D2.1 First Version of GSL Recognizer

SL-ReDu
Sign Language Recognition in Education

Partner Responsible	UTH-ECE
Other Contributors	AthenaRC
Document Reference	**D2.1**
Dissemination Level	Public
Version	1.0 (Final)
Due Date	January 2021 (**M12**)
Date of Preparation	January 2021

Contract No.: HFRI-FM17-2456

ΕΛ.Ι.Δ.Ε.Κ. Ελληνικό Ίδρυμα Έρευνας & Καινοτομίας

ATHENA Research & Innovation Information Technologies

Editor

Gerasimos Potamianos (**UTH-ECE**)

Contributors

UTH-ECE: Katerina Papadimitriou, Gerasimos Potamianos

AthenaRC: Eleni Efthimiou, Stavroula-Evita Fotinea, Petros Maragos

SL-ReDu Principal Investigator:

Assoc. Prof. Gerasimos Potamianos

University of Thessaly, Electrical and Computer Engineering Department (**UTH-ECE**)

Volos, Greece 38221

email: gpotamianos@uth.gr (gpotam@ieee.org)

Executive Summary

The SL-ReDu project aims to advance the state-of-the-art in the automatic recognition of Greek Sign Language (GSL) from videos, while focusing on the education use-case of standardized teaching of GSL as a second language. In this deliverable (D2.1), we present our first version of the sign language (SL) recognizer, focusing on two recognition problems: (a) that of isolated signs of GSL, and (b) that of continuous sequences of fingerspelled characters. Specifically, we build upon the visual tracking and feature extraction methods that we developed earlier as part of D1.1 (M06), exploiting the OpenPose human skeleton tracking algorithm to detect both manual and non-manual SL articulators. Based on these, we extract low-level SL information, namely handshapes, body postures, and mouth gestures/shapes. We subsequently fuse such representations to yield higher-level SL information, thus being able to recognize complex signs. For this purpose, we investigate five deep-learning based algorithms that we adapt to the problem of SL recognition. To evaluate the developed algorithms and decide on the best approach, we conduct experiments on a total of four databases: Three of these are suitable for the problem of recognizing isolated signs of GSL, while the fourth is employed for continuous fingerspelling, albeit in the American SL due to the lack of a corresponding dataset in GSL. This deliverable constitutes part of the first project milestone (MS1 – M12), and it will be updated in the future as D2.2 and D2.3, gradually adding complexity to the GSL recognition task and algorithms.

Table of Contents

1 Introduction

Automatic sign language (SL) recognition from video constitutes a challenging problem that has attracted much interest in the literature [1-31]. Indeed, SL is a non-vocal form of communication, involving complex articulation in the 3D visible space around the speaker, with numerous upper-body "articulators" carrying specific information content [32]. Clearly, central to SL is the manual articulation, i.e. the shape, motion pattern, and relative position of one or both hands and arms w.r.t. the signer's body. However, non-manual articulators, in particular body leaning, shoulder motion, head pose, mouthing patterns, eye gaze, and eyebrow movement, all contribute to the formation of basic SL signs, complementing manual articulation [32-36]. It is thus clear that a successful SL recognition system should be able to accurately track both manual and non-manual articulators in space and time, recognize patterns in the respective articulatory streams, and fuse them at the appropriate temporal level to yield basic signs and their temporal sequence. Therefore, crucial components in addressing the problem are the visual detection, tracking, and visual feature representation of both manual and non-manual SL articulation, and subsequently the employment of suitable learning approaches for classification and fusion based on the extracted visual feature representations. Achieving such goals constitutes the focus of this WP2 deliverable, building on our earlier work on visual tracking and feature extraction that we have reported in D1.1 [37] as part of the WP1 activities.

Specifically, in D2.1 we follow the best performing approaches of D1.1, starting with the OpenPose framework [38-40] to detect the human skeleton in the 2D image plane and from it the regions of interest for the signer hands and mouth. Subsequently, we extract visual features from these, employing 2D convolutional neural networks (CNNs) [41-47], and we combine the resulting representations by early fusion. We summarize this process in Section 2.

The main part of the deliverable concerns the first version of classification algorithms developed for SL recognition as part of the WP2 project activities, based on the extracted features and trained on large SL databases. For this purpose, we focus on deep-learning techniques, since these are known to achieve state-of-the-art results on a number of visual spatio-temporal classification problems [41-52]. In more detail, we consider an attention-based encoder-decoder, motivated by recent work in automatic speech recognition [53-55] and machine translation [56-61], that is equipped with temporal convolutions, a multi-step attention mechanism, and gated linear units over the convolution output [31, 62]. Further, we investigate a number of alternative techniques, namely the Transformer architecture [63] based on a multi-head attention-based network, as well three attentional recurrent neural network (RNN) variations, based on the long short-term memory (LSTM) [64], the bidirectional LSTM (BLSTM) [65], and the gated recurrent unit (GRU) [56] architectures. We provide pertinent details in Section 3.

The aforementioned approaches are developed here for two recognition tasks: (a) that of isolated signs of the Greek Sign Language (GSL), and (b) that of fingerspelling, i.e. the recognition of continuous sequences of alphabet letter signs. The two problems constitute a medium-vocabulary isolated task and a small-vocabulary continuous task, respectively, thus providing desirable variability in our SL recognition efforts, which will progressively become more complex as our WP2 work advances and will be reported in future deliverables D2.2 and D2.3. Concerning the isolated GSL recognizer, this is developed and evaluated on data from three appropriate GSL datasets, namely the *Polytropon GSL corpus* [66] the *ITI GSL dataset* of [20], and the GSL part of the *Dicta-Sign database* [67]. These data resources were identified as part of the WP3 project activities and are further described in deliverable D3.1 [68]. Concerning the fingerspelling task, it should be noted that this holds great importance to SLs, due to the fact that fingerspelling signing is commonly employed for prominent words that lack unique signs, such as names, technical terms, and foreign words, all of which often hold a crucial content role [32], and as such has attracted significant interest in the SL recognition literature [21-31]. Note though, that due to the lack of a corresponding dataset in GSL, we develop and evaluate fingerspelling in the American Sign Language (ASL), employing the *ChicagoFSWild dataset* [30] for this purpose. We provide the corresponding algorithmic implementation details, experimental framework, and results in Sections 4 and 5 for the two recognition tasks, respectively, and we conclude this deliverable in Section 6.

2 Visual Tracking and Feature Extraction

As mentioned in the Introduction, concerning visual tracking and feature extraction, we employ the best-performing approaches among the ones investigated in D1.1 [37]. In particular, due to its robustness, we exploit the OpenPose framework [38-40] to detect the human skeletal joints of the signer and a number of facial keypoints, from which we extract regions-of-interest (ROIs) of the signer hands and mouth. Subsequently, we feed these ROIs to ResNet-18 CNNs [44] to produce appearance features of the handshapes and mouthing patterns, since such CNN-based representations have shown superior performance over a number of alternative features that we investigated in D1.1. Further, in addition to the approach of D1.1, we also consider shape/pose features in the form of the 2D coordinates of a number of human skeletal joints (including hand joints) that are returned by OpenPose, motivated by [47-49]. The resulting feature streams are then combined by simple feature fusion and fed to the classifier architectures that are detailed in Section 3. An overview of the overall approach is depicted in Figure 1. Further details are provided in the next subsections.

