Jurnal Teknik Informatika Vol. 16 No. 2, October 2023 (170-184)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: : https://doi.org/10.15408/jti.v16i2.27030

JURNAL TEKNIK INFORMATIKA
Homepage : http://journal.uinjkt.ac.id/index.php/ti

Continuous Sign Language Recognition Using Combination of Two Stream 3DCNN and SubUNet

Haryo Pramanto[1], Suharjito[2]

[1,2]Computer Science Department, BINUS Graduate Program
[1,2]Master of Computer Science, Bina Nusantara University

[1,2]Jakarta, Indonesia 11480.

Article:
Accepted: June 18, 2023
Revised: March 11, 2023
Issued: October 28, 2023

***Correspondence Address:**
haryo.pramanto@binus.ac.id

ABSTRACT

Research on sign language recognition using deep learning has been carried out by many researchers in the field of computer science but there are still obstacles in achieving the expected level of accuracy. Not a few researchers who want to do research for Continuous Sign Language Recognition but are trapped into research for Isolated Sign Language Recognition. The purpose of this study was to find the best method for performing Continuous Sign Language Recognition using Deep Learning. The 2014 RWTH-PHOENIX-Weather dataset was used in this study. The dataset was obtained from a literature study conducted to find datasets that are commonly used in Continuous Sign Language Recognition research. The dataset is used to develop the proposed method. The combination of 3DCNN, LSTM and CTC models is used to form part of the proposed method architecture. The collected dataset is also converted into an Optical Flow frame sequence to be used as Two Stream input along with the original RGB frame sequence. Word Error Rate on the prediction results is used to review the performance of the developed method. Through this research, the best achieved Word Error Rate is 94.1% using the C3D BLSTM CTC model with spatio stream input.

Keywords: *continuous sign language recognition, two stream mode, 3DCNN, LSTM, CTC*

1. INTRODUCTION

Humans are social creatures. With that in mind, it's only natural for an interaction to be needed by them, all in order to fulfil their daily needs [1]. Oral communication is the one that is generally used when it comes to direct interaction. The reason is that it can be directly spoken by the communicator. Also, it can be directly heard and understood by the communicant [2]. However, this is an obstacle for humans who are with hearing disability and speech impairment. Humans who have these limitations, will generally use gestures, or even sign language to interact with each other. Communication using sign language is in fact highly ineffective when it is demonstrated towards humans who do not understand sign language. This obstacle will certainly be a challenge in everyday life, especially when spoken language users have to meet with sign language users to carry out activities together.

Many researchers are inspired by this challenge. Solutions have been aimed to be found, in hope of communication support that could be used either by spoken language users, and sign language users. In the field of computer science, researchers have developed various approaches so that computers are able to recognize sign language (Sign Language Recognition / SLR) and translate it into a natural language whose output can be text or audio. Some are using a glove-based approach [3] and some are using a vision-based approach [4]. Among these approaches is a vision-based approach that uses a single camera [5].

Deep learning is a part of the AI discipline that mimics the performance of a human brain in terms of processing data and understanding patterns from that data to predict a decision [6]. Patterns of association can be immediately found by researchers with the use of deep learning. In a certain problem, it's even rather quickly to spot patterns of association between the input data (features) in the input layer, and the output data (label) in the output layer, compared to having to do it manually one by one. Deep learning has also been widely used by researchers in helping build Sign Language Recognition. In order for deep learning to perform well, a dataset is needed. Especially one that can represent the problems of the selected research topic [7].

Sign Language Recognition using deep learning can be grouped based on the selected dataset problem, namely Static Sign Language Recognition (SSLR), Isolated Sign Language Recognition (ISLR), and Continuous Sign Language Recognition (CSLR) [8]. SSLR focuses on datasets in the form of static images. SSLR usually focuses on the recognition of letters and numbers in sign language [9] [10]. ISLR focuses on the dataset in the form of videos but usually each video contains one word in sign language [11] [12]. CSLR is sign language recognition that focuses on sentence-level recognition. The dataset used is a video recording containing one sentence in sign language [13] [14].

Many studies that had an initial aim to conduct CSLR research were trapped into ISLR studies [15]. As explained in the previous explanation, a video dataset might've been used by both CSLR and ISLR, but there were also differences that exist in the focus of solving the problem. CSLR focuses on recognizing sign languages that demonstrate a complete sentence in one input video, while ISLR focuses on recognizing sign languages that demonstrate one word / symbol in one input video. Some studies also rely on static images to perform sign language recognition so that they are trapped in SSLR research. Various results of near-perfect speed and accuracy were generated through SSLR research. Among the researchers using the SSLR method are Singha and Das [16]. However, in practice in real life the sign language used is quite complex and is not limited to a series of alphabets. There are various combinations of gestures and gestures that indicate an activity in sign language that is not sufficiently recognizable by static images alone. The SLR needs to recognize the motion of the cue which is a series of still images (frames). The video dataset can be used for SLR training in recognizing these gesture movements [17].

Among the researchers who conducted research on SLR with video datasets were Huang et al. [15]. They use CNN's Two Stream 3D method with LS-HAN Framework. The results were satisfactory and there was an increase in accuracy compared to the LSTM, S2VT, LSTM-A, LSTM-E and Hierarchical Attention Network (HAN) methods they chose as their comparison. CNN's Two Stream 3D method extracts features from Video SLR by processing one frame into global streams and local streams. Global flow extracts the overall frame to capture a wider range of limb movements, and local flows are used to extract frames that focus on gestures of the hand only. Huang et al. inspired by the work of Tran et al. [18] to build a model of the 3DCNN. LS-HAN or Latent Space – Hierarchical Attention Network is a framework proposed by Huang et al. [15] who make use of the HAN [19] and the LS model [20]. HAN is an extension of Long Short-Term Memory (LSTM) that pays attention to information structures and latency mechanisms in order to provide output in the form of whole sentences word for word from a given SLR video. Combined with the Latent Space model so that the relationship between the resulting sentence and the video input can be more unified.

Research on CSLR was also conducted by Camgoz et al. [21]. They carry the SubUNet method. SubUNet is a neural network model classification method that breaks the focus of learning into several subunits. This subunit is devoted to the sub-network of temporal spatial learning for CSLR so that it is able to assemble the meaning of a series of movements / motions (tasks) in CSLR into complete equivalent sentences. In that study, Camgoz et al. defines a subunit as a fraction of a sequence of movements in temporal spatial learning, where in the context of continuous sign language recognition this means gestures or small movements that exist in a series of motion of a particular word or sentence [21]. SubUNet uses Two-Dimensional Convolutional Neural Network (2DCNN) [22] as feature extraction whose results are used for processing the spatial temporal learning sub-unit in continuous sign language recognition. The use of 2DCNN in the method that was promoted because the scope of their research was only for the concept of a series of gestures / movements in the local area of the fist. In contrast to other video-to-text approaches, the method they use in addition to modeling the subunits using a series of gestures as input, also carries out training through these series of motion.

In 2019, Cui et al [23] also built a deep learning model for CSLR by leveraging the Two Stream Model. Cui et al. utilized the same dataset as Camgoz et al. [21] namely RWTH-PHOENIX-Weather 2014: Continuous Sign Language Recognition Dataset [24]. In their research, Cui et al. prepare two types of dataset, namely RGB frame sequence and Optical Flow frame sequence. The RGB frame sequence is used to get the spatial features from the video dataset which becomes the input for its neural network model. Optical flow frame sequences are used to obtain more accurate temporal features by reading the pixel movements of the video sign language gestures that are input to the neural network model. The neural network model he built for feature extraction in RGB frame sequences and Optical Flow frame sequences is the same. Cui et al. utilize the Temporal Convolutional Network [25] to obtain spatial temporal features from the two input frame sequences. The results of the two features extraction are then combined (fusion) and directed to sequence to sequence learning using the Bidirectional Long Short Term Model (BLSTM) to obtain the predictive output of word sequences which become annotations of the video sign language used as input.

