice that my best models Sign2(Gloss + Text) have on the TEST set a 28.9 WER score for the Recognition task and a 21.14 BLEU-4 score for the Translation task. The results are quite significantly worse than the results of the article : They have on the TEST set a 24.49 WER score for the Recognition task and a 21.32 BLEU-4 score for the Translation task. I observed this phenomenon across all my experiments (even with the pose estimations) : I had a noticeably higher WER score than them. However, my results are not far for the Translation scores and I observed the same qualitative results (number of layers, loss weights).

4. Adding Pose estimations

The second part of my work was to add the the pose estimations of the different parts of the body : hands, face, torso. In theory, adding this more precise information should increase the overall performance of the model.

4.1. Extracting Pose estimations Keypoints

In order to extract the keypoints of the pose estimation (in 2D/3D), I used the suggested DOPE algorithm [3]. This complex model allows us to extract 42 keypoints for the hands (21 per hand), 84 keypoints for the face and 13 keypoints for the torso. The algorithm often finds several keypoints per body part. In this case I took the keypoints with the highest probability. If no keypoints are found for a body part, I use an empty vector as the keypoints. Then, for each image, I center the body parts (remove the mean of the keypoints).

I noticed that torso and head were found almost every time : heads keypoints exist in $98,25\%$ of the images and torso keypoints exist in $97,87\%$ of the images. However, the results are very bad for the hands : in 45.27% of the cases no hands are found, in 40.11% of the cases only one hand is found, in 6.13% of the cases the two hands are found. As we can see on the images of the annex, hands are detected only if they are steady. It doesn't seem to work when they are moving (when they are blurred).

4.2. Fusion methods

With the processed keypoints (in 2D and 3D) I was able to implement various methods to feed them to the SLT network. I tried 2 methods suggested by [1] namely early fusion and late fusion and another which I invented myself : mid fusion.

- The simplest method is the **Early fusion**. It consists in merging the inputs from various channels (here the images and the 3 keypoints types) before feeding them to the SLT. I merged the channels just before doing the positional encoding.

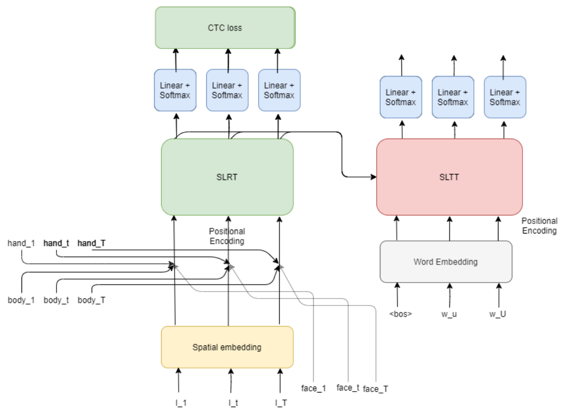

Figure 2: Early fusion

- I also implemented **Late fusion**. It consists in passing the inputs of the different channels separately, then we merge the output of the decoders using linear layers. (We also merge the outputs of the encoders to generate the probability glosses).

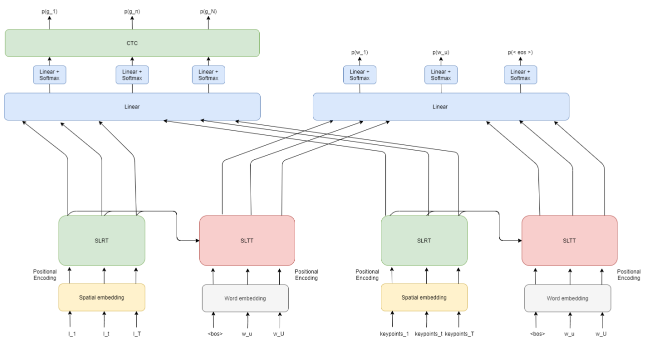

Figure 3: Late fusion

- Finally, I added **Mid fusion**. As in late fusion, it consists in passing the inputs of the different channels separately. The difference is that we merge the results at the output of the encoder. In this setting we have only one SLTT instead of multiples for the late fusion.

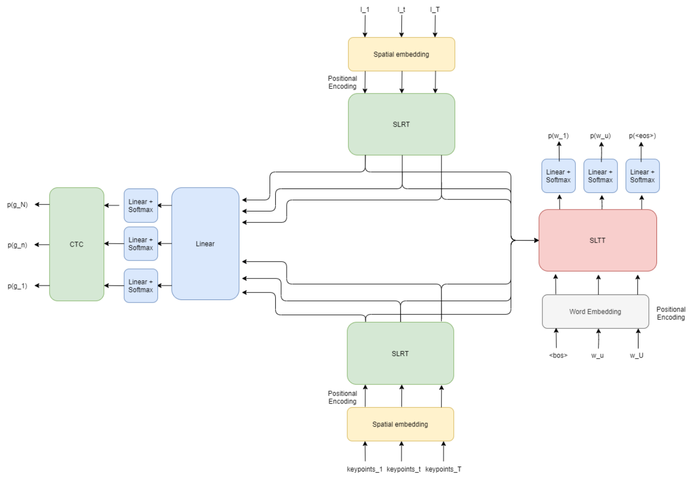

Figure 4: Mid fusion

The results are displayed on the table 2 below. We can observe quite poor performance for the models with only the keypoints. We could have expected these results given the poor number of hands detected. However some models seems to perform slightly better than the baseline : On the DEV set **Mid fusion** always perform better than the baseline. On the test set **Late fusion** seems to perform the best but the results do not outperform the baseline. Finally, 2D and 3D keypoints seem to produce roughly the same results.

