Comparing End-to-End and Pipeline Neural Approaches for Machine Translation from NGT to Dutch

Hristov, Kristiyan D.

Award date:
2022

Comparing End-to-End and Pipeline Neural Approaches for Machine Translation from NGT to Dutch

Kristiyan Danielov Hristov

supervisor: dr. Dimitar Shterionov

second reader: dr. Juan Sebastian Olier

30 June 2022

Abstract

Access to information is one of the key human rights in the modern world. However, information is most often distributed via a spoken language (either in textual or in audio format). This way of information dissemination is often limiting for deaf people. Deaf people communicate in Sign Languages (SL) which are fully-fledged languages in the visual-gestural modality. In the Netherlands, the primary SL is the Sign Language of the Netherlands or Nederlandse Gebarentaal (NGT).

Automatic sign-to-text translation is a challenging task. First, there are limited data resources available. Next, data are often collected in the scope of different projects and do not always follow the same formatting or annotation protocols. Third, SLs exploit manual and non-manual features to convey meaning, hand and body movements and facial expressions, as well as the space around the signer, making SL recognition a computational challenge. In addition, SLs have no officially accepted written form making SL transcriptions inconsistent. In this work we explore the power of Deep Learning to learn multi-dimensional features without their explicit definition by investigating end-to-end neural machine translation models for translation from NGT utterances (in a video format) to text.

Contents

List of Figures

Chapter 1

Introduction

Access to information is one of the key human rights in the modern world. However, information is most often distributed via a spoken language (either in textual or in audio format). This way of information dissemination is often limiting for deaf people. Deaf people communicate in Sign Language (SL). Among hearing people, a misconception is that there is only one SL. However, there are more than 300 SLs in the world[1]. Most of the time the only people who learn and use SLs are deaf or hard of hearing, and their families and close people. SL interpreters are currently the only way for signers and non signers to cross the language barrier. As such, the exchange of information between signers and non signers is limited by the availability of such professionals, which is the case even for small, informal conversations. In 2021, two European projects have started working on automatic translation tools for sign and spoken languages aiming to reduce the communication gap.[2]

Following the advances in deep neural networks, current state-of-the-art in automatic translation for spoken translation is achieved through recurrent or transformer networks trained on large amounts of data [1],[2],[3]. Advanced models reach unprecedented translation quality nowadays [4],[5]. But automatic translation between sign and spoken languages is in its infancy due to the many differences between sign and spoken languages, i.e. not only the different modalities, but also differences from syntactic and semantic nature [6]. The typical process for sign to spoken language automatic translation involves a series of transformations from a video (a recorded message in SL) into text or audio. This pipeline involves many challenges [7]. One of them is related to the different models that have to collaborate. Another is related to the different modalities (video for sign languages, text or audio for spoken language), as well as the fact that SLs have no officially accepted written form. Another one is related to the amount of collected data - there are simply too few resources that can be employed in a machine learning process to create effective models. Many sign languages have been accepted as official languages very recently. For example, Bulgarian Sign Language (BGSL) was officially accepted in 2021. The Sign Language of the Netherlands (NGT) was officially recognized in October 2020.

In this project we aim to investigate two different deep learning approaches adopted from spoken language automatic translation for the task of translating between sign and spoken languages. Our goal is to establish which approach is better suited for the task of SL machine translation (SLMT). So we formulate our research question as follows:

[1]https://www.ethnologue.com/subgroups/sign-language

[2]These projects are SignON (www.signon-project.eu) and EASIER (www.project-easier.eu/)

RQ: How does a neural model trained on videos compare to a model trained on visual representations?

In order to answer this question we build two different models from scratch - CNN-based encoder-decoder and also standard MT encoder-decoder the inputs of which are the visual embeddings obtained with MediaPipe Holistic from the input videos. We can now formulate the following subquestions:

SQ1: Which approach for extracting the spatial features (CNN-based or MediaPipe-based) works better for sign language translation task?

SQ2: To what extent a simple encoder-decoder architecture based on text2text MT RNN model can translate signed utterances (videos) into text?

Chapter 2

Related Work

SLMT lies in the intersection of three separate disciplines: Computer Vision (CV), Machine Translation (MT) and Linguistics.[8] CV is a field of AI, the purpose of which is to make computers gain high level understanding from images and videos. Its role in SLMT is to extract meaningful spatial features which can then be used for performing the task of translation. In MT we develop systems which translate between different spoken languages. Since the main task in SLMT is translation, the two fields are deeply connected. Linguistics is the scientific study of all aspects of languages but most importantly their nature and structure. Linguistics knowledge is crucial for all translation tasks. Before exploring the different tasks in SLMT, it is first important to understand how text to text MT works, so that we can then draw inspiration from there.

2.1 Machine Translation Approaches

Before Deep Learning became popular MT was done by classical rule-based systems (RBMT). It was based only on linguistic information about the two languages (source and target languages). RBMT performed the translation tasks relying solely on Grammars and Dictionaries built for the respective languages. However, the quality of the translation was not even close to matching human level.