Based on research conducted by Huang et al. [15], Camgoz et al. [21] and Cui et al. [23], this study tried to combine their method with the title "Continuous Sign Language Recognition Using Combination of Two Stream 3DCNN and SubUNet". The advantage of SubUNet is that it is not only able to encode expert knowledge of a gesture / motion but also transfer learning between a series of gestures that have been studied by the previous series of processes. The advantage of Two Stream 3DCNN which is able to answer the challenges of continuous sign language recognition is by inputting a video which is then split into a global area to see the overall motion and a local area to see the movement of the hand area. Then recombine the meaning that has been obtained as output. The advantages of the research of Cui et al. which utilizes Optical Flow frame sequences to obtain more accurate spatial temporal features by inputting them into the Two Stream Model CSLR along with their RGB frame sequences.

It is hoped that by combining the works of previous SLR researchers, it can answer the challenges and fill in the weaknesses of their research. In the study of Huang et al. [15] namely dealing with latent spatial that occurs. In a study by Camgoz et al [21], the input that only uses 2DCNN for the local area of the hand does not include the overall global area of body movement as a whole. In the research of Cui et al. [23] who only utilize 1DCNN for its temporal convolution. With this research, we also hope to contribute to increasing the speed and accuracy of research in the field of Sign Language Recognition.

2. METHODS

2.1. Research Methodology

This research was done by following 6 steps of research methodology (see Figure 1). Research preparations are conducted by choosing research topics and identify backgrounds and problems. By determining these three things we can more focus for literature study as the next step. Related works are studied in this step for finding the sufficient solutions, models and dataset. The third step is to collecting dataset for this research. Next step is building the proposed models, testing them with the prepared dataset and compare the accuracy for evaluating the proposed models.

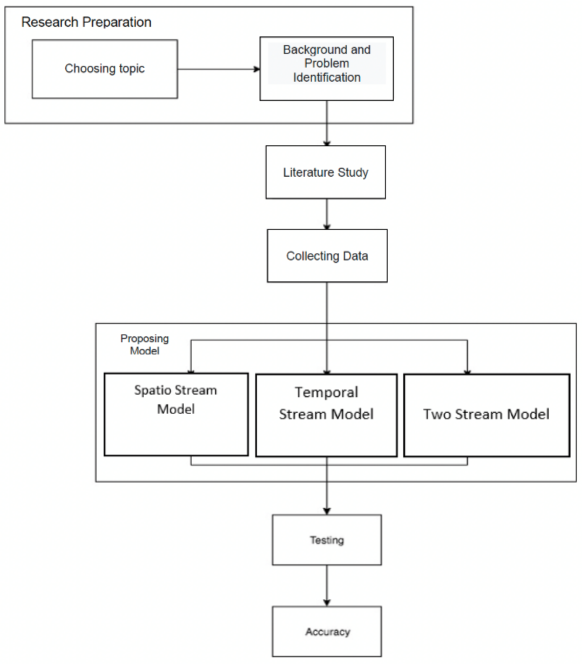


Figure 1.Research methodology

2.2. Proposed Model

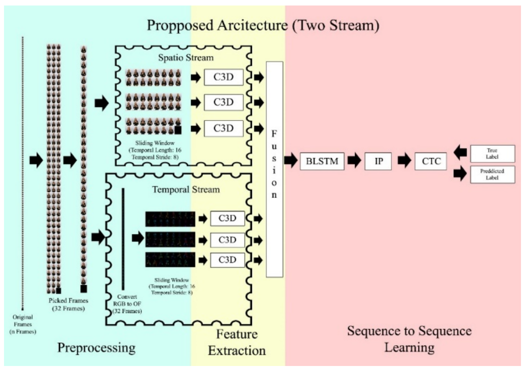

Proppposed Arcitecture (Two Stream)
Spatio Stream
C3D
C3D
C3D
Sliding Window (Temporal Length: 16 Temporal Stride: 8)
Temporal Stream
C3D
C3D
C3D
Sliding Window (Temporal Length: 16 Temporal Stride: 8)
Convert RGB to OF (32 Frames)
Fusion
BLSTM
IP
CTC
True Label
Preddicted Label
Original Frames (n Frames)
Picked Frames (32 Frames)
Preprocessing
Feature Extraction
Sequence to Sequence Learning

Figure 2. Proposed architecture with two stream feature extraction

The architectural design that we built was inspired by the research of Camgoz et al. [21], Huang et al. [15], and Cui et al. [23]. The neural network model is built in parallel (Two Stream) by separating between Spatial Extraction and Temporal Extraction and then the results are recombined and linked with LSTM and CTC (see Figure 2). The video input is converted to RGB frame sequence. The RGB frame sequence is then copied and converted to Optical Flow frame sequence. The spatial features are extracted from the RGB frame sequence which is the input for the 3DCNN model. The temporal features are extracted from the Optical Flow frame sequence which also becomes the input for 3DCNN model. The results of the feature extraction are then combined and then directed to sequence to sequence learning to obtain the prediction results of annotated words that are sequential to form a sentence of ordered annotation.

In SubUNet Model [21] Camgoz et al. state that their feature extraction model can exploit any Convolution Neural Network (CNN) architecture for spatial modeling. They choose CaffeNet with ImageNet weight for their feature extraction. For our experiment we choose Tran et al. C3D [18] as the three dimensional convolution neural network to extract features from the input frames. C3D is made to support Three-Dimensional Convolutional Networks. C3D can be used to train, test, or fine-tune 3D ConvNets efficiently. C3D is consist of five block of convolution neural network and followed with two fully connected layer of 4096 unit each. Originally C3D is use for video classification in action recognition. C3D can extract feature form Fame Sequence of video to predict what action in it. We implement C3D with transfer learning from available C3D Sports-1M weight.

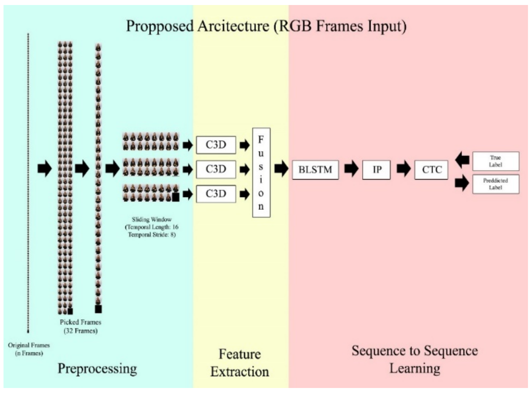

Proppposed Arcitecture (RGB Frames Input)
Original Frames (n Frames)
Picked Frames (32 Frames)
Sliding Window (Temporal Length: 16 Temporal Stride: 8)
C3D
C3D
C3D
Fusion
BLSTM
IP
CTC
True Label
Preddicted Label
Preprocessing
Feature Extraction
Sequence to Sequence Learning

Figure 3.Proposed architecture with single spatio stream RGB frames input feature extraction