5. Conclusion

After reproducing the results of the paper, I noticed the same qualitative results on the tuning of the parameters but different quantitative results : My result scores were systematically bellow theirs. I suppose it's because they don't give the exact configuration and they may have run much more test than I did. Then, I adapted their code and implemented some variations to incorporate the body pose estimations that I had previously recovered. No method clearly stands out from the other but mid and late fusion seem to give the best result. On some cases the models beat the baseline, but I'm not sure if this is significant. Finally, one solution to increase noticeably the performances would be to improve the extraction of keypoints. We could try to use a video-based pose estimation algorithm, i.e. which uses past knowledge of the pose estimations from one frame to another, instead of doing pose estimation independently for each image. This would prevent hands keypoints from disappearing from one frame to the next.

fusion method	keypoints dimension	DEV					TEST				
		WER	BLEU-1	BLEU-2	BLEU-3	BLEU-4	WER	BLEU-1	BLEU-2	BLEU-3	BLEU-4
early fusion	2D	34,35	43,47	31,06	23,75	19,15	33,98	43,09	30,81	23,64	19,24
late fusion		30,05	44,88	32,15	24,83	20,12	30,41	**45,23**	**32,81**	**25,42**	**20,69**
mid fusion		**29,84**	**45,07**	**32,74**	**25,41**	**20,73**	**29,96**	44,04	31,45	24,04	19,43
only keypoints		89.89	21.61	13.56	9.87	7.55	90.12	20.97	12.59	9.51	7.34
early fusion	3D	**28,45**	45,84	33,09	25,7	20,83	**29,21**	45,2	32,69	25,16	20,41
late fusion		29,3	**45,6**	32,64	25,1	20,35	30,19	**45,48**	**32,96**	**25,37**	**20,5**
mid fusion		29,54	45,64	**33,19**	**25,77**	**21**	29,77	44,97	32,4	25,12	20,47
only keypoints		88.57	20,42	13.89	10.59	7.68	89.57	21.62	13.97	10.59	7.93
Baseline	-	**30,22**	**44,4**	**32,36**	**25,37**	**20,8**	**29,75**	**45,86**	**33,32**	**25,85**	**21,14**

Table 2: Overall performances of the SLT architecture with keypoints of the pose estimation

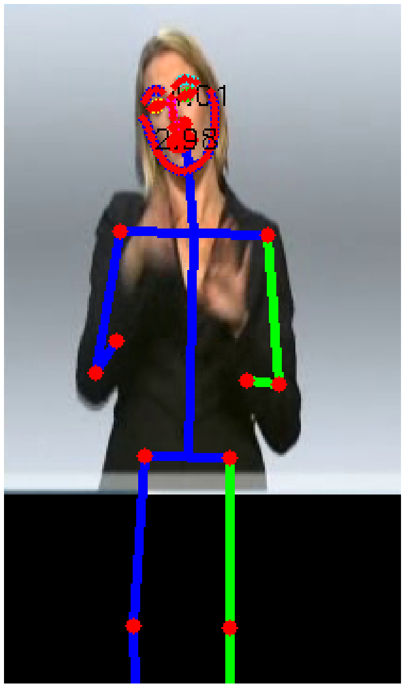
Figure 5: No hands detected

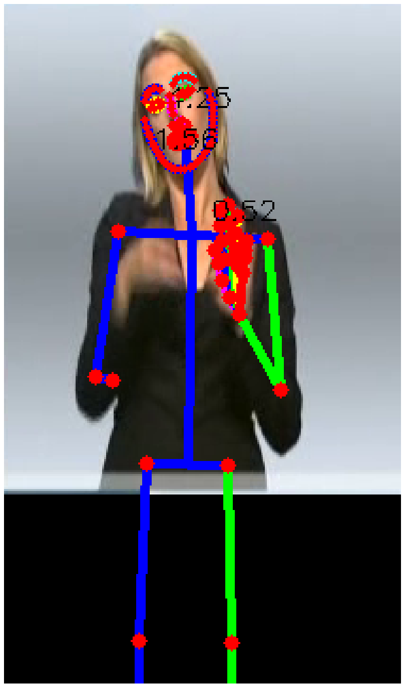
Figure 6: One hand detected

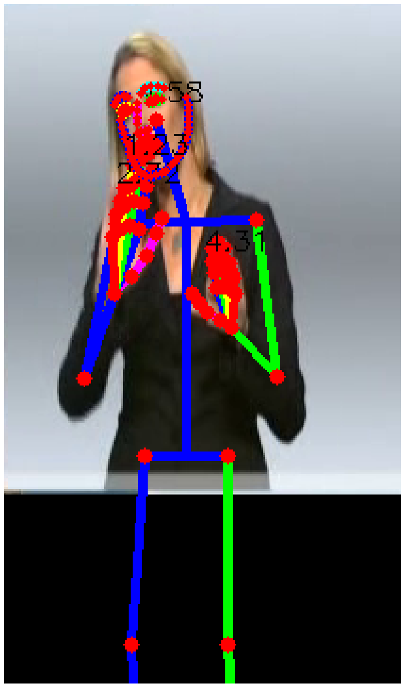
Figure 7: Both hands detected

Sequence of images and hand detections

Code can be found at this link : https://github.com/SamuelDiai/slt.

References

[1] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Multi-channel transformers for multi-articulatory sign language translation, 2020. 3

[2] 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] 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. 2