With the emergence of new powerful machines and processing units (GPUs), Neural Networks that were introduced in 1943 by McCulloch and Pitts [9], and studied theoretically from the 1960s onwards, now proved to perform much better in translation tasks than the old RBMT systems. There are two main architectures used in Neural Machine Translation (NMT): RNN-based and and Transformer based.

2.1.1 RNN-based

Recurrent Neural Networks (RNN) were first introduced in 1986 [10]. The architecture was specifically designed to model sequential data. In translation, since we are working with sentences (sequences of words), RNNs are particularly useful. In 1997 Sepp Hochreiter and Jürgen Schmidhuber proposed an improved version of the basic RNN architecture called Long-Short Term Memory (LSTM) [11]. The main problem of the standard RNN was that it could not model well relationships between distant parts of the input sequence. LSTM was designed with the purpose of solving this issue. By the use of forget, input and output gates it manages to preserve the important in-

formation regarding the sequence even for distant dependencies in it. In 2014, in order to address the same issue, Junyoung Chung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio proposed another architecture - Gated Recurrent Unit [12]. Here instead of three gates they use just two - reset and update gate.

The first successful RNN-based method for MT was the Sequence to Sequence model, proposed by Google's researchers Ilya Sutskever, Oriol Vinyals and Quoc V. Le in 2016 [2]. The main idea of the method is to have an RNN model (Encoder) which learns the input language and then from its hidden representations, a second RNN model (Decoder) that translates into the desired output language. Their particular model used LSTM layers.

2.1.2 Transformer-based

Transformers were first introduced in 2017 in [3]. They rely on the power of the Attention mechanism [1] that takes into account the context of the words in order to improve the quality of the translation. While Attention was originally designed to be used in an RNN setting, in this new approach it is part of an architecture similar to the original Artificial Neural Networks (or Multi Layer Perceptrons - MLPs). The key idea is to generate three different representations from each word in the input sequence (key, value, query) which are then used to figure out the connections in the input and the proper mapping to the output sentence. The current State Of The Art (SOTA) in MT are Transformer models.

Now that we have covered the main ideas in spoken to spoken language (text to text) MT, we can focus on SLMT, which shares a lot of similarities with it but also has some distinct differences that are very important to take into account.

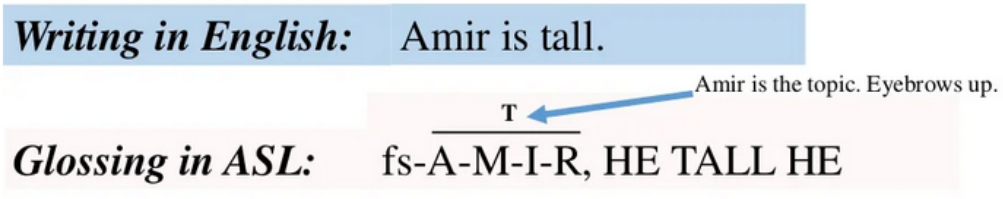
2.2 Tasks in SLMT

There are four distinct tasks in SLMT, namely: Gloss to Text, Sign to Gloss, Sign to Gloss to Text and Sign to Text. [8]

2.2.1 Gloss to Text

Glosses are high level representations of signs that capture meaning, while reducing the signed utterance to a sequence of tokens. Figure **??** illustrates an example Gloss in American Sign Language (ASL).

Figure 2.1: Gloss example



In Gloss to Text, both the inputs and the targets are sequences of tokens, which is why this is seen as the easiest task in SLMT and is modeled like a standard text to text MT system, because of their huge success in spoken to spoken language translation. The purpose of the Gloss to Text systems is to serve as guidelines for other SLMT tasks, specifically for how to design the representations inside the models in order to help the learning procedure. It is important to note, however, that glosses do not encapsulate the full linguistics meaning of the signs, so these systems cannot be seen as an upper bound on the performance of other SLMT systems [8].

2.2.2 Sign to Gloss

If we have perfect Gloss to Text systems, the only step towards achieving full SLMT will be to have a model that can map from Sign to Gloss. One might imagine that Sign to Gloss systems need to just recognize individual signs in the videos and try to tackle the problem as a standard CV classification task. However, SLs are very complex and usually multiple things can be expressed at any given time (one frame from the input video): the strong hand of the signer can express one thing while the face can sign some emotion for example. This makes the task very challenging.

2.2.3 Sign to Gloss to Text

Sign to Gloss to Text systems can be seen as the composition of the the two tasks we listed before: Sign to Gloss + Gloss to Text. This approach is seen as a good candidate to solve the SLMT problem as a whole - translate from a sentence in sign language to one in spoken language. The assumption is that glosses are very helpful middle representations, which if our models can learn will decrease the difficulty of the problem.

2.2.4 Sign to Text

Sign to Text systems do not rely on the glosses in order to perform the translation. In most cases they also generate the text from some intermediate state, but here we do not force the system explicitly to learn the glosses as the useful intermediate representation. The assumption is that there might be other important information in the videos that can be learned besides the glosses which could improve the translation performed in the second module of the system.