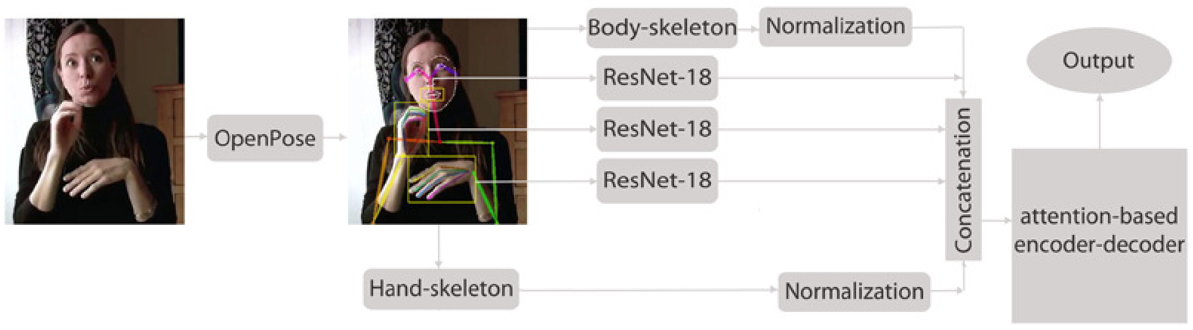

Figure 1: Overview of the visual tracking and feature extraction methodology adopted in D2.1.

2.1 Human Skeletal Features

To extract human skeletal data, the OpenPose human joint detector [38-40] is employed, which provides a descriptive structural representation of the human body (body pose, hands, and face) relying on deep convolutional pose models. To operate, the OpenPose network initially extracts image features employing the first 10 layers of VGG-19 [43], which are then fed to two parallel convolutional layer branches. The first branch generates a set of confidence maps, each representing a specific human pose skeleton part, while the second produces a set of part affinity fields [38] that represent the degree of confidence of the association for each pair of body part detections. OpenPose provides a detailed spatio-temporal representation of the human skeleton, extracting in total 137 human skeleton joint descriptors in the form of image coordinates (see Figure 2(a)). Specifically, OpenPose renders 25 body pose keypoints, 21 joints for each hand, as well as 70 facial keypoints, as also depicted in Figure 2(b).

For our problem, since the majority of SL videos typically include only the signer upper-body (as these are involved in the signing process), we exploit only 57 estimated coordinates, removing 10 human body joints associated with the lower body of the signer, as well as the face keypoints (that do not involve manual articulation). To obtain translation and scale invariance, all extracted human skeletal joints are subjected to normalization by converting the image to a local coordinate system with the neck keypoint being its origin, whereas further normalization is applied based on the distance between the left and right shoulder keypoints. This yields 114-dimensional (dim) feature vectors, capturing the coordinates of the upper-body skeleton (30-dim) and the two hands (84-dim in total).

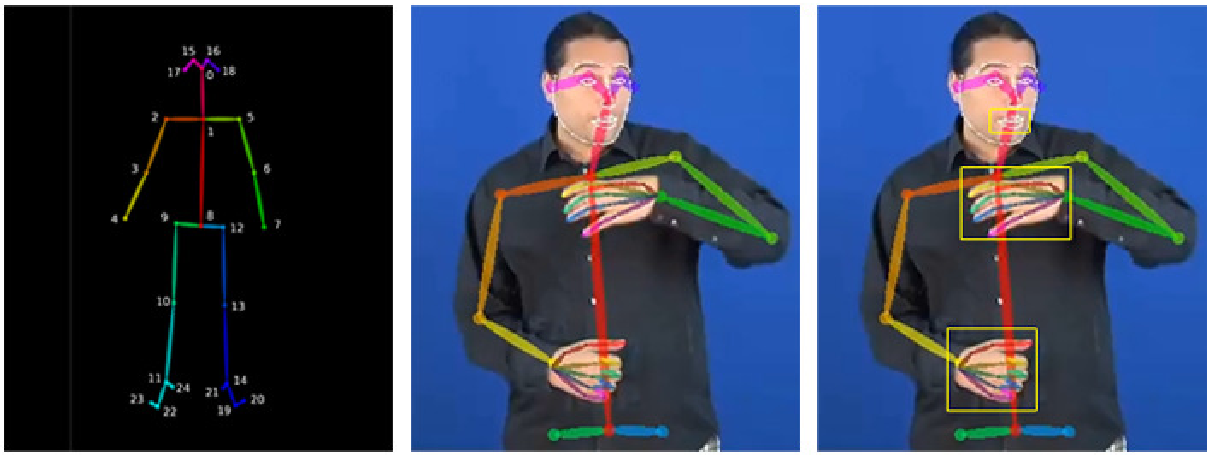

Figure 2: **(a)** An example of the skeleton representation obtained by the OpenPose library [40]; **(b)** input image frame from the Polytropon GSL corpus [66] with super-imposed keypoints generated by OpenPose; and **(c)** input image marked with rectangular boxes enclosing the handshapes and the mouth region derived based on the human skeleton.

2.2 Handshape and Mouthing Visual Features

The primary source of SL information is the manual articulation involving handshape deformation and orientation, with an additional source provided by mouth region gestures. For this purpose, we segment the two hands and mouth ROIs, based on the corresponding skeletal coordinates obtained by OpenPose (see also Figure 2(c)). This process yields three ROIs, each of which is subsequently fed to a pre-trained ResNet-18 convolutional network [44] (trained on the ImageNet database [69]) in order to extract feature maps by taking the output of the fully-connected layer of the CNN. The network uses 3 x 3 convolutional kernels, down-sampling with stride 2, and it is trained using the mean squared error loss function. Note that all ROIs are resized to the fixed size of the ResNet-18 network input layer (224 x 224 pixels). The network outputs 512-dim feature maps for each ROI, by considering the output of its global average pooling layer.

2.3 Feature Fusion

The resulting feature streams are fused via simple concatenation, generating a 1,650-dim feature vector (114-dim for the human skeleton, and 512-dim for the ROIs of each of the mouth and two hands), which is subsequently fed to the prediction module. Additional systems with fewer feature streams (hence lower dimensionalities) are also evaluated (see Sections 4.3 and 5.3). It should be noted that in case of missing streams due to OpenPose failures or occluded hands, the respective features are filled by zeros.

3 SL Recognition Architectures

Treating SL recognition from videos as a sequence-to-sequence prediction problem, we address it using sequence learning models based on encoder-decoder architectures equipped with attention [56-63]. In this deliverable, we investigate five such approaches, namely the attentional LSTM [64] encoder-decoder, the attentional BLSTM [65] encoder-decoder, the attentional GRU [56, 57] encoder-decoder, the Transformer encoder-decoder [63], as well as the attentional CNN encoder-decoder [31]. We provide more details in the next subsections.

3.1 Attentional LSTM Encoder-Decoder

Sequence-to-sequence prediction is mainly associated with attention-based RNN encoder-decoder models. A significant portion of attentional RNN such schemes has been proposed in the literature differing in the RNN types and in the context vector calculation. The most popular RNN encoder-decoder alternative is the LSTM network [28, 64].

In its general form, the RNN encoder-decoder module comprises two processes: encoding and decoding (see also Figure 3). In particular, the LSTM encoder [64] is fed with the latent representations x derived from the feature learner generating hidden state representations $h_m = LSTM\ (x_m\ ,\ h_{m-1})$. During the decoding process, hidden state sequence h is processed by the LSTM decoder producing the elements of the output sequence y, one by one. Further, the attentional models are based on the alignment between input and output denoted by the context vector c that expresses the likelihood of each chunk of source sequence being related to the current output. In attention-based architectures the context vector c is computed as the weighted sum of each encoder hidden state h at each time step t. The alignment score is given by a score function normalized by softmax. To date, several alternatives have been proposed for this task, like the alignment functions in [31, 58, 59].

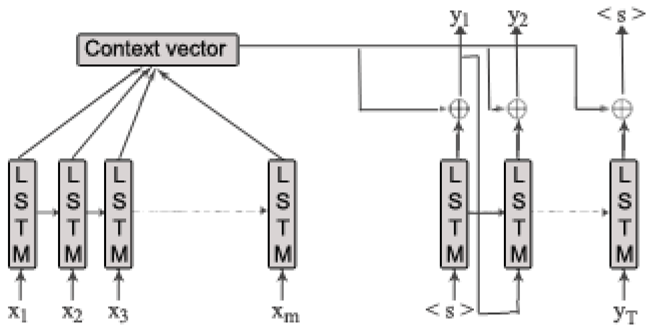

Figure 3: Attention-based RNN encoder-decoder architecture based on LSTMs [64].