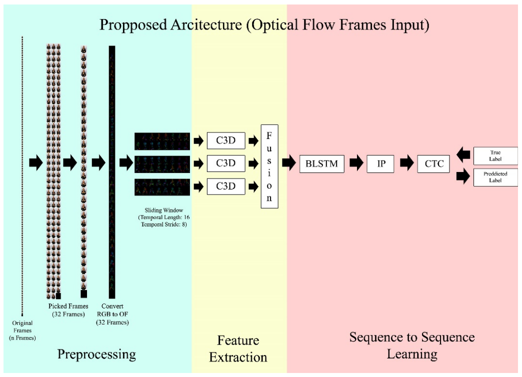

Proppposed Arcitecture (Optical Flow Frames Input)
Original Frames (n Frames)
Picked Frames (32 Frames)
Convert RGB to OF (32 Frames)
Sliding Window (Temporal Length: 16 Temporal Stride: 8)
C3D
C3D
C3D
Fusion
BLSTM
IP
CTC
True Label
Preddicted Label
Preprocessing
Feature Extraction
Sequence to Sequence Learning

Figure 4.Proposed architecture with single spatio stream optical flow frames input

In the sequence to sequence learning model we employ BLSTM as the language model, Inner Product (IP) /dense network for learn the vocabularies in the dataset and CTC to predict the ordered annotation. With this architecture we try to achieve end to end continuous sign language recognition that can handle video input to text directly. We also proposed separate architecture model with single stream of RGB only (see Figure 3) and Optical Flow only (see Figure 4). These three models than will be compared to see the best approach for achieving good end to end continuous sign language recognition.

2.3. Dataset

The dataset chosen as material for this research is RWTH-PHOENIX-Weather 2014: Continuous Sign Language Recognition Dataset [24]. The dataset is a German Sign Language demonstration recording (Deutsche Gebärdensprache / DGS) of the weather forecast broadcast of the German TV station Phoenix. Sign language recordings are recorded using a stationary single color camera that is positioned in front of the sign language display. The display of sign language is wearing black on a gray gradient computer background. Sign language demonstration footage is recorded at 25 frames per second at 210 x 260 Pixels. Each recording for one sentence has a different number of frames. There are many other dataset choices to be used as sign language recognition research [26] [27] [28] but we chose the 2014 RWTH-PHOENIX-Weather dataset because it fits the research needs. We choose it because continuous sign language recognition requires a dataset in the form of video input demonstrating sign language in a sentence and outputting the text of the sentence.

EINS ZWANZIG FLUSS __ON__ __OFF__ SONNTAG VIEL WOLKE REGEN

Figure 5.Example of frames sequence of a video and its annotation in 2014 RWTH-PHOENIX-Weather multi signer full frame dataset

Table 1.Multi signer in full frames sets

	Dev		Test		Train	
	Total Frames	Total Words	Total Frames	Total Words	Total Frames	Total Words
#	540	540	629	629	5672	5672
Mean	139	10	142	10	140	11
Min	32	3	39	3	16	2
max	251	23	251	21	299	28

The 2014 RWTH-PHOENIX-Weather dataset itself is consist of many sets such as sets of multi signer in full frames (see Table 1), sets of multi signer with tracked right hand only, and sets of single signer in full frames. Each sets has separated dev set, test set, and train set. For our experiment we use the sets of multi signer in full frames as our dataset.

2.4. Evaluation Method

The three methods that we proposed (see section II.B) will obtain a level of accuracy in predicting the meaning of sign language which is the input data. That way we can determine which proposed method is the best proposal to be presented as an alternative in deep learning CSLR. The evaluation for the model that the author proposes will be carried out by calculating the Word Error Rate (WER) of the annotation sequence predicted from the model [29]. WER is an evaluation method commonly used in CSLR research [24] [21] [23].

$$\boldsymbol{WER} = \frac{\boldsymbol{S+D+I}}{\boldsymbol{GT}} \qquad (1)$$

Where WER is the Word Error Rate, S is the number of substitution, D is the number of deletion, I is the number of insertion, while GT is the Ground Truth.

3. MODEL IMPLEMENTATION

Table 2.Multi signer in full frames sets that has 100 – 125 frames each video

	Dev		Test		Train	
	Total Frames	Total Words	Total Frames	Total Words	Total Frames	Total Words
#	122	122	125	125	1281	1281
Mean	113	8	112	8	112	9
Min	100	5	100	4	100	2
max	125	14	125	17	125	17

We build our model in Google Colab with Python, Keras and Tensorflow. We using the available provided GPU by Google Colab, sometimes NVDIA P100, sometimes NVDIA T4, sometimes NVDIA K80. Due to limited memory and session time out in Google Colab we only use the video in dataset that consist of 100 to 125 frames in single video. The details on the selected data can be seen in Table 2.

Since the dataset is contain various amount of frames in single video, our strategy is down sampled the frames to 32 frames for single video. The 32 frames are picked respectfully to keep it sorted as ordered as the original frames amount. If we play the picked 32 frames as video it just looks fast forward version of the original frames amount video. From that 32 picked frames then we applied sliding window technique and resize the frame size to 112 height and width to feed the frames sequence to C3D for feature extraction. Following the structure of C3D, the frame sequence needs to be in 16 frames length with 112 image width and height in 3 channels color of RGB. The C3D paper [18] also implement this sliding window technique by set 16 frames for temporal length and 8 frames temporal stride to overlap the previous frame sequence of the same video data source. We set our sliding window setup as the original paper of C3D do. The C3D structure that been uesd in this study can be seen in Figure 6.