2.3 Visual Embeddings

Computer Vision has a crucial role in SLMT because if we want to perform the translation properly we need to extract all meaningful information about the sign language from the input videos. We usually refer to this information as Visual Embeddings or Visual Representations. There are two possibilities here: to use an already built feature extractor, or to develop a completely new model that learns the features in the same time as learning the translation - learning the embeddings similarly as in standard text to text MT systems.

Currently, the most common approach is using already established feature extractors. Information in SLs is conveyed through the pose of the speaker. As of now, the established way of representing this pose is by storing the key body points of the hands, face and shoulders. These key body points represent features which can be extracted from the input videos using key point

extractors such as MediaPipe[1], OpenPose[2], AlphaPose[3] and DensePose[4]. These extractors drastically reduce the dimensionality of the input. Usually, the key points from a given frame can be stored in 1000 to 2000 dimensional vectors. In contrast, if we input the whole frame, even if it is in small resolution (224x224px), we already have more than 40000 dimensions.

The alternative of using pre-trained feature extractors is building new models from scratch that learn proper visual embeddings for the given task. Usually they are either Convolutional Neural Networks (first introduced by Yann LeCun et al. [13].) or Vision Transformers (Dosovitskiy et al. [14]). Most of the time these models are incorporated in the Encoder part of SLMT model and they learn the spatial embeddings during the training of the whole system. It is important to note that with this approach we do not restrict the model with assumptions on which spatial features are important for the task of SLMT, but rather let it learn the spatial embeddings that work best (minimize the loss; achieve highest BLUE score).

Our work lies in the fourth task of SLMT - Sign to Text (video to text). The two approaches that we investigate and compare are based around the two different methods for embedding the spatial information from the videos - our first approach learns the spatial embeddings along with the training of the whole SLMT system; while the second (using pre-trained feature extractors) relies on key points extracted from the videos with MediPipe Holistic .

[1]https://mediapipe.dev/
[2]https://github.com/CMU-Perceptual-Computing-Lab/openpose
[3]https://github.com/MVIG-SJTU/AlphaPose
[4]http://densepose.org/

Chapter 3

Methodology

3.1 Data Explanation

The data that we use in this project is the NGT corpus [1]. It contains videos of people communicating in NGT. In some of the videos, the deaf person is asked questions and his or her answers are recorded. In others there are whole conversations in NGT between deaf people that communicate with one another. For all those videos we have the respective translations in Dutch. This data set contains recordings from people of different gender, age groups and regions of the Netherlands. This makes the corpus very diverse.
The SignON project initially pre-processed the data so that the long videos of conversations are split into different sentences. So now we have a parallel corpus consisting of 22141 MPG videos of different length and their corresponding Dutch translations in a TXT file.
For the purposes of this project we need the data to be in the proper format and types so that we can input it to the models that we build afterwards. The videos were converted into arrays of numbers in a straightforward manner using the python library - Open CV [2]. We perform tokenization (process which gives distinct IDs to all unique tokens present in a document) on the set of all target sentences. Then using the IDs we convert the Dutch sentences into arrays.

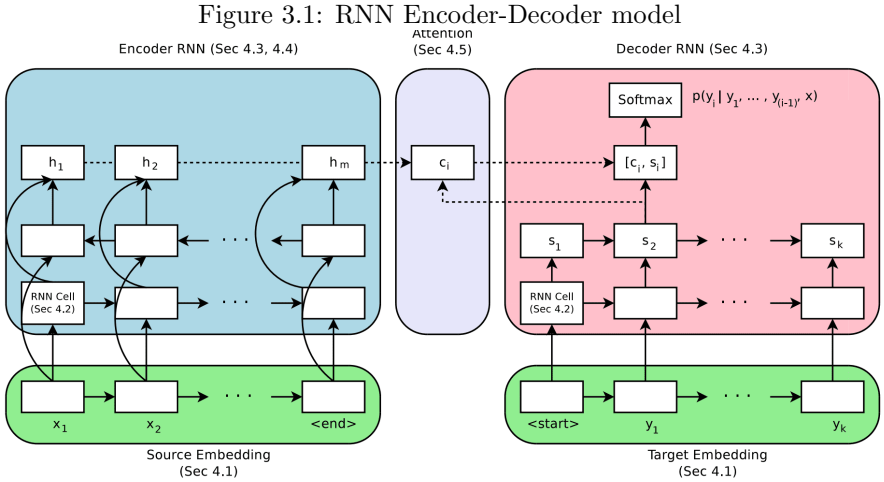
3.2 Encoder-Decoder Architecture

In this project we develop and compare two different approaches for neural end-to-end sign to text translation. Both methods are Encoder-Decoder based.
The basic idea of Encoder-Decoder neural networks is that there are two separate models, where the output of the first model (Encoder) is used as an input to the second model (Decoder). This is a very useful approach in the setting of Machine Translation, because despite the difference in expression between languages, the meaning behind remains the same (point to the same object, or express the same feeling). The encoder model receives the input sentence in the first language and captures the meaning of it in some representation - context vector. Afterwards this 'language free' context vector is fed as an input to the Decoder model, which tries to generate the proper translation in the target language.