3.2 Attentional BLSTM Encoder-Decoder

Recently, several RNN architectures based on bidirectional LSTM (BLSTM) [65] networks have been proposed [58, 60, 61]. Instead of predicting the current output based on the past context as in the case of LSTMs, in BLSTMs in addition to the previous observations there is also access to the future ones. This is obtained by applying two LSTMs: one processing the input sequence forward (left-to-right) and the other backward (right-to-left), both related to the same output (see also Figure 4).

As described in Section 3.1, during encoding the latent variable sequence x derived from the feature learner is fed to an RNN encoder, generating state representations h. The only difference with the attentional LSTM encoder-decoder is that, here, a BLSTM network is employed as encoder. During encoding the latent variable sequence *x* derived from the feature learner is fed to a forward LSTM encoder, generating state representations $\overrightarrow{h}_m = LSTM\ (x_m\ ,\ \overrightarrow{h}_{m-1})$. In addition, the input sequence is also processed in the opposite direction $\overleftarrow{h}_m = LSTM\ (x_m\ ,\ \overleftarrow{h}_{m+1})$. These two calculations are concatenated generating hidden state h_m. Then, as in the attention-based LSTM encoder-decoder of Section 3.1, hidden states *h* are processed by the LSTM decoder, producing the predicted output. As mentioned previously, in attentional models a set of alignment scores are first computed, which are multiplied by the encoded representations of the hidden states, producing the context vector *c*.

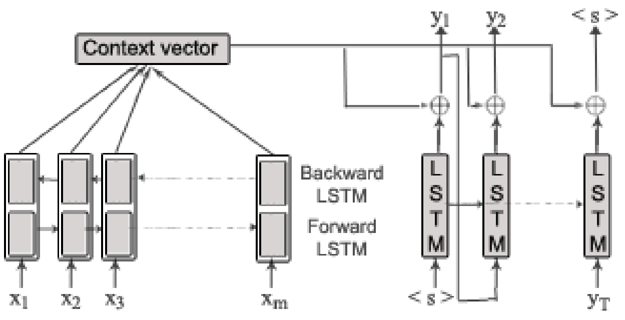

Figure 4: Attention-based RNN encoder-decoder architecture based on BLSTMs [65].

3.3 Attentional GRU Encoder-Decoder

As previously described (Sections 3.1 and 3.2), given the input sequence *x*, the LSTM is adopted as encoder to estimate the corresponding sequence of hidden state *h*. However, LSTM training is susceptible to problems such as the vanishing gradient and the exploding gradient as described in [70]. Therefore, gated recurrent units (GRUs) [56] have been introduced as an improved version of the recurrent block to adaptively capture various time-scale dependencies. Similarly to LSTMs, GRU has a gating mechanism adjusting the information flow within the GRU, however, with fewer parameters and operations.

Similar in spirit to the attentional LSTM / BLSTM encoder-decoder model, the attention-based GRU encoder-decoder comprises two processes, namely encoding and decoding (see also Figure 5). Specifically, the encoder maps an input sequence *x* to a sequence of hidden states *h*, while the decoder generates an output sequence *y* (one element at a time), given the hidden states *h*. In addition, since the model is equipped with an attention mechanism, the GRU decoder is fed with the previous target element y_{t-1} and the context vector *c*, generating the decoder hidden state, *d*, through which current predicted output y_t is produced.

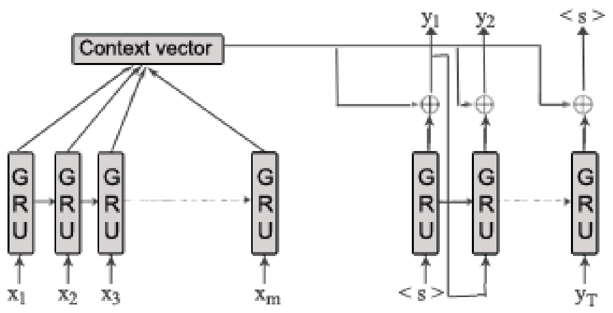

Figure 5: Attention-based RNN encoder-decoder architecture based on GRUs [56].

3.4 Transformer Encoder-Decoder

The transformer encoder-decoder [63] is an architecture based solely on attention mechanisms to outline the dependencies between input and output, abstaining from recurrence and convolutions. Specifically, transformers employ self-attention and point-wise fully connected layers in the encoder-decoder, as shown in Figure 6. The latent variable sequence output by the feature learner x is fed to the model generating the predicted output y. Specifically, the transformer encoder maps the input sequence x to a state representations h, and, subsequently, the decoder returns the predicted output, one by one, given h.

As in attention-based RNN encoder-decoders, the transformer encoder-decoder model is auto-regressive [71], utilizing the previously generated output element as additional input for generating the current output. The encoder consists of a stack of identical layers, each of which comprises two sub-layers, namely, a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Residual connections around the two sub-layers are added and followed by layer normalization. Further, the decoder follows the same architecture with the same number of stacked layers composing instead of three sub-layers, with the third one performing multi-head attention over the encoder output. Finally, residual connections around each of the sub-layers are added and also followed by a normalization layer.

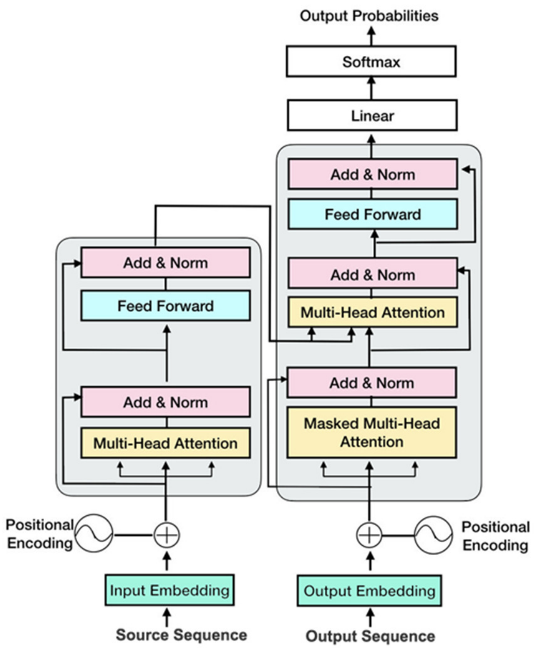

Figure 6: The transformer encoder-decoder architecture (figure from [63]).

3.5 Attentional CNN Encoder-Decoder

Another sequence learning architecture introduced in the literature is the attention-based CNN encoder-decoder [31], which relies on convolutional block structures on both encoder and decoder to compute the latent state representations (see also Figure 7). In its typical form, the l-th encoder layer reads in latent representation sequential data x and outputs a sequence of hidden states h_l, while the l-th decoder layer generates d_l hidden states and maps the latter to the desired output y. As in [31, 62], each layer composes of a 1-dim convolution followed by a gated linear unit [72], which behaves as a gating scheme that assists in dealing with the convolution output.

Considering a single-layer decoder with kernel width *k*, each output hidden state will be associated with *k* inputs. Thus, if we add multiple layers on top of each other, under the assumption that later layers will process *k* outputs of the previous ones, the resulting state will be related to more inputs than previously. To smoothly optimize and leverage the performance of deep convolutional networks, residual functions with reference to each convolution input and layer output are added.

Compared to attention-based recurrent encoder-decoder architectures, a multi-layer CNN encoder-decoder implies a multi-step attention mechanism [62]. The alignment scores *a* are computed through the alignment function in [31], conferred upon the corresponding decoder layer and each output of the last encoder layer. Subsequently, the context vectors *c*, which derive from the weighted sum of each output of the last encoder layer combined with the embeddings *e* of the input elements *x* in distributional space are in turn fed to the next decoder layer exploiting specific information during attentional vectors calculation.