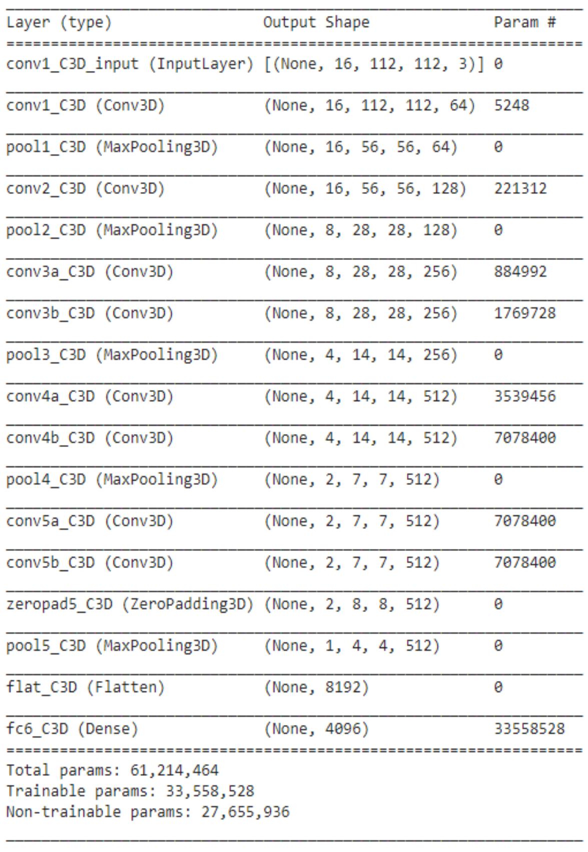
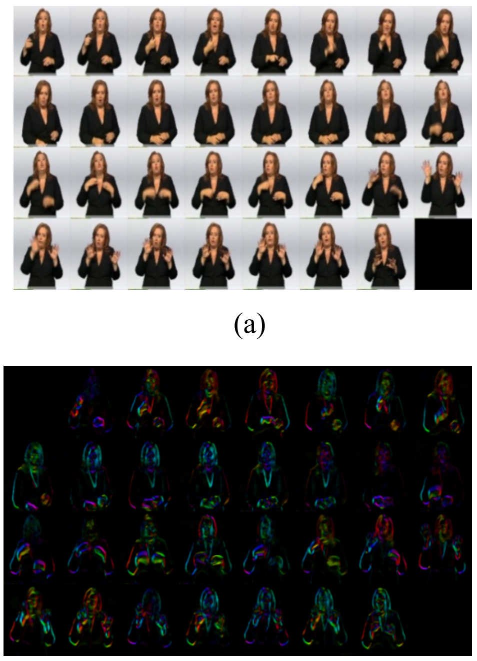

```
Layer (type)                  Output Shape                Param #
=================================================================
conv1_C3D_input (InputLayer) [(None, 16, 112, 112, 3)] 0
_________________________________________________________________
conv1_C3D (Conv3D)           (None, 16, 112, 112, 64)  5248
_________________________________________________________________
pool1_C3D (MaxPooling3D)     (None, 16, 56, 56, 64)    0
_________________________________________________________________
conv2_C3D (Conv3D)           (None, 16, 56, 56, 128)   221312
_________________________________________________________________
pool2_C3D (MaxPooling3D)     (None, 8, 28, 28, 128)    0
_________________________________________________________________
conv3a_C3D (Conv3D)          (None, 8, 28, 28, 256)    884992
_________________________________________________________________
conv3b_C3D (Conv3D)          (None, 8, 28, 28, 256)    1769728
_________________________________________________________________
pool3_C3D (MaxPooling3D)     (None, 4, 14, 14, 256)    0
_________________________________________________________________
conv4a_C3D (Conv3D)          (None, 4, 14, 14, 512)    3539456
_________________________________________________________________
conv4b_C3D (Conv3D)          (None, 4, 14, 14, 512)    7078400
_________________________________________________________________
pool4_C3D (MaxPooling3D)     (None, 2, 7, 7, 512)      0
_________________________________________________________________
conv5a_C3D (Conv3D)          (None, 2, 7, 7, 512)      7078400
_________________________________________________________________
conv5b_C3D (Conv3D)          (None, 2, 7, 7, 512)      7078400
_________________________________________________________________
zeropad5_C3D (ZeroPadding3D) (None, 2, 8, 8, 512)      0
_________________________________________________________________
pool5_C3D (MaxPooling3D)     (None, 1, 4, 4, 512)      0
_________________________________________________________________
flat_C3D (Flatten)           (None, 8192)              0
_________________________________________________________________
fc6_C3D (Dense)              (None, 4096)              33558528
=================================================================
Total params: 61,214,464
Trainable params: 33,558,528
Non-trainable params: 27,655,936
_________________________________________________________________
```

Figure 6.Model structure of C3D

This method also refers to the research of Cui et al. [14] [23]. The number of frames that enter the neural network model will be divided based on a certain temporal length and temporal stride. In several other studies this technique is also referred to as a sliding window [30] [31]. The temporal length is the number of frames that enter the neural network model in one batch of frames. Temporal Stride further shifts the frame into the neural network model being built. The settings on the temporal length and temporal stride should overlap between the next batch of frames so that the temporal information on one video is not interrupted. The illustration of the use of the temporal length and temporal stride can be seen in preprocessing part of Figure 2, Figure 3 and Figure 4.

Since we have 32 frames for each video to be feed in C3D and the sliding window setup is 16 frames temporal length and 8 frames temporal stride, we will have 3 slides to feed all the picked frames in. Each slide will have their own C3D model to do feature extraction. Then we fusion the extracted feature by concatenating the three output of three blocks of C3D feature extraction. The illustration of the use of this fusion can be seen in feature extraction part and Figure 4 . As for the Two Stream architecture, we need to do the extracting for each temporal stream and spatio stream. We convert from RGB to Optical Flow frame sequence for the spatio stream using TV-L1 optical flow from Python OpenCV library. TV-L1 optical flow in OpenCV library is based on Sánchez Pérez et al. [32] and Pock et al, [33] papers. We choose TV-L1 because it fast and generate clearer optical flow frames sequence as we can see in Figure 7. Each extracted feature of temporal stream and spatio stream will be concatenate before feed to the sequence to sequence learning part.

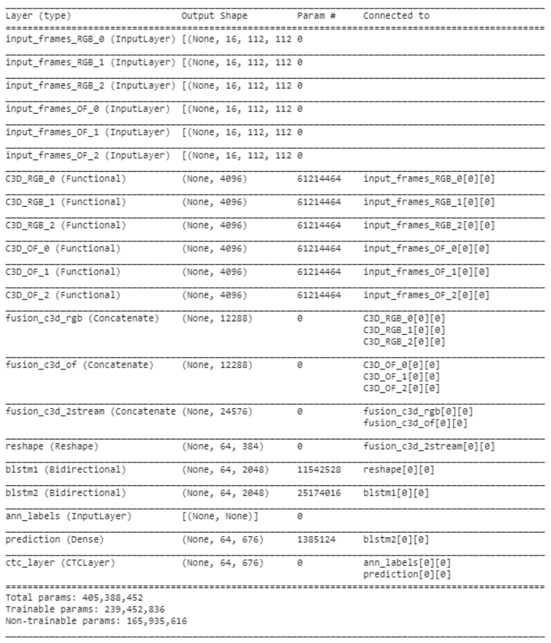

(a)

(b)

Figure 7.The example of 32 picked frames of a single video for spatio stream (a) and temporal stream (b). See Figure 5 for the original frames amount

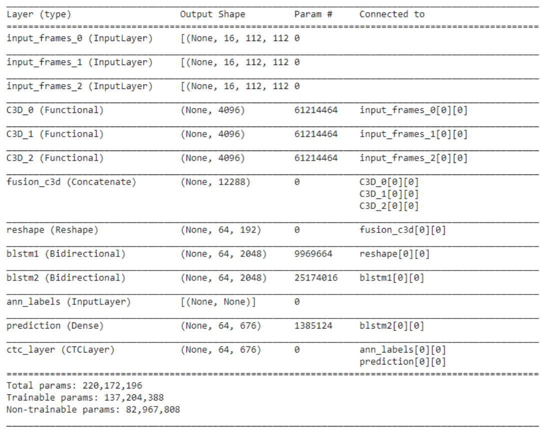
In the sequence to sequence learning part we use two BLSTM layers with 1024 units applied each of them. Since we use CTC to predict the ordered annotations labels we need to configure the time slices before entering the BLSTM layers. It being said that the best practice is should be more than “2 x maximum word available in the dataset + 1”. So after we do fusion for our extracted feature we need to reshape the result before entering the BLSTM layers. After the BLSTM layers we need to create Inner Product by adding dense layer with units of vocabulary size of the full dataset that we use + 1 extra vocabulary for blank entities that CTC need to predict the outcome. Finally CTC layer is added to predict the ordered predicted annotations labels.