In our case, however, we translate from a Sign Language to a Spoken Language, so we have two

[1] https://www.corpusngt.nl/
[2] https://opencv.org/

Figure 3.1: RNN Encoder-Decoder model



different modalities. Here the logic of the Decoder is still the same - from a given context vector as input, generate the proper target translation in Dutch. So, we mainly focus our attention to the Encoder network and try to figure out what is the best way to extract a good context vector from our videos, than can be afterwards translated by the RNN Decoder.

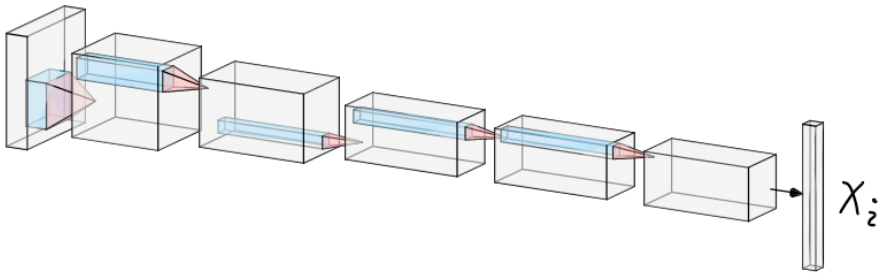
3.3 Encoder

In this section we present the two approaches that we want to compare.

3.3.1 CNN approach

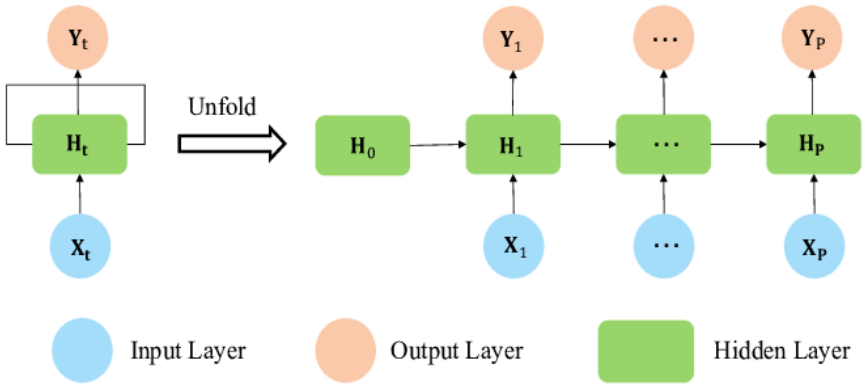
Our first approach is based on Convolutional Neural Networks (CNNs). CNNs are specifically designed for image data and can learn very well the spatial dependencies of the given input. Because videos are just a sequence of images (frames) we chose to use CNNs as our first approach.
Each frame of a given input video is passed through a CNN module, of which the final layer is Linear, so that the output is a single vector x_i of fixed size which is equal to the number of nodes in the final layer. After processing all P frames of our video we have P vectors that capture the spatial information of the input (see figure 3.5).

Figure 3.2: CNN Module



After encoding the spatial information, we also need to capture the temporal information of our feature vectors x_i. This is a sequence of vectors so we can model them the same way we model time series, or text - via Recurrent Neural Networks (RNNs).

Figure 3.3: RNN Module



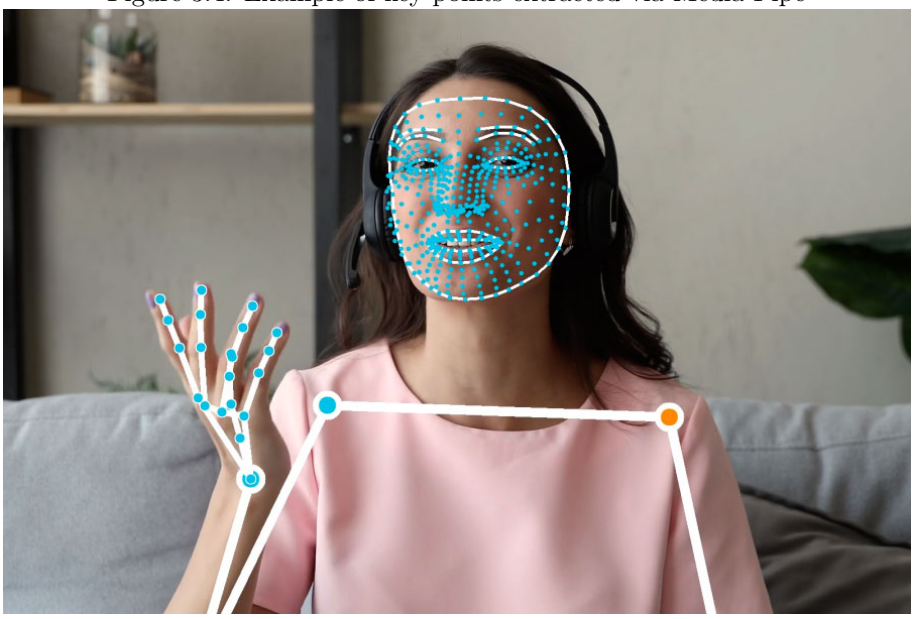
After the whole sequence of spatial features has passed through the RNN, its last hidden state H_p contains the spatiotemporal information of the whole input video and this is the context vector from which we want our Decoder to generate the correct translation.