In addition, the model is complemented with an input feeding scheme [59], where previous attentional vectors are taken into account during current alignment score calculation. Thus, the model constitutes a fully-connected deep network in both directions (vertical and horizontal) that deploys substantially previous alignment information during the estimation of new one.

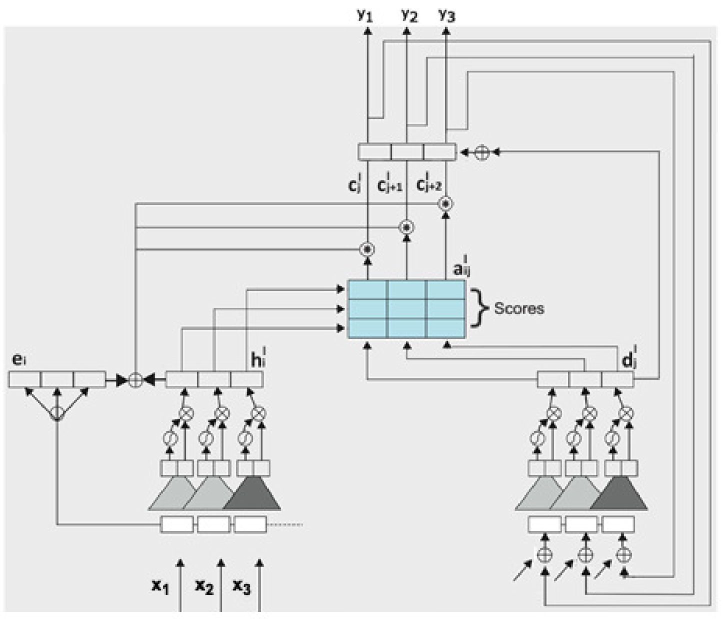

Figure 7: Attention-based CNN encoder-decoder architecture [31].

4 Isolated GSL Recognition

As mentioned in the Introduction, we consider two SL recognition tasks in D2.1. In this Section, we focus on the first task, namely that of recognizing isolated signs of GSL. Specifically, we first discuss the implementation details of the recognition models of Section 3, followed by the three datasets used in their training and evaluation, and finally we present our experimental results.

4.1 Model Implementation Details

The task of isolated SL recognition is viewed as a sequence learning problem that can be addressed by an attention-based encoder-decoder. In its typical form, the encoder reads in latent representation sequential data and outputs a sequence of hidden states, while the decoder maps the latter to the desired output (recognized sign). The attention mechanism performs alignment between the input and output, attending to the most relevant information in the source sequence. As already described in Section 3, we investigate five models within this general encoder-decoder framework, with the following details:

- An *attentional LSTM encoder-decoder* (ALSTM), where an LSTM [64] is employed as the RNN. Specifically, the model consists of a one-layer encoder and a one-layer decoder, both with 128 hidden units. Training is conducted using the Adam optimizer [73] with initial learning rate of 0.001 decayed by a factor of 0.3. The alignment scores are calculated using the function introduced in [58].
- An *attentional BLSTM encoder - LSTM decoder* (ABLSTM), where a BLSTM [65] is employed as the RNN encoder and an LSTM is used as the RNN decoder. The model comprises a one-layer BLSTM encoder and a one-layer LSTM decoder with 128 hidden units. Network training employs the Adam optimizer with an initial learning rate of 0.001 decreased by a factor of 0.3. The alignment is performed as in the ALSTM.
- An *attentional GRU encoder-decoder* (AGRU), where GRUs [56] are employed instead of LSTMs. In particular, the model constitutes a 2-layer GRU [31] encoder-decoder with 128 hidden units. During training the Adam optimizer is used with an initial learning rate of 0.001 decreased by a factor of 0.3. The attention score calculation is carried out as in the previous models.
- A *transformer encoder-decoder* (Transformer), which substitutes recurrent layers with multi-head attention ones [63]. Here, a 4-layer transformer is employed, with 8 heads for transformer self-attention, 2048-dimension hidden transformer feed-forward, and 512 hidden units. Training is carried out by Adam optimization with an initial learning rate of 0.003 decreased by a factor of 0.3. Parameter initialization is conducted through the Xavier process [74].
- An *attentional CNN encoder-decoder* (ACNN) [31], which enables parallelization, since CNNs do not depend on previous-time computations. The model includes 3-layer CNNs on both encoder and decoder with kernel width 5 and 128 hidden units. Training is conducted via the Adam optimizer with an initial learning rate of 0.001 decayed by a factor of 0.1. The attention score calculation uses the dot alignment function [59].

All models are trained employing a dropout rate of 0.3 with a mini-batch size fixed to 256. To achieve a better matching of a target element, the beam search strategy [75] with beam width of 5 during decoding is applied. In addition, label smoothing [63] rate of 0.5 is employed for reducing over-confidence in predictions. In all cases, GPU acceleration is used for both model training and evaluation.

4.2 Databases

The performance of the aforementioned algorithms is assessed on three publicly available isolated-sign GSL datasets that we have identified as part of WP3 activities, as reported in D3.1 [68]. These data resources are: (a) the *Polytropon GSL corpus* [66]; (b) the *ITI GSL dataset* [20]; and (c) the *Dicta-Sign database* [67]. The three corpora exhibit significant differences among them concerning the acted task and the recorded subjects, thus offering a desirable variation in the vocabulary content. Example frames from the three datasets are depicted in Figure 8, with more details provided next.

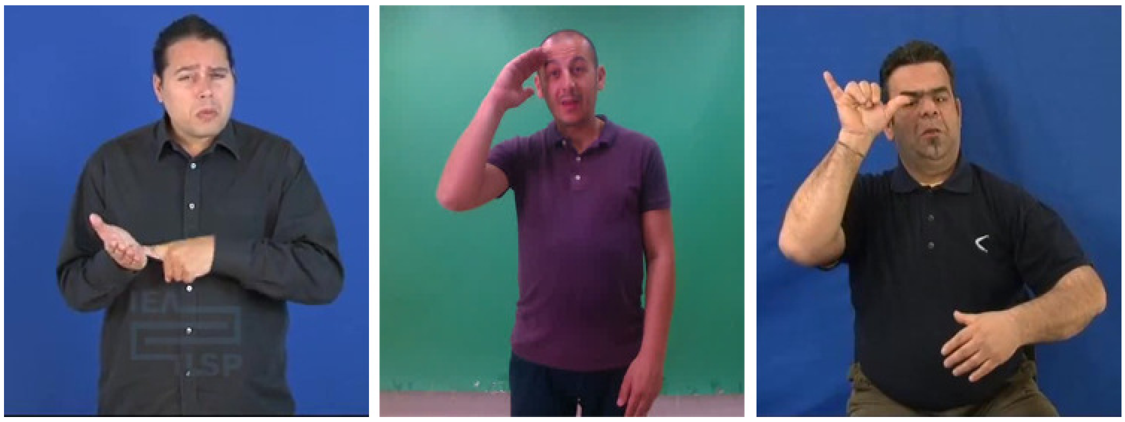

Figure 8: Example frames from the three GSL isolated datasets. Shown, left-to-right, the Polytropon GSL corpus, the ITI GSL dataset, and the Dicta-Sign database.

The *Polytropon GSL corpus* [66] contains 3 repetitions of 3,600 sentences performed by a single signer, recorded by two frontal-view cameras, a Kinect and an RGB one. Here, the RGB video data are used that are provided at a frame-rate of 25 Hz and 848×480-pixel resolution. Corpus annotations based on ELAN [76, 77] are available at both the signed sentence and signed word level. The corpus signed vocabulary includes 2,664 unique words corresponding to proper nouns, adverbs, and verbs characterized by variability in signing duration. In this study, we explore a vocabulary of 103 isolated signs that appear at least 30 times in the data (between 30 to 110 times, with 52.6 on average). These yield 5,414 video snippets, obtained by "cutting" the longer video database files based on the ELAN annotation time-stamps of the words of interest.