In Figure 8 and Figure 9 are shown the structure of the Keras model used for this study. The structure of the two stream model can be seen in Figure 8 with a total of 405,388,452 parameters with 165,935,616 non-trainable parameters. The spatio stream and temporal stream model structure has a total of 405388452 parameters with non-trainable parameters of 82,967,808 parameters. the spatio stream and temporal stream models have the same structure because the differences only exist in the input data.



```
__________________________________________________________________________________________________
Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
input_frames_RGB_0 (InputLayer) [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_RGB_1 (InputLayer) [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_RGB_2 (InputLayer) [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_OF_0 (InputLayer)  [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_OF_1 (InputLayer)  [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_OF_2 (InputLayer)  [(None, 16, 112, 112 0
__________________________________________________________________________________________________
C3D_RGB_0 (Functional)          (None, 4096)         61214464    input_frames_RGB_0[0][0]
__________________________________________________________________________________________________
C3D_RGB_1 (Functional)          (None, 4096)         61214464    input_frames_RGB_1[0][0]
__________________________________________________________________________________________________
C3D_RGB_2 (Functional)          (None, 4096)         61214464    input_frames_RGB_2[0][0]
__________________________________________________________________________________________________
C3D_OF_0 (Functional)           (None, 4096)         61214464    input_frames_OF_0[0][0]
__________________________________________________________________________________________________
C3D_OF_1 (Functional)           (None, 4096)         61214464    input_frames_OF_1[0][0]
__________________________________________________________________________________________________
C3D_OF_2 (Functional)           (None, 4096)         61214464    input_frames_OF_2[0][0]
__________________________________________________________________________________________________
fusion_c3d_rgb (Concatenate)    (None, 12288)        0           C3D_RGB_0[0][0]
                                                                 C3D_RGB_1[0][0]
                                                                 C3D_RGB_2[0][0]
__________________________________________________________________________________________________
fusion_c3d_of (Concatenate)     (None, 12288)        0           C3D_OF_0[0][0]
                                                                 C3D_OF_1[0][0]
                                                                 C3D_OF_2[0][0]
__________________________________________________________________________________________________
fusion_c3d_2stream (Concatenate (None, 24576)        0           fusion_c3d_rgb[0][0]
                                                                 fusion_c3d_of[0][0]
__________________________________________________________________________________________________
reshape (Reshape)               (None, 64, 384)      0           fusion_c3d_2stream[0][0]
__________________________________________________________________________________________________
blstm1 (Bidirectional)          (None, 64, 2048)     11542528    reshape[0][0]
__________________________________________________________________________________________________
blstm2 (Bidirectional)          (None, 64, 2048)     25174016    blstm1[0][0]
__________________________________________________________________________________________________
ann_labels (InputLayer)         [(None, None)]       0
__________________________________________________________________________________________________
prediction (Dense)              (None, 64, 676)      1385124     blstm2[0][0]
__________________________________________________________________________________________________
ctc_layer (CTCLayer)            (None, 64, 676)      0           ann_labels[0][0]
                                                                 prediction[0][0]
==================================================================================================
Total params: 405,388,452
Trainable params: 239,452,836
Non-trainable params: 165,935,616
__________________________________________________________________________________________________
```

Figure 8.Model structure of proposed two stream model



```
__________________________________________________________________________________________________
Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
input_frames_0 (InputLayer)    [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_1 (InputLayer)    [(None, 16, 112, 112 0
__________________________________________________________________________________________________
input_frames_2 (InputLayer)    [(None, 16, 112, 112 0
__________________________________________________________________________________________________
C3D_0 (Functional)             (None, 4096)         61214464    input_frames_0[0][0]
__________________________________________________________________________________________________
C3D_1 (Functional)             (None, 4096)         61214464    input_frames_1[0][0]
__________________________________________________________________________________________________
C3D_2 (Functional)             (None, 4096)         61214464    input_frames_2[0][0]
__________________________________________________________________________________________________
fusion_c3d (Concatenate)       (None, 12288)        0           C3D_0[0][0]
                                                                C3D_1[0][0]
                                                                C3D_2[0][0]
__________________________________________________________________________________________________
reshape (Reshape)              (None, 64, 192)      0           fusion_c3d[0][0]
__________________________________________________________________________________________________
blstm1 (Bidirectional)         (None, 64, 2048)     9969664     reshape[0][0]
__________________________________________________________________________________________________
blstm2 (Bidirectional)         (None, 64, 2048)     25174016    blstm1[0][0]
__________________________________________________________________________________________________
ann_labels (InputLayer)        [(None, None)]       0
__________________________________________________________________________________________________
prediction (Dense)             (None, 64, 676)      1385124     blstm2[0][0]
__________________________________________________________________________________________________
ctc_layer (CTCLayer)           (None, 64, 676)      0           ann_labels[0][0]
                                                                prediction[0][0]
==================================================================================================
Total params: 220,172,196
Trainable params: 137,204,388
Non-trainable params: 82,967,808
__________________________________________________________________________________________________
```

Figure 9.Model structure of proposed single stream model (RGB/Optical Flow)

4. RESULTS AND DISCUSSIONS

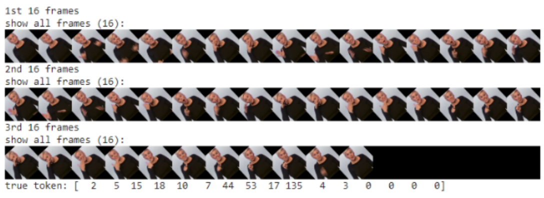
We feed the data in the same manner to our proposed model. First we prepare the annotation label. If we look closer to the annotation labels of the dataset we found that there is word "__ON__" and "__OFF__" inside the sentence. Most of the time the "__ON__" is put at the beginning of the sentence and the "__OFF__" is put at the end of the sentence. But the problem is it's not always like that. There is some annotation that doesn't have "__ON__" and "__OFF__" at all, some have them in the middle of the sentence and some have only one of it in the sentence.

Since the provided annotation from the dataset didn't have uniformed word that we can use as token to determine where the start and the end of the sentence is, we add our own token "<start>" and "<end>". We put "<start>" at the start of the sentence followed with " " space before followed with the actual sentence. We put "<end>" at the end of the sentence but we also put " " space before closed the sentence with "<end>". After that we create tokenizer using available library from Tensorflow to tokenize the words of the annotation labels. We only filter " " space that not to be tokenize and let other punctuation as part of word since the available punctuation is only "-" like in "IM-VERLAUF".

```
original labels of one single video:
IX MORGEN IM-VERLAUF IX ANFANG FREUNDLICH ABER
WOLKE KOMMEN OST

preprocessed labels:
<start> ix morgen im-verlauf ix anfang
freundlich aber wolke kommen ost <end> ? ? ? ?

Token of the preprocessed labels:
[  2   8   7  88   8 122  40  56  13  33  38
 3   0   0   0   0]
```

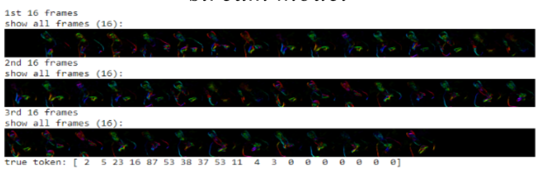
Figure 10.The preprocessed annotation labels and token (true labels and true token)

In Figure 10 notice that there are "?" in the preprocessed label and "0" in the token is mean to be paddings of the labels to the maximum word count in the sentence of the whole dataset we use. Why we doing this because at the time we write our code the Tensorflow doesn't support yet to generate data that consist of different length to feed to Keras model that we built.