3.3.2 Media Pipe approach

We assume that the input video contains a lot of information that is not really useful for the task of sign language translation (for example background or clothing). We know that the actual information is expressed through several body parts - hands, face and shoulders. Thus we were looking for a method that takes into account only those parts of the input video. Media Pipe is a well established tool that extracts key points from human body in a video. So for our second approach

we decided to use Media Pipe as a way of extracting the spatial features (body key points). After a video is passed through Media Pipe, we get a vector of key points for each frame. This is exactly the same format of output as in the CNN approach. So we can again feed the sequence of key point vectors k_i as an input to the RNN module of the Encoder. Once again after the RNN processes the whole sequence we can take its last hidden state H_p and feed it to the Decoder so it can do the translation into Dutch (see figure 3.6).

Figure 3.4: Example of key points extracted via Media Pipe



3.4 Decoder

The Decoder is an RNN module that takes the last hidden state H_p of the decoder as its own initial hidden state and as first input the Start of Sentence (SOS) token from our vocabulary. Given the hidden state and the current input, it predicts the next token in our output sequence. This token is in turn provided as the next input to the RNN and so on. In the end we take all of the output tokens which represent the predicted sentence. With this predicted sentence we can calculate the loss function between the prediction and the target and we can train our model using Backpropagation [10].

3.5 Evaluation

After performing all our experiments, we evaluate our models with their loss (standard evaluation metric for all supervised models) and also BLEU score,[3] which is the most popular metric for measuring the quality of an MT system.

[3]https://en.wikipedia.org/wiki/BLEU

Figure 3.5: End-to-End Approach

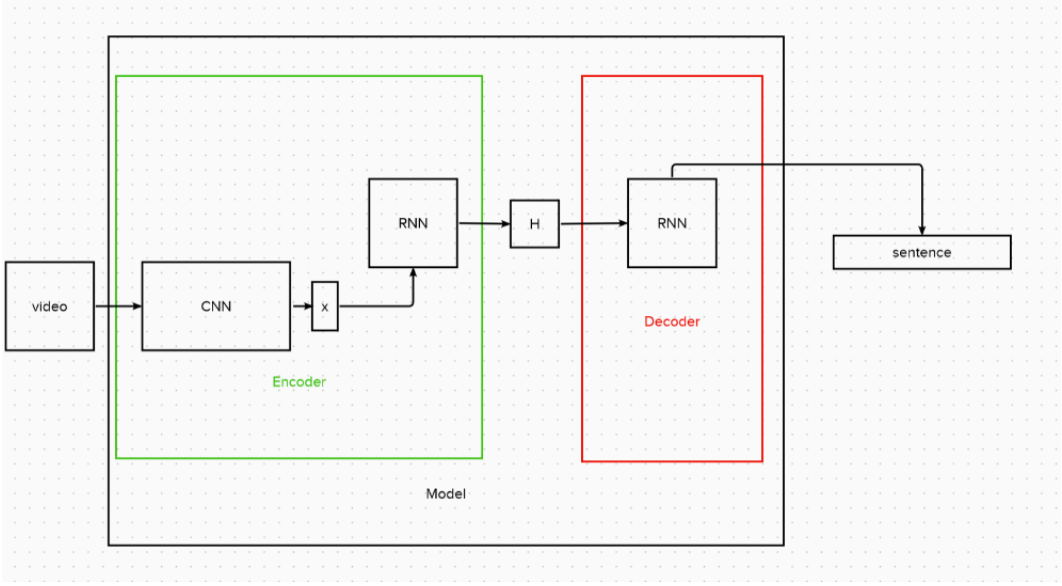
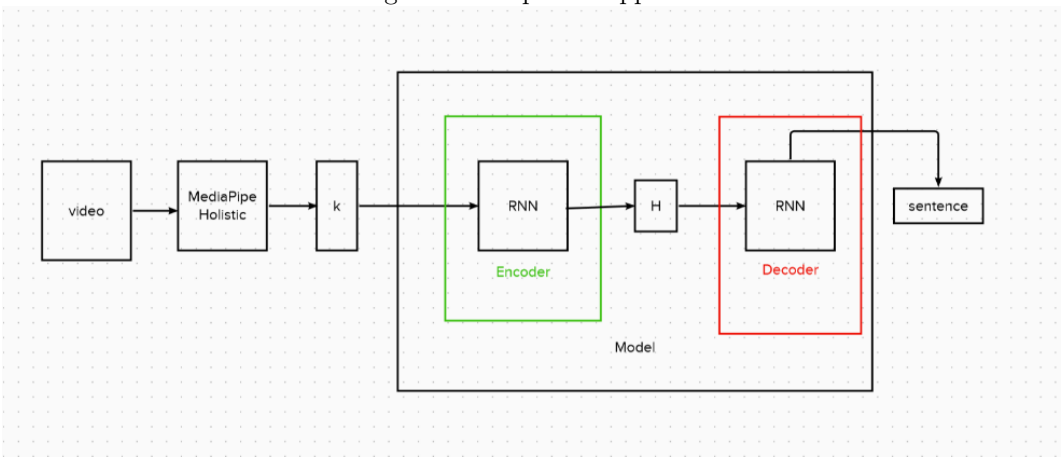


Figure 3.6: Pipeline Approach



Chapter 4

Experiments and Results

4.1 Setup

During our investigation we performed multiple experiments for both approaches in order to test our hypotheses. We trained each model on different sizes of our training data and drew conclusions upon the results we achieved.