The *ITI GSL dataset* of [20] includes 5x3 different dialogues organized in sets of 5 individual tasks in 3 public services, performed by 7 different signers. The dialogues, which appertain to a communication between a deaf person and a single service employee, are pre-defined and are performed by each signer 5 consecutive times (5x7x5x3). Signing is captured by an Intel RealSense D435 RGB-D camera at a rate of 30 Hz, providing simultaneously RGB and depth streams at 648×480-pixel resolution. During recording, camera pose adjustments are made, offering a desirable variation. Corpus annotations by GSL linguistic experts are provided at both the signed sentence and signed word levels. The corpus signed vocabulary consists of 310 unique glosses (40,785 gloss instances) and 331 unique sentences (10,290 sentences), with 4.23 glosses per sentence on average. Here, an isolated sign recognition task is built for 305 unique words that appear between 4 and 10 times by each signer in the dataset, yielding 12,897 video snippets in total.

The *Dicta-Sign dataset* [67] is a multilingual corpus on the domain "Travel across Europe" in four sign languages (including GSL), concerning communication for transport by different means and contexts as well as related personal experiences. The corpus comprises 10 different tasks with a session duration of approximately 2 hours on the same elicitation material, covering various interaction formats from monologues to sequences of very short turns, also with different levels of predictability. The data are recorded by seven cameras, two of them stereo ones, capturing signing from different view-points (front, side, footage, and bird's eye view). The GSL part of Dicta-Sign contains data expressed by 8 pairs of different signers (16 signers in total) consisting of 8 to 10 hours of signing. Corpus annotations are based on iLex export format [78] as well as ELAN [76, 77], and they are provided at two different levels (signed sentence and signed word) containing labels and time-stamps. The corpus signed vocabulary consists of 1,704 unique words. Here, we employ an isolated small-vocabulary subset of 152 unique words with a sufficient number of occurrences among the 7 signers between 4 and 20 times. These yield 5,959 video snippets of words obtained by "cutting" the longer video database files based on the ELAN annotation time-stamps of the words of interest.

Since the GSL multi-signer datasets (i.e., excluding the single-signer Polytropon dataset) contain more than 4 recordings for each sign and every signer, experiments are performed in a multi-signer framework in an effort to retain a balance between the sets. In addition, all experiments on the three corpora are conducted using ten-fold cross-validation, where 80% of each fold is allocated to training, 10% to validation, and 10% to testing.

4.3 Experimental Results

For the task of isolated GSL recognition we evaluate the performance of the sequence learning models described in Section 3 on all three datasets, i.e. the Polytropon GSL (PGSL) corpus, the ITI GSL dataset, and the Dicta-Sign dataset. All our results are reported in word (gloss) accuracy (%).

In our first experiment, reported in Figure 9, we employ the feature representations of Section 2 individually, namely handshape features only (1024-dim), mouth region ones only (512-dim), and human skeletal features alone (114-dim). From the figure, it can be observed that the handshape feature stream yields the best results on all datasets. Further, the human skeleton seems to constitute a more powerful representation than the mouth region features, while using just the latter yields the lowest accuracy.

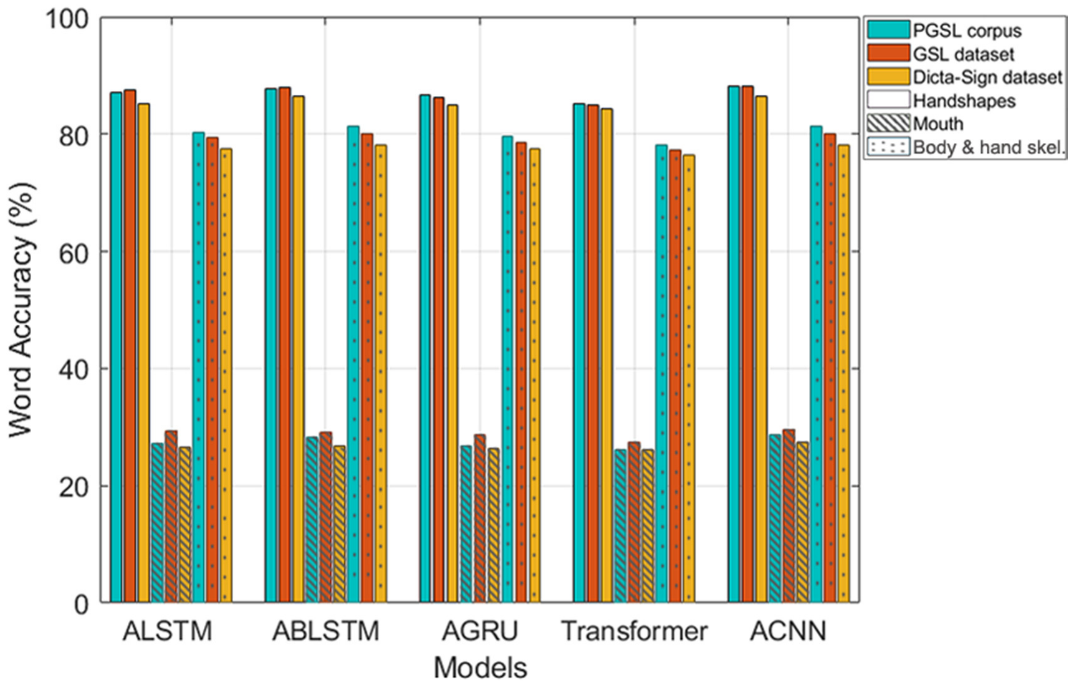

Figure 9: Word accuracy (%) of the evaluated encoder-decoder models on the three isolated GSL datasets employing individual feature streams (handshapes, mouth region, or human skeleton).

It can also be observed that the attention-based CNN encoder-decoder (ACNN) turns out superior to the considered alternatives in terms of word accuracy (PGSL: 88.17%, ITI GSL: 88.11%, Dicta-Sign: 86.55% (handshape feature stream)), revealing the power of exploiting convolutional block structures on the encoder-decoder, while the worst results for all GSL datasets are obtained by the Transformer encoder-decoder module. It should be noted that the attentional BLSTM (ABLSTM) encoder-decoder proves to be a powerful learning model, obtaining word accuracy results close to the ones achieved by the ACNN encoder-decoder. In particular, the ABLSTM model evaluated using the handshape feature stream yields 87.68% word accuracy on PGSL, 87.92% on ITI GSL, and 86.41% on Dicta-Sign.

In a second experiment, reported in Figure 10, we compare the performance of our models on all datasets investigating various combinations of the feature streams of Section 2 by feature fusion (concatenation). Our evaluation reveals that the combination of all feature streams, namely the handshapes, mouth, and human skeleton, yields the best word accuracy results compared to other feature combinations, while the combination of just handshapes and mouth region features performs the worst. It can also be observed that the attentional CNN encoder-decoder (ACNN) yields the best results on all three datasets, namely 92.25% word accuracy on PGSL, 92.77% on ITI GSL, and 90.23% on Dicta-Sign, when all feature streams are fused.