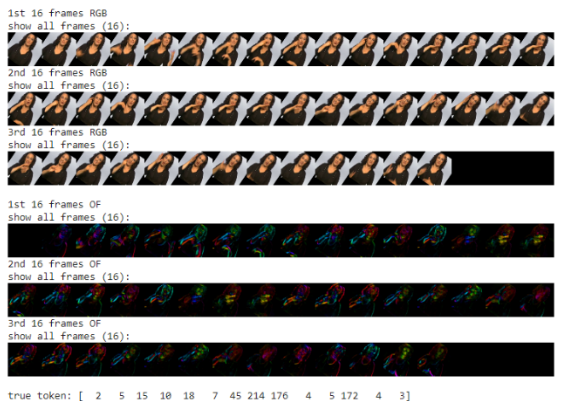
```
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [  2   5  15  18  10   7  44  53  17 135   4   3   0   0   0   0]
```

Figure 11. Example of input data for spatio stream model



```
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [ 2  5 23 16 87 53 38 37 53 11  4  3  0  0  0  0  0  0  0]
```

Figure 12.Example of input data for temporal stream proposed model



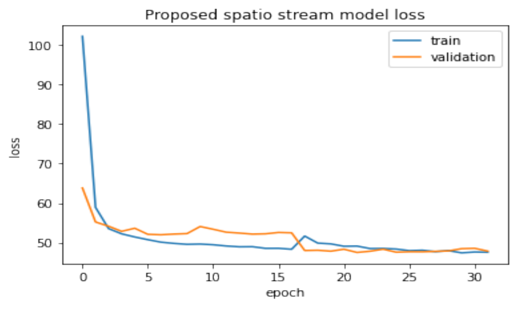
```
1st 16 frames RGB
show all frames (16):
2nd 16 frames RGB
show all frames (16):
3rd 16 frames RGB
show all frames (16):

1st 16 frames OF
show all frames (16):
2nd 16 frames OF
show all frames (16):
3rd 16 frames OF
show all frames (16):

true token: [  2   5  15  10  18   7  45 214 176   4   5 172   4   3]
```

Figure 13.Example of input data for two stream proposed model

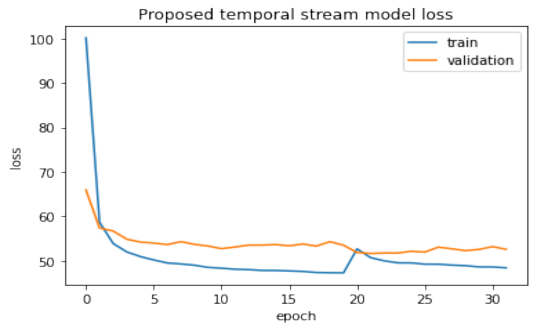
The data that being feed for our each proposed model can be seen in Figure 11, Figure 12 and Figure 13. We apply augmentation for the frames input. The augmentation that we use are center cropped and rotation. The original frame size of the dataset is 210 x 260 Pixels. We can't feed it to our feature extraction model because C3D needs to be feed with specific size of 112x112 Pixels. Before we center cropped each frame to 112x112 Pixels, we resize each frame to 135x135 Pixels. Why we doing this is for achieve clearer image that being feed to the feature extraction model. The black blank frames in the input data is padding to fit in sliding window with 16 frame temporal length and 8 frame temporal stride.

We split the 1281 train data into two parts and train them in two training sessions for each proposed model. Then we split again train data of each training session into 70% for training, 10% for validation and 20% for testing. Each training session conducted 20 epochs with mini batch size of 16 batches. We apply three callbacks on our training model. Early stop callback if there are no improvement achieve in the last 10 epochs. Training log callbacks to save the training loss and validation loss results for each epoch in csv file. Checkpoint callback to save model weight in Keras hdf5 file if it hits the lowest validation loss score. Every time we finish the training session we will save the model into h5 Keras saved model file. We do prediction with the h5 saved model file and continue the train in the second training session by load the hdf5 model weight file.

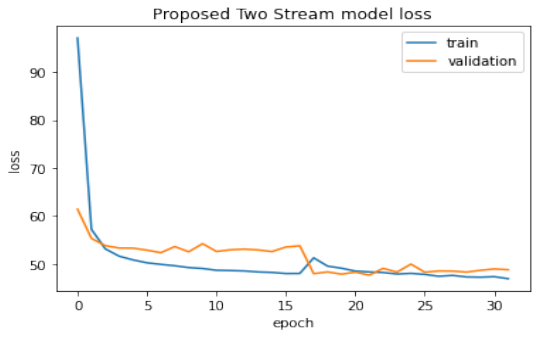
The overall training performance can be seen on Figure 14. Most of the result is achieve plateauing even though the loss score is not so great. We can see there is improvement when the transition happens between first training session and second training session by looking at the sudden drop of the loss score. We can also see that not all training session is completely train in 20 epochs because we apply early stop callback when there is no improvement in validation score. If we choose the better result based on the gap between training loss and the validation, the spatio stream model is achieving better result than other our proposed model.



(a)

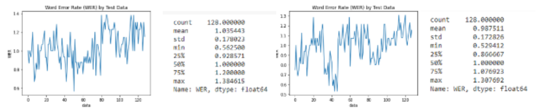


(b)



(c)

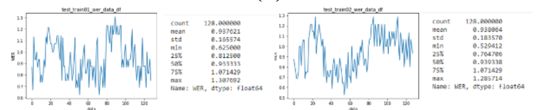
Figure 14.Model training loss of each proposed model (a) spatio stream proposed model (b) temporal stream proposed model (c) two stream proposed model



```
count    128.000000
mean       1.035443
std        0.178023
min        0.562500
25%        0.928571
50%        1.000000
75%        1.200000
max        1.384615
Name: WER, dtype: float64
```

```
count    128.000000
mean       0.987511
std        0.172826
min        0.529412
25%        0.866667
50%        1.000000
75%        1.076923
max        1.307692
Name: WER, dtype: float64
```

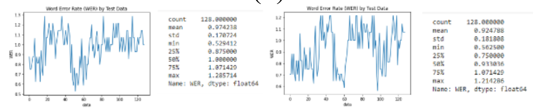
(a)



```
count    128.000000
mean       0.937621
std        0.165574
min        0.625000
25%        0.812500
50%        0.933333
75%        1.071429
max        1.307692
Name: WER, dtype: float64
```

```
count    128.000000
mean       0.938064
std        0.183570
min        0.529412
25%        0.764706
50%        0.939338
75%        1.071429
max        1.285714
Name: WER, dtype: float64
```

(b)



```
count    128.000000
mean       0.974238
std        0.170724
min        0.529412
25%        0.875000
50%        1.000000
75%        1.071429
max        1.285714
Name: WER, dtype: float64
```

```
count    128.000000
mean       0.924788
std        0.181808
min        0.562500
25%        0.750000
50%        0.933036
75%        1.071429
max        1.214286
Name: WER, dtype: float64
```

(c)

Figure 15.Word error rate performances of each proposed model (a) proposed spatio stream model (b) proposed two stream proposed model (c) proposed temporal stream model

We calculate WER of each training session for each proposed method by making prediction using the testing set from 10% of 1281 training data. We calculate the final WER by making prediction using dev set and test set as describe in Table 2.