4.1.1 Model 1

- CNN + RNN Encoder
  - standard VGG-11 model with last linear layer of size 256
  - 1 layer GRU with size 256
- RNN Decoder
  - 1 layer GRU with size 256

4.1.2 Model 2

- Key pints from MediaPipe Holistic for each frame
- RNN Encoder
  - 1 layer GRU with size 256
- RNN Decoder
  - 1 layer GRU with size 256

4.1.3 Shared for both models

- Early Stopping mechanism
- Teacher forcing ratio = 0.5
- Adam optimizer
- Learning Rate = 0.001

4.1.4 Technical specifications

- GPU: A40
- 48 vRAM
- Pytorch implementation from scratch

4.2 Experiment 1

Our first experiment compares the two approaches trained on 1000 videos. As we see in figure 4.1 both the train and validation loss behave similarly for the two models. Moreover, they achieve similar BLEU scores on the train and test set as seen in table 4.1. This could possibly indicate that the CNN-based model learns either similar features to the key points used in our second model, or other features that are of similar importance for the task of SL translation.
Both models perform poorly: they do not generalize well - the validation loss is increasing as seen in figure 4.1; and do not translate well - both have low BLEU scores (table 4.1). This could be due to the very small training set. Only 1000 sentences cannot capture the complexity and richness of a language, so we cannot definitively determine whether the bad performance is due to the simplicity of our models or the size of the training data.

Figure 4.1: Loss plot Experiment 1

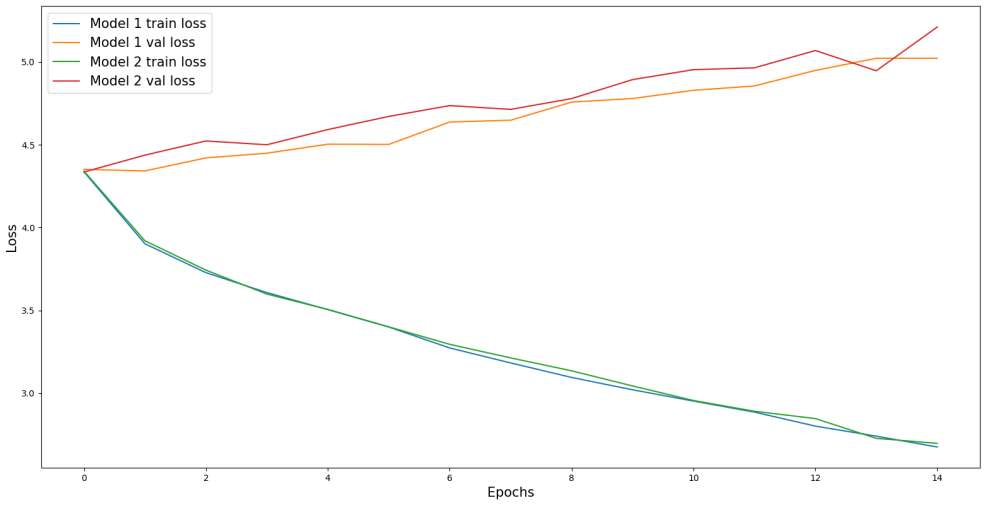


Table 4.1: BLEU scores Experiment 1

	Train BLEU	Test BLEU
Model 1	1.33	1.11
Model 2	1.36	1.11

4.3 Experiment 2

For our second experiment we tried training both models with 5000 videos. Our CNN-based model (Model 1) could not processes it because it ran out of memory. So we obtained results only for Model 2. As seen in figure 4.2 we still have the same problem as in our first experiment - the model does not generalize well on the test data. Also again the BLEU scores that we get (figure 4.2) for both the train and test sets are low which means poor translation quality, although we see a slight improvement on the test set from the models that we trained during Experiment 1.
Due to the fact that Model 1 did not manage to produce results we no longer can investigate the similarities between the two approaches.