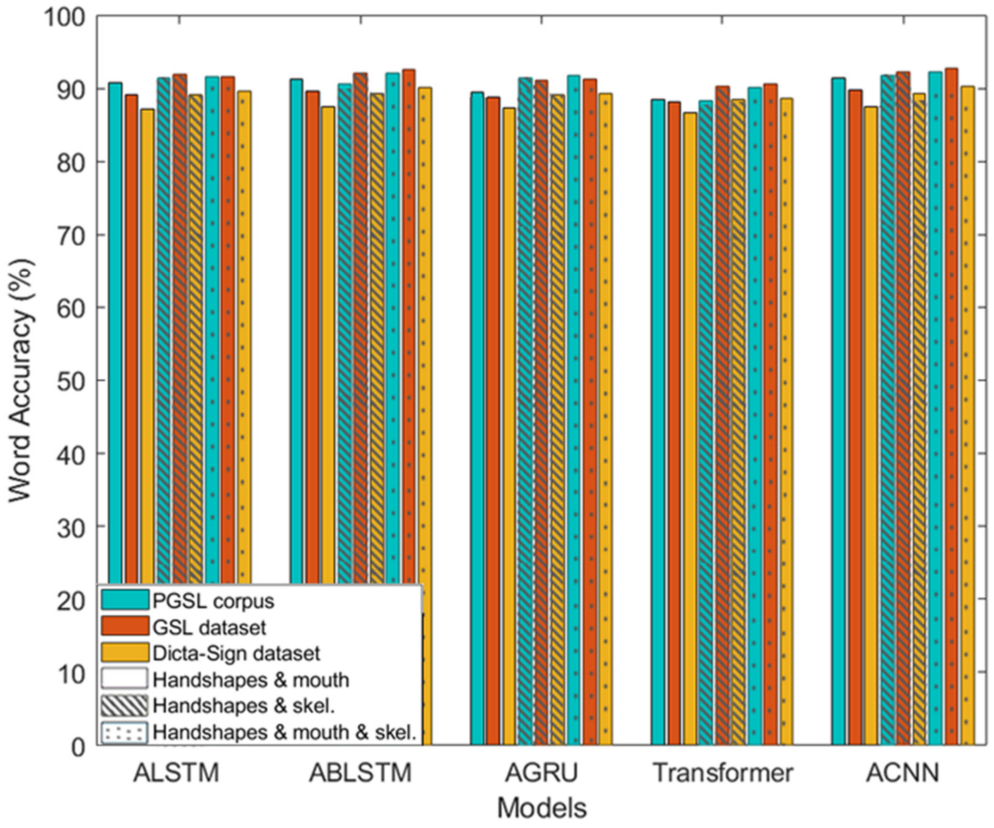

Figure 10: Word accuracy (%) of the evaluated encoder-decoder models on the three isolated GSL datasets, when concatenating various feature streams of Section 2 (handshapes + mouth; handshapes + human skeleton; all three feature streams).

5 Continuous Fingerspelling Recognition in ASL

The second SL recognition task considered in D2.1 is that of continuous fingerspelling. This involves a small vocabulary of signs (alphabet letters), however its continuous nature (i.e., the unconstrained letter length of fingerspelled words) presents orthogonal challenges to the task of isolated recognition of Section 4. As mentioned in the Introduction, we consider this task for the American sign language (ASL), due to the fact that corresponding data in GSL will only be available in later stages of SL-ReDu, as part of its WP3 activities. In the task presentation, we follow a similar structure to that of Section 4.

5.1 Model Implementation Details

The fingerspelling recognition task can be treated as a sequence-to-sequence prediction problem addressed by the attentional encoder-decoder general approach of Section 3. According to this methodology, a source word (sequence of letters) is observed via a sequence of raw image frames that is processed to produce a sequence of features, as discussed in Section 2. These are then passed through an attention-based encoder-decoder module, producing a sequence of letters as output. Like most sequence-to-sequence prediction problems, there is no one-to-one alignment between the input and output sequences, as several frames (and their corresponding feature vectors) can be associated with each letter. More specifically, we consider for the fingerspelling task the five attention-based sequence models of Section 3, which we implement as follows:

- Our *attentional LSTM encoder-decoder* (ALSTM) comprises a one-layer LSTM encoder and a one-layer LSTM decoder, both with hidden dimensionality equal to 128. Training is carried out via the Adam optimizer [73] with initial learning rate of 0.001 decreased by a factor of 0.3. The alignment scores are computed using the function proposed in [58].
- Our *attentional BLSTM encoder - LSTM decoder* (ABLSTM) includes a one-layer BLSTM for encoding and a one-layer LSTM decoder with 128 hidden units. For its training, the network employs the Adam optimizer with an initial learning rate of 0.001 decreased by a factor of 0.3. The alignment is obtained via the function in [58].
- Our *attentional GRU encoder-decoder* (AGRU) contains a single GRU layer at each model side with 256 hidden units. Its training is conducted via Adam with an initial learning rate of 0.001 decreased by a factor of 0.3, while its attention scores are calculated as in the preceding models.
- Our *transformer encoder-decoder* (Transformer) constitutes a 4-layer model with 8 heads for transformer self-attention, 2048-dimension hidden transformer feed-forward, and 512 hidden units. For its training we employ Adam with 0.001 initial learning rate decayed by a factor of 0.5, while for parameter initialization we conduct the Xavier process [74].
- Our *attentional CNN encoder-decoder* (ACNN) is a 3-layer CNN model with kernel width 5 and 256 hidden units. Its training is carried out via the Adam optimizer with an initial learning rate of 0.003 decreased by a factor of 0.1, and its attention score is calculated based on the dot alignment function [59]. In addition, the model is complemented with an input feeding scheme.

All models are trained employing a dropout rate of 0.3 with a mini-batch size fixed to 256. To achieve a better matching of a target element, the beam search strategy [75] with beam width of 5 during decoding is applied. In addition, a label smoothing [63] rate of 0.5 is employed for reducing over-confidence in predictions. In all cases, GPU acceleration is used for training and evaluation.

5.2 Database

The performance of the aforementioned algorithms is assessed on the publicly available *ChicagoFSWild dataset* [30]. This corpus includes clips of ASL fingerspelling sequences collected from online videos at 640×360-pixel frame resolution, providing data in real-world settings. The database was annotated using ELAN [76, 77] by students that have studied ASL. The corpus consists of 7,304 ASL fingerspelling sequences expressed by 160 signers, with 640×360-pixel resolution, leading to a 3,553 unique word vocabulary. Here, we employ a small-vocabulary subset concerning 103 unique fingerspelled words, involving 26 English letters with a sufficient number of occurrences in the data, between 10 and 130 times in the corpus. These yield 3,076 video snippets of words (from 143 signers), obtained by the ELAN annotation time-stamps of the words of interest. Training is conducted under a multi-signer setting, through ten-fold cross-validation with 80% of each fold used for training, 10% for validation, and 10% for testing. Example frames from the database are depicted in Figure 11.

Figure 11: Example frames of the ChicagoFSWild dataset, used here for continuous ASL fingerspelling.

5.3 Experimental Results

In Table 1 we report the performance of the various sequence-learning techniques of Section 3 for ASL fingerspelling recognition on the ChicagoFS-Wild database (in word accuracy (%), in all cases), when employing the individual feature representation streams of Section 2 (handshapes, mouth, and skeletal keypoints) alone, as well as some of their combinations.

Handshape (1024-dim)	**Mouth** (512-dim)	**Body & hand points** (144-dim)	**ALSTM**	**ABLSTM**	**AGRU**	**Transformer**	**ACNN**
x			84.12	84.58	81.06	83.44	84.71
	x		23.31	23.44	22.89	22.36	23.57
		x	79.64	80.04	79.73	78.92	80.19
x	x		85.27	86.37	83.27	83.82	86.54
x		x	86.03	89.25	84.15	85.12	90.81
x	x	x	**86.11**	**90.54**	**84.34**	**85.20**	**91.01**

Table 1: ASL fingerspelling results (in word accuracy (%)) on the ChicagoFS-Wild database using the feature streams of Section 2, individually or in combinations, in conjunction with the attention-based encoder-decoder models of Section 3.

As can be seen in the table, among the single feature streams (upper 3 lines of the table results), the handshape features perform the best, achieving the highest accuracies on all evaluated encoder-decoder models, with the skeletal keypoint representation being slightly worse and the mouth appearance features performing quite poorly. Nevertheless, the combination of handshape features with mouth appearance ones, as well as with skeletal keypoints of the body and hands, improves performance, demonstrating that all three representations carry useful SL information, complementary to each other. Further, the concatenation of all feature streams yields the best results for all five recognition models.