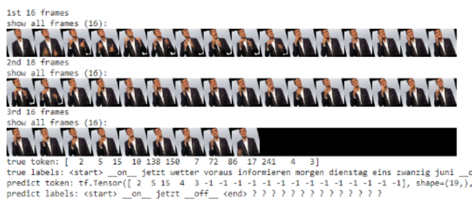
Table 3.Comparison of model evaluation results

Model	WER	
	DEV	TEST
Hybrid CNN HMM [13]	38.3	38.8
BLSTM+DetNet [14]	39.4	38.7
SubUNet [21]	43.1	42.1
Two Stream Local Global input + LS HAN [15]		38.3
Robust Hybrid CNN HMM [36]	39.4	38.7
Two Stream Spatio Temporal Input + RCNN [23]	23.10	22.86
Spatio Stream Input + C3D + BLSTM + CTC (DEV & TEST 100-125 frames)	106.8	94.1
Temporal Stream + C3D + BLSTM + CTC (DEV & TEST 100-125 frames)	105.5	105.9
Two stream + C3D + BLSTM + CTC (DEV & TEST 100-125 frames)	102.8	102.0
Spatio Stream Input + C3D + BLSTM + CTC (DEV & TEST full)	121.3	117.9
Temporal Stream Input + C3D + BLSTM + CTC (DEV & TEST full)	122.6	121.0
Two Stream Spatio Temporal Input + C3D + BLSTM + CTC (DEV & TEST full)	123.4	122.4
Spatio Stream Input + C3D + BLSTM + CTC (Full train data + DEV & TEST full)	121.0	123.2

Table 3 continued…

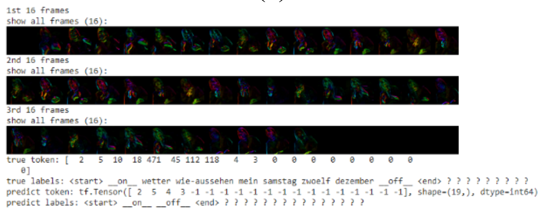
As can be seen in Figure 15 the proposed temporal model showed best performance out of the other proposed model. The WER is improving from 97% on the first training session to 92% on the second training session. The proposed two stream model perform better than the proposed spatio model. This result is quite different from the training performance as we can see in Figure 14. If we see the temporal stream in Figure 14 it has quite big gap between the training loss and validation score. It's quite surprising that the proposed temporal stream model alone is out perform the proposed two stream model stream model. We expect that the two stream model will show better result because as the frames shows in Figure 13 it process both the RGB frame sequence and Optical Flow frame sequence. Even though the result is far from perfect, these results show that using two stream input is a good method to do continuous sign language recognition. Furthermore, predictions are also made using the existing dataset in the DEV and TEST sets as described in Table 2

. The results of WER calculations carried out using this dataset become the final results which are then compared with research on CSLR with the 2014 RWTH-PHOENIX-Weather dataset conducted by previous researchers. In Table 3, it appears that the WER calculation results from previous researchers still exceed what has been achieved by the proposed method. The best results are still held by the two stream spatio temporal input + recurrent convolutional neural network model [23] with WER on DEV set of 23.10 and TEST set of 22.86. Among the three proposed models, the proposed two stream model gave the best WER result on DEV set with a value of 102.8 as shown in Table 3. However, the best WER result in the TEST set is to use the proposed spatio stream model with a value of 94.1 when compared with the other two proposed models as can be seen in Table 3.



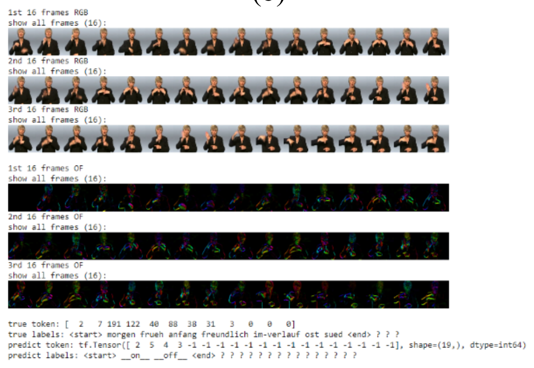
```
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [  2   5  15  10 138 150   7  72  86  17 241   4   3]
true labels: <start> __on__ jetzt wetter voraus informieren morgen dienstag eins zwanzig juni __off__ <end>
predict token: tf.Tensor([ 2  5 15  4  3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1], shape=(19,), dtype=int64)
predict labels: <start> __on__ jetzt __off__ <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```

(a)



```
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [  2   5  10  18 471  45 112 118   4   3   0   0   0   0   0   0   0   0
   0]
true labels: <start> __on__ wetter wie-aussehen mein samstag zwoelf dezember __off__ <end> ? ? ? ? ? ? ? ? ?
predict token: tf.Tensor([ 2  5  4  3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1], shape=(19,), dtype=int64)
predict labels: <start> __on__ __off__ <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```

(b)



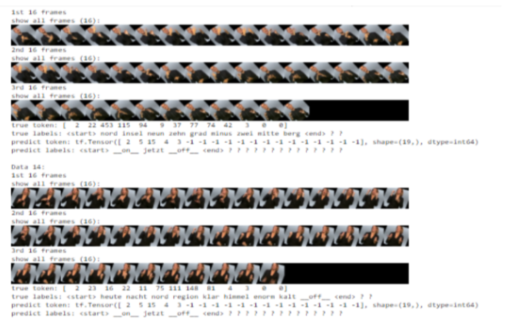
```
1st 16 frames RGB
show all frames (16):
2nd 16 frames RGB
show all frames (16):
3rd 16 frames RGB
show all frames (16):
1st 16 frames OF
show all frames (16):
2nd 16 frames OF
show all frames (16):
3rd 16 frames OF
show all frames (16):

true token: [  2   7 191 122  40  88  38  31   3   0   0   0]
true labels: <start> morgen frueh anfang freundlich im-verlauf ost sued <end> ? ? ?
predict token: tf.Tensor([ 2  5  4  3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1], shape=(19,), dtype=int64)
predict labels: <start> __on__ __off__ <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```

(c)

Figure 16.Example of predicted result (a) Spatio stream proposed model (b) Temporal stream proposed model (c) Two stream proposed model

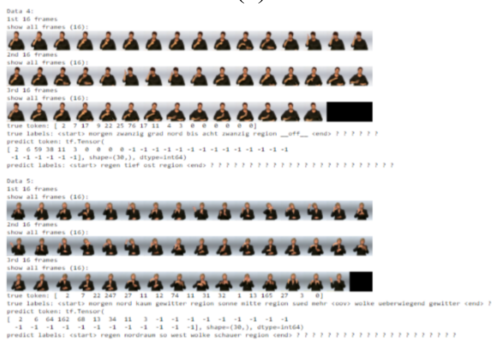
From Figure 16 it can be seen one of the prediction results of each of the proposed methods. The proposed spatio-temporal method provides predictive results of tokens and annotation labels that are closer to ground truth than the other proposed methods. The proposed spatio temporal method can predict up to five words as can be seen in Figure 16 (a). The other proposed method is only capable of guessing four words as can be seen in in Figure 16 (b) for the proposed temporal stream model and in Figure 16 (c) for the proposed two stream model. The proposed method seems to have been able to determine what words are the beginning and end of ground truth, namely "<start>" and "<end>" but still unable to predict other words properly as can be seen in Figure 16.

Furthermore, retraining is carried out using the complete TRAIN data set on the best model proposal from the previous training session, namely the spatio stream model. The training process was carried out using 5672 videos which were then predicted on the complete DEV data set containing 540 videos and the complete TEST set containing 629 videos. The training process using the complete TRAIN data set is divided into six training sessions with 20 epochs being held in each session. The final results of the predictions made can be seen in Table 3. The spatio stream model which was retrained using the complete TRAIN data set resulted in a WER with a value of 121.0 in the DEV data set and 123.2 in the TEST data set. As for how the results of the annotation predictions produced can be seen in Figure 17.



```
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [  2  22 453 115  94   9  37  77  74  42   3   0   0]
true labels: <start> nord insel neun zehn grad minus zwei mitte berg <end> ? ?
predict token: tf.Tensor([ 2  5 15  4  3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1], shape=(19,), dtype=int64)
predict labels: <start> __on__ jetzt __off__ <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ?