Figure 4.2: Loss plot Experiment 2

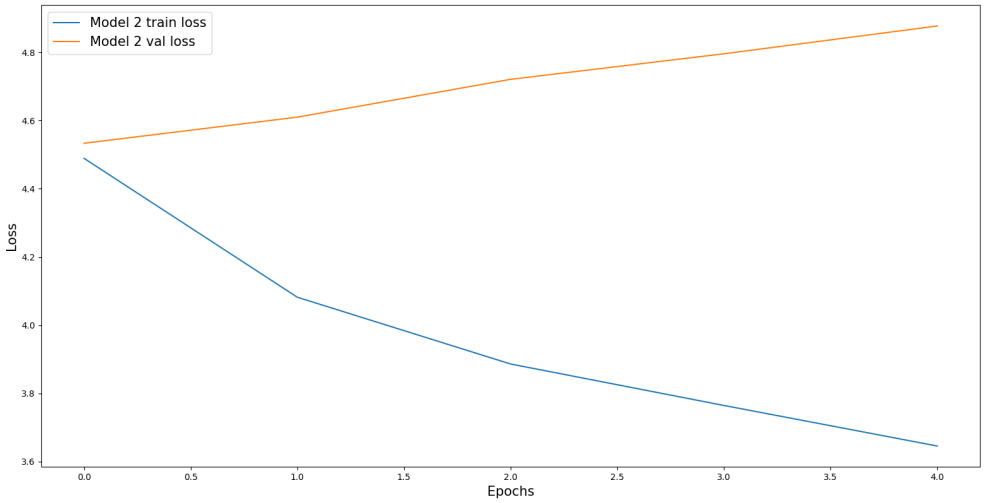


Table 4.2: BLEU scores Experiment 2

	Train BLEU	Test BLEU
Model 1	-	-
Model 2	1.35	1.29

4.4 Experiment 3

The last experiment we performed was with 10000 training videos. We again obtained results only for Model 2 for the same reason as in Experiment 2 - with Model 1 we run out of memory for such a big data set. Similarly to the first two experiments our model could not generalize to the validation set (figure 4.3) and also the quality of the translations achieved by it was again low (table 4.3). Despite these facts, it is important to note that the BLEU scores (both for train and test set) improved over the previous experiments which further confirms the already established principle in Deep Learning that increasing the size of the training data improves the performance of the models.

Figure 4.3: Loss plot Experiment 3

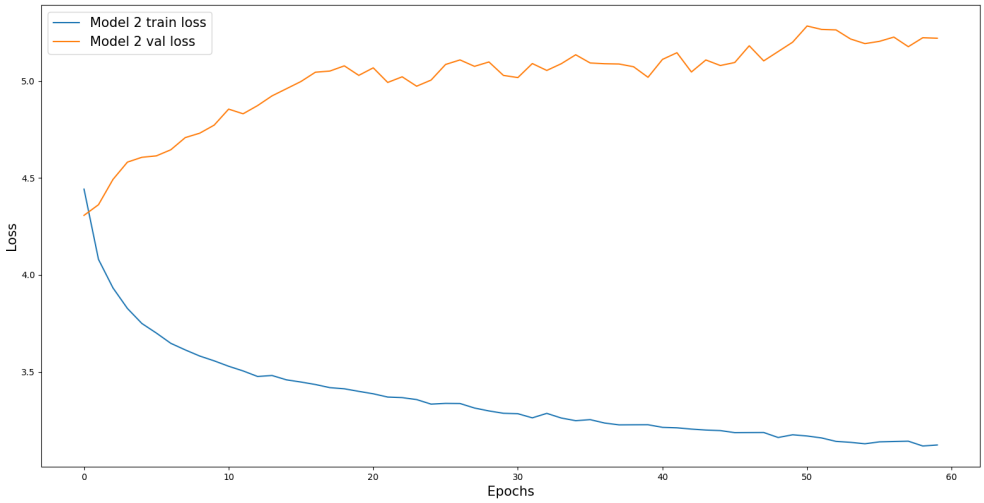


Table 4.3: BLEU scores Experiment 3

	Train BLEU	Test BLEU
Model 1	-	-
Model 2	1.60	1.93

Chapter 5

Discussion

So far we have established that increasing the size of the training set improves the quality of translation but still our best model does not perform well enough on the training data and cannot generalize to the test set. The comparison between the two proposed approaches in terms of translation quality cannot be properly done since in only one of our experiments we managed to obtain results from both models. This experiment suggested that they are on par with each other both in terms of translation quality (on both training and test sets) and generalization power. However, this being the setting with the least amount of data, nothing conclusive can be said about which of them is better - achieves higher BLEU score consistently. Despite that, we found out that increasing the size of the training data is something that very much affects the End-to-End approaches (CNN-based), since they take directly the raw videos as input, which requires a tremendous amount of memory (RAM) to store all this data before feeding it to the model. In contrast to that, pipeline approaches do not suffer from this memory problem. This is due to the fact that they rely on feature extractors (pretrained models such as MediaPipe Holistic in our case) which vastly reduce the dimension of the inputs and then the needed memory storage is almost negligible compared to the one required for the End-to-End approaches. This makes the pipeline approaches easily scalable.

For further investigation on the comparison between End-to-End and Pipeline approaches for SL translation, the first problem that should be addressed is the memory issue of storing the data before feeding it to the model. With our approach we first process and store all the videos in memory, then split them in training, validation and test sets respectively and after that we train our model. Since videos take a lot of memory to store, our machine runs out of RAM with quite a small number of them (5000 are already too much). One suggestion we can provide is to process one video at a time and train the model on it before proceeding with the next video. By doing this only one video will be stored in memory at any given moment, so no matter how many videos we want to use in the training set we will not face the same issue.