In addition, and similarly to the isolated GSL task (Section 4.3), it can be observed that the best results are obtained by the attentional CNN encoder-decoder, revealing its superiority to the considered modeling alternatives. Notably, the attentional BLSTM encoder - LSTM decoder performs quite close to it, and significantly better than both other RNN alternatives (ALSTM and AGRU) and the Transformer.

Finally, Table 2 reports the number of model parameters in millions for all five models considered. It can be observed that the best-performing model (ACNN) has significantly more parameters than the RNN-based encoder-decoder approaches (ALSTM, ABLSTM, and AGRU), but is much leaner than the transformer model. Taking also into consideration the recognition results of Table 1, it seems that the best modeling choices are provided by the ACNN and ABLSTM models.

Model	**ALSTM**	**ABLSTM**	**AGRU**	**Transformer**	**ACNN**
# Parameters (M)	0.554	0.619	0.734	7.321	3.007

Table 2: Number of model parameters (in million) for all encoder-decoder models of Section 3.

6 Conclusions

In this deliverable, we presented the SL-ReDu project initial approach on the classifier design for SL recognition, as part of the WP2 project activities. The work follows our WP1 activities concerning visual tracking and feature extraction that was reported in D1.1 and appropriately extended here, as well as our WP3 activities on data resource harvesting, as reported in D3.1. Based on these, we developed a number of attention-based encoder-decoder architectures operating on visual features and evaluated them on four databases, so as to decide on the best modeling approach. Specifically, we considered two recognition tasks, that of isolated signs of GSL and a second one concerning continuous sequences of fingerspelled characters of ASL, thus exploring tasks with different characteristics and challenges. Based on our evaluation, we determined that the attentional CNN encoder-decoder architecture is the most appropriate, providing the best performance on all four datasets considered for both recognition tasks. The deliverable will be extended in our upcoming WP2 activities in the form of D2.2 and D2.3, addressing more complex GSL recognition problems.

References

[1] K. Grobel and M. Assan, “Isolated sign language recognition using hidden Markov models,” in *Proc. International Conference on Systems, Man, and Cybernetics*, vol. 1, pp. 162–167, 1997.

[2] T. Starner, J. Weaver, and A. Pentland, “Real-time American sign language recognition using desk and wearable computer-based video,” *Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.

[3] C. Vogler and D. Metaxas, “Parallel hidden Markov models for American Sign Language recognition,” in *Proc. IEEE International Conference on Computer Vision*, vol. 1, pp. 116–122, 1999.

[4] P. Dreuw, D. Rybach, T. Deselaers, M. Zahedi, and H. Ney, “Speech recognition techniques for a sign language recognition system,” in *Proc. Annual Conference of the International Speech Communication Association*, pp. 2513–2516, 2007.

[5] S. Theodorakis, A. Katsamanis, and P.Maragos, “Product-HMMs for automatic sign language recognition,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1601–1604, 2009.

[6] V. Pitsikalis, S. Theodorakis, C. Vogler, and P. Maragos, “Advances in phonetics-based subunit modeling for transcription alignment and sign language recognition,” in *Proc. CVPR Workshops*, 2011.

[7] A. Roussos, S. Theodorakis, V. Pitsikalis, and P. Maragos, “Dynamic affine-invariant shape-appearance handshape features and classification in sign language videos,” *Journal of Machine Learning Research*, vol. 14, no. 15, pp. 1627–1663, 2013.

[8] S. Theodorakis, V. Pitsikalis, and P. Maragos, “Dynamic-static unsupervised sequentiality, statistical subunits and lexicon for sign language recognition,” *Image and Vision Computing*, vol. 32, no. 8, pp. 533–549, 2014.

[9] L. Pigou, S. Dieleman, P. Kindermans, and B. Schrauwen, “Sign language recognition using convolutional neural networks,” in *Proc. European Conference on Computer Vision*, pp. 572–578, 2015.

[10] D. Konstantinidis, K. Dimitropoulos, and P. Daras, “A deep learning approach for analyzing video and skeletal features in sign language recognition,” in *Proc. IEEE International Conference on Imaging Systems and Techniques*, 2018.

[11] B. G. Lee and S. M. Lee, “Smart wearable hand device for sign language interpretation system with sensors fusion,” *IEEE Sensors Journal*, vol. 18, no. 3, pp. 1224–1232, 2018.

[12] S. Ko, J. Son, and H. Jung, “Sign language recognition with recurrent neural network using human keypoint detection,” in *Proc. Conference on Research in Adaptive and Convergent Systems*, pp. 326–328, 2018.

[13] F. Nugraha and E. C. Djamal, “Video recognition of American sign language using two-stream convolution neural networks,” in *Proc. International Conference on Electrical Engineering and Informatics*, pp. 400–405, 2019.

[14] A. Mittal, P. Kumar, P. P. Roy, R. Balasubramanian, and B. B. Chaudhuri, “A modified LSTM model for continuous sign language recognition using Leap Motion,” *IEEE Sensors Journal*, vol. 19, no. 16, pp. 7056–7063, 2019.

[15] J. Pu, W. Zhou, and H. Li, “Iterative alignment network for continuous sign language recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4160–4169, 2019.

[16] Z. Yang, Z. Shi, X. Shen, and Y. Tai, “SF-Net: Structured feature network for continuous sign language recognition,” *Computing Research Repository*, arXiv:1908.01341, 2019.

[17] G. Potamianos, K. Papadimitriou, E. Efthimiou, S.-E. Fotinea, G. Sapountzaki, and P. Maragos, “SL-ReDu: Greek sign language recognition for educational applications. Project description and early results,” in *Proc. ACM International Conference on PErvarsive Technologies Related to Assistive Environments,* 2020.

[18] M. Parelli, K. Papadimitriou, G. Potamianos**,** G. Pavlakos, and P. Maragos, “Exploiting 3D hand pose estimation in deep learning-based sign language recognition from RGB videos,” *Computer Vision – ECCV 2020 Workshops Proceedings, Part II,* A. Bartoli and A. Fusiello (Eds.), pp. 249–263, LNCS/LNIP vol. 12536, 2020.

[19] K. Papadimitriou and G. Potamianos, "Multimodal sign language recognition via temporal deformable convolutional sequence learning," in *Proc. Conference of the International Speech Communication Association*, pp. 2752–2756, 2020.

[20] N. M. Adaloglou, T. Chatzis, I. Papastratis, A. Stergioulas, G. Th. Papadopoulos, V. Zacharopoulou, G. Xydopoulos, K. Antzakas, D. Papazachariou, and P. Daras, "A comprehensive study on deep learning-based methods for sign language recognition," *IEEE Transactions on Multimedia* (Early Access), 2021.

[21] P. Goh and E.-j. Holden, "Dynamic fingerspelling recognition using geometric and motion features," in *Proc. International Conference on Image Processing*, pp. 2741–2744, 2006.

[22] S. Ricco and C. Tomasi, "Fingerspelling recognition through classification of letter-to-letter transitions," in *Proc. Asian Conference on Computer Vision*, pp. 214–225, 2009.

[23] S. Liwicki and M. Everingham, "Automatic recognition of fingerspelled words in British Sign Language," in *Proc. IEEE Computer Society International Conference on Computer Vision and Pattern Recognition Workshops*, pp. 50–57, 2009.

[24] N. Pugeault and R. Bowden, "Spelling it out: Real-time ASL fingerspelling recognition," in *Proc. IEEE International Conference on Computer Vision Workshops*, pp. 1114–1119, 2011.

[25] T. Kim, W. Wang, H. Tang, and K. Livescu, "Signer-independent fingerspelling recognition with deep neural network adaptation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6160–6164, 2016.

[26] T. Kim, G. Shakhnarovich, and K. Livescu, "Fingerspelling recognition with semi-Markov conditional random fields," in *Proc. IEEE International Conference on Computer Vision*, pp. 1521–1528, 2013.