The poor performance of our models can be due to multiple factors. The most obvious one was the amount of data with which we trained them - 10000 sentences (in Experiment 3) are generally very little for training a good MT system. Usually MT models that achieve good translation quality are trained on corpora of millions of sentences as shown in [15], which can explain our low BLEU scores (both for training and test sets). The poor generalization of our models could be explained by heterogeneity in the data (test set contained completely different signs and words from the training set), but this is very likely not the case. Therefore, it is logical to state that our model architecture was just too simple for the task of SL translation.

One of our model's problems is that it does not have an Attention mechanism. This was initially planned as a next experiment in our investigation but due to time limitations[1] we could not incorporate it. Attention is the key part of the success of recent MT systems [1], so this is the first thing that should be integrated in this model in future work.

Another problem that we could not solve in this project was batching of the inputs. Because our inputs are videos of varying frame sizes, there is no established method for grouping them in batches that the models can process in a single iteration. This forced us to perform the optimization stochastically (one video at a time), which could also be an explanation of the lack of generalization power of our models. This problem is worth considering in future work. One idea that came to mind, was to get fixed number of samples from the train set, pad the shorter length videos with empty frames until they reach the length of the longest in the group and then use it as a batch. We could not use that in our experiments, because this padding would increase even further the required memory to store the inputs, so again this approach cannot be scaled.

If batching is to be used with our first suggestion, instead of processing one video at a time, a given batch size can be specified with how many videos are to be processed at a time. This will not be an issue since usually batches are of 32, 64, 128 or 256 training examples. In our first experiment we managed to store 1000 videos in memory so everything below that will be possible. Here the padding idea would work, because the batches will be composed of a relatively small number of videos, so increasing their lengths will not lead to memory issues.

As an alternative to batching it could also be possible to employ Gradient accumulation. This is a technique where instead of updating the weights of our model after each batch, we save the gradient values and accumulate them throughout few iterations. After a specified number of iterations N, we update the values of the weights with this accumulated gradient divided by N. When this method is employed in the case of only one input in a batch, it can be seen as performing the batching in the model itself. Same as batching, its main purpose is to reduce the fluctuations which are likely to happen when training Neural Networks stochastically.

Another possible way of improving the results is to increase the number of RNN layers in both the Encoder and the Decoder. This will increase the complexity and possibly capture more structure and meaningful information from the input, thus leading to better generalization capability of the models.

In future work, if the scaling problem of the CNN-based model is resolved, an interesting hypothesis that could be worth investigating, is if the two approaches perform similarly (as in our first experiment) on large amounts of training data; what are the spatial embeddings that the CNN-based model learns? And more specifically, if the size of the final linear feature vector at the end of the CNN part of the Encoder is the same length as the key points vectors extracted from MediaPipe Holistic (that we use in the second approach), then a comparison between the two could be made (measure distance for example). If the two different systems are very similar in terms of their performance, and also the CNN part learns spatial embeddings similar to the key points, then our hypothesis is that indeed the key points the best possible low dimensional representation of the videos which work well in SLMT.

[1]There were some technical problems that we faced during the developing and training stages of our first experiments that delayed the whole process.

Chapter 6

Conclusion

From our experiments and analysis we cannot state with confidence whether the Pipeline approach outperforms the End-to-End approach in the task of SL translation from NGT to Dutch in terms of quality (BLEU score). However, we found significant issues with the CNN-based model in terms of memory required to store all the input videos before being able to train the model with them. We proposed a possible solution that works around this problem and it is worth investigating it in the future. As it stands, our experiments have shown that only the Pipeline approach can be trained on big data sets, but still the translation quality is not sufficient to say that we have built a model that performs translation from NGT to Dutch well.

Bibliography

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014. [Online]. Available: https://arxiv.org/abs/1409.0473

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014. [Online]. Available: https://arxiv.org/abs/1409.3215

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[4] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T.-Y. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, and M. Zhou, "Achieving human parity on automatic chinese to english news translation," 2018. [Online]. Available: https://arxiv.org/abs/1803.05567

[5] S. Läubli, R. Sennrich, and M. Volk, "Has machine translation achieved human parity? a case for document-level evaluation," 2018. [Online]. Available: https://arxiv.org/abs/1808.07048

[6] J. Quer and M. Steinbach, "Handling sign language data: The impact of modality," *Frontiers in Psychology*, vol. 10, 03 2019.

[7] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudrealt, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, C. Vogler, and M. R. Morris, "Sign language recognition, generation, and translation: An interdisciplinary perspective," 2019. [Online]. Available: https://arxiv.org/abs/1908.08597

[8] M. D. Coster, D. Shterionov, M. V. Herreweghe, and J. Dambre, "Machine translation from signed to spoken languages: State of the art and challenges," *CoRR*, vol. abs/2202.03086, 2022. [Online]. Available: https://arxiv.org/abs/2202.03086

[9] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014. [Online]. Available: https://arxiv.org/abs/1412.3555

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[15] A. Poncelas, D. Shterionov, A. Way, G. Wenniger, and P. Passban, "Investigating backtranslation in neural machine translation," 04 2018.