[27] T. Kim, J. Keane, W. Wang, H. Tang, J. Riggle, G. Shakhnarovich, D. Brentari, and K. Livescu, "Lexicon-free fingerspelling recognition from video: data, models, and signer adaptation," *Computer Speech and Language*, vol. 46, pp. 209–232, 2017.

[28] B. Shi and K. Livescu, "Multitask training with unlabeled data for end-to-end sign language fingerspelling recognition", in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 389–396, 2017.

[29] B. Shi, A. M. D. Rio, J. Keane, J. Michaux, D. Brentari, G. Shakhnarovich, and K. Livescu, "American sign language fingerspelling recognition in the wild," in *Proc. IEEE Spoken Language Technology Workshop*, pp. 145–152, 2018.

[30] B. Shi, A. M. D. Rio, J. Keane, D. Brentari, G. Shakhnarovich, and K. Livescu, "Fingerspelling recognition in the wild with iterative visual attention," in *Proc. IEEE International Conference on Computer Vision*, pp. 5399–5408, 2019.

[31] K. Papadimitriou and G. Potamianos, "End-to-end convolutional sequence learning for ASL fingerspelling recognition," in *Proc. Annual Conference of the International Speech Communication Association*, pp. 2315–2319, 2019.

[32] D. F. Armstrong, M. A. Karchmer, and J. V. Van Cleve, eds., *The Study of Signed Languages: Essays in Honor of William C. Stoko*e. Gallaudet University Press, Washington DC, USA, 2002.

[33] K. Antzakas and B. Woll, "Head movements and negation in Greek sign language." In I. Wachsmuth and T. Sowa (eds.), *Gesture and Sign Language in Human-Computer Interaction*, LNCS vol. 2298, pp. 193–196, Springer, 2002.

[34] K. Antzakas, "The use of negative head movements in Greek sign language." In U. Zeshan (ed.), *Interrogative and Negative Constructions in Sign Language*s, Ishara Press, pp. 258–269, 2006.

[35] K. Schönström and J. Mesch, "Use of nonmanuals by adult L2 signers in Swedish Sign Language – Annotating the nonmanuals," in *Proc. Workshop on the Representation and Processing of Sign Languages: Beyond the Manual Channel (Satellite to the International Conference on Language Resources and Evaluation)*, pp. 153–156, 2014.

[36] M. Brennan, "Word order: Introducing the issues." In: M. Brennan and G. Turner (eds.), *Word-Order Issues in Sign Language: Working Papers*, pp. 9–46, Int. Sign Linguistics Assoc., 1994.

[37] K. Papadimitriou, G. Potamianos, E. Efthimiou, S.-E. Fotinea, and P. Maragos, "D1.1: First version of visual tracking and feature extraction components," *Tech. Report, SL-ReDu Project Deliverable*, Volos, Greece, 2020.

[38] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation using part affinity fields,” in *Proc. IEEE Conference on Computer Vision and Patter Recognition (CVPR)*, pp. 1302–1310, 2017.

[39] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstrapping,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 4645–4653, 2017.

[40] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: Realtime multi-person 2D pose estimation using part affinity fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021.

[41] Y. LeCun and Y. Bengio, “Convolutional neural networks for images, speech, and time series.” In M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 255–258, MIT Press, 1998.

[42] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, *Nature*, 521: 436–444, 2015.

[43] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. International Conference on Learning Representations*, 2015.

[44] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Computing Research Repository*, arXiv:1512.03385, 2015.

[45] P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, “Large-scale isolated gesture recognition using convolutional neural networks,” in *Proc. International Conference on Pattern Recognition*, pp. 7–12, 2016.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[47] P. Wang, Z. Li, Y. Hou, and W. Li, “Action recognition based on joint trajectory maps using convolutional neural networks,” in *Proc. ACM International Conference on Multimedia*, pp. 102–106, 2016.

[48] Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1110–1118, 2015.

[49] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, “Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks,” in *Proc. AAAI Conference on Artificial Intelligence*, pp. 3697–3703, 2016.

[50] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3D residual networks,” in *Proc. International Conference on Computer Vision*, pp. 5534–5542, 2017.

[51] P. Lei and S. Todorovic, “Temporal deformable residual networks for action segmentation in videos,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6742–6751, 2018.

[52] J. Chen, Y. Pan, Y. Li, T. Yao, H. Chao, and T. Mei, “Temporal deformable convolutional encoder-decoder networks for video captioning,” in *Proc. AAAI Conference on Artificial Intelligence*, pp. 8167–8174, 2019.

[53] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”, *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[54] D. Yu and L. Deng, *Automatic Speech Recognition – A Deep Learning Approach*. Springer-Verlag, London, 2015.

[55] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, “Achieving human parity in conversational speech recognition,” *Computing Research Repository*, arXiv:1610.05256v2, 2017.

[56] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *Computing Research Repository*, arXiv:1409.1259, 2014.

[57] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proc. Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

[58] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *Computing Research Repository*, arXiv:1409.0473, 2014.

[59] M. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *Computing Research Repository*, arXiv:1508.04025, 2015.

[60] J. Zhou, Y. Cao, X. Wang, P. Li, and W. Xu, “Deep recurrent models with fast-forward connections for neural machine translation,” *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 371–383, 2016.

[61] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *Computing Research Repository*, arXiv:1609.08144v2, 2016.

[62] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” *Computing Research Repository*, arXiv:1705.03122v3, 2017.

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need”, *Computing Research Repository*, arXiv:1706.03762, 2017.

[64] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, 1997.

[65] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[66] E. Efthimiou, K. Vasilaki, S.-E. Fotinea, A. Vacalopoulou, T. Goulas, and A.-L. Dimou, “The POLYTROPON parallel corpus,” in *Proc. Workshop on the Representation and Processing of Sign Languages: Involving the Language Community (Satellite to the International Conference on Language Resources and Evaluation)*, pp. 39–44, 2018.

[67] S. Matthes, T. Hanke, A. Regen, J. Storz, S. Worseck, E. Efthimiou, A.-L. Dimou, A. Braffort, J. Glauert, and E. Safar, “Dicta-Sign – Building a multilingual sign language corpus,” in *Proc. Workshop on the Representation and Processing of Sign Languages: Interaction Between Corpus and Lexicon (Satellite to the International Conference on Language Resources and Evaluation)*, pp. 117–122, 2012.

[68] E. Efthimiou, S.-E. Fotinea, G. Sapountzaki, and K. Papadimitriou, “D3.1: First version of data resources,” *Tech. Report, SL-ReDu Project Deliverable*, Volos, Greece, 2021.

[69] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[70] J. Zhang, J. Tang, and L.-R. Dai, “RNN-BLSTM based multi-pitch estimation,” in in *Proc. Annual Conference of the International Speech Communication Association*, pp. 1785–1789, 2016.

[71] A. Graves, “Generating sequences with recurrent neural networks,” *Computing Research Repository*, arXiv:1308.0850, 2013.

[72] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” in *Proc. International Conference on Machine Learning*, pp. 933–941, 2017.

[73] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Computing Research Repository*, arXiv:1412.6980, 2014.

[74] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feed forward neural networks,” in *Proc. International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 249–256, 2010.

[75] M. Freitag and Y. Al-Onaizan, “Beam search strategies for neural machine translation,” *Computing Research Repository*, arXiv:1702.01806, 2017.

[76] ELAN (Version 5.8) [Computer software]. Nijmegen: Max Planck Institute for Psycholinguistics, The Language Archive [Online:] https://archive.mpi.nl/tla/elan.

[77] O. Crasborn and H. Sloetjes, “Enhanced ELAN functionality for sign language corpora,” in *Proc. Workshop on the Representation and Processing of Sign Languages: Construction and Exploitation of Sign Language Corpora (Satellite to the International Conference on Language Resources and Evaluation)*, pp. 39–43, 2008.

[78] T. Hanke, “iLex - A tool for sign language lexicography and corpus analysis,” in *Proc. International Conference on Language Resources and Evaluation*, pp. 923–926, 2002.