Data 14:
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [  2  23  16  22  11  75 111 148  81   4   3   0   0]
true labels: <start> heute nacht nord region klar himmel enorm kalt __off__ <end> ? ?
predict token: tf.Tensor([ 2  5 15  4  3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1], shape=(19,), dtype=int64)
predict labels: <start> __on__ jetzt __off__ <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```

(a)



```
Data 4:
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [ 2  7 17  9 22 25 76 17 11  4  3  0  0  0  0  0  0]
true labels: <start> morgen zwanzig grad nord bis acht zwanzig region __off__ <end> ? ? ? ? ? ?
predict token: tf.Tensor(
[ 2  6 59 38 11  3  0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1], shape=(30,), dtype=int64)
predict labels: <start> regen tief ost region <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?

Data 5:
1st 16 frames
show all frames (16):
2nd 16 frames
show all frames (16):
3rd 16 frames
show all frames (16):
true token: [  2   7  22 247  27  11  12  74  11  31  32   1  13 165  27   3   0]
true labels: <start> morgen nord kaum gewitter region sonne mitte region sued mehr <oov> wolke ueberwiegend gewitter <end> ?
predict token: tf.Tensor(
[  2   6  64 162  68  13  34  11   3  -1  -1  -1  -1  -1  -1  -1  -1  -1
  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1], shape=(30,), dtype=int64)
predict labels: <start> regen nordraum so west wolke schauer region <end> ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```

(b)

Figure 17.An example of the prediction results of a spatio stream model that was trained using (a) 100-125 frames train data set and (b) the complete train data set

Figure.16 (a) and Figure.16 (b) show a comparison between the predicted results of the spatio-temporal model with training using a train set containing videos with 100 – 125 frames and a complete train set. It can be seen in Figure 17 (a) that the results of the first and second video predictions produce the same results. Then it can be seen in Figure 17 (b) the results of the first and second video predictions produce different results. This shows that increasing the number of data sets can increase the vocabulary of word predictions made by the model. However, the prediction results have not shown optimal results due to the limitations of training sessions on the Google Colab platform which does not allow to train all data at once, so it is divided into six training sessions with each training consisting of 20 epochs.

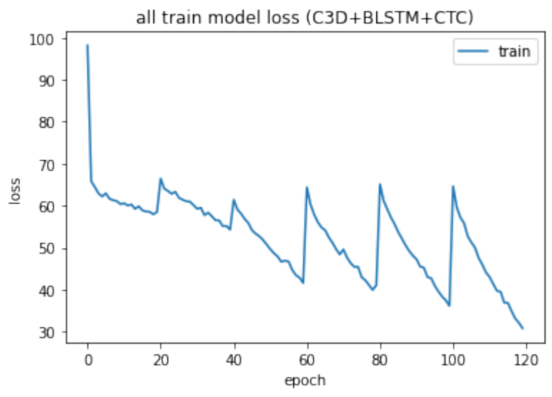


Figure 18.The results of the training using the spatio stream method with the complete train data set

Figure 18 shows the loss graph of the spatio stream model training using the complete TRAIN data set. The total epochs that were carried out for six training sessions were 120 epochs which were carried out within one week using the Google Colab platform. It can be seen in the graph that it has a sloping tendency even though there are interesting patterns that can be seen at every change of training session. The lowest score that appears in the training loss is 30,85275 which appears on the 120th epoch as the closing of the entire training session. In this complete training data train set, no random rotation data augmentation was performed at all.

CONCLUSION

In this study, we tried to propose a neural network model capable of recognizing continuous sign language (Continuous Sign Language Recognition / CSLR). The proposed model is a combination of a Three-Dimensional Convolutional Neural Network (3DCNN) with Long Short-Term Memory (LSTM) and Connectionist Temporal Classification (CTC). 3DCNN will extract video features in two stream input, a Spatio Stream with RGB Frames and a Temporal Stream with Optical Flow frames. The extracted features will be combined and directed to a sequence to sequence learning using LSTM and CTC to obtain a prediction of the ordered annotation following the gesture sequence displayed on the video input. The evaluation for the model that we propose will be done by calculating the Word Error Rate (WER) of the predicted annotations from the model. The model that we propose manages to get a Word Error Rate (WER) around 92%. These results indicate that the model that we propose is able to recognize sign language even though it's far from perfect.

The fact that our model is not perform well it might be caused by we did not use all the available dataset. To make it works in limited resource in Google Colab in the future research, we might adapt curriculum learning [34]. With curriculum learning we might feed the model with batches of dataset by continuing the training with last trained model. We could sort the dataset based on the frames amount from the smallest amount to the largest amount. Or we could sort the dataset based on the word amount from the smallest amount to the largest amount. And then we split into few batches and feed them to the model. We also could apply more recent 3DCNN model for our feature extraction model like I3D [35] in hopping to achieve better feature extraction result.

REFERENCES

[1] M. Tomasello, "The ultra-social animal," *Eur. J. Soc. Psychol.*, vol. 44, no. 3, pp. 187–194, 2014, doi: 10.1002/ejsp.2015.

[2] S. E. Jones and C. D. LeBaron, "Research on the Relationship between Verbal and Nonverbal Communication: Emerging Integrations," *J. Commun.*, vol. 52, no. 3, pp. 499–521, 2002, doi: 10.1111/j.1460-2466.2002.tb02559.x.

[3] M. Elmahgiubi, M. Ennajar, N. Drawil, and M. S. Elbuni, "Sign language translator and gesture recognition," *GSCIT 2015 - Glob. Summit Comput. Inf. Technol. - Proc.*, 2015, doi: 10.1109/GSCIT.2015.7353332.

[4] A. Kuznetsova, L. Leal-Taixe, and B. Rosenhahn, "Real-Time Sign Language Recognition Using a Consumer Depth Camera," *2013 IEEE Int. Conf. Comput. Vis. Work.*, pp. 83–90, 2013, doi: 10.1109/ICCVW.2013.18.

[5] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 1, pp. 131–153, 2019, doi: 10.1007/s13042-017-0705-5.

[6] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[7] A. Munappy, J. Bosch, H. H. Olsson, A. Arpteg, and B. Brinne, "Data Management Challenges for Deep Learning," *Proc. - 45th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2019*, pp. 140–147, 2019, doi: 10.1109/SEAA.2019.00030.

[8] N. Aloysius and M. Geetha, "Understanding vision-based continuous sign language recognition," *Multimed. Tools Appl.*, vol. 79, no. 31–32, pp. 22177–22209, 2020, doi: 10.1007/s11042-020-08961-z.

[9] L. Yusnita, R. Roestam, and R. B. Wahyu, "Implementation of Real-Time Static Hand," *CommIT (Communication & Information Technology)*, vol. 11, no. 2. pp. 85–91, 2017.

[10] R. Hartanto, A. Susanto, and P. I. Santosa, "Real time static hand gesture recognition system prototype for Indonesian sign language," in *Proceedings - 2014 6th International Conference on Information Technology and Electrical Engineering: Leveraging Research and Technology Through University-Industry Collaboration, ICITEE 2014*, 2014, no. January 2015, doi: 10.1109/ICITEED.2014.7007911.

[11] D. Guo, W. Zhou, H. Li, and M. Wang, "Online early-late fusion based on adaptive HMM for sign language recognition," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 14, no. 1, pp. 1–18, 2017, doi: 10.1145/3152121.

[12] Suharjito, H. Gunawan, N. Thiracitta, and A. Nugroho, "Sign Language Recognition Using Modified Convolutional Neural Network Model," *1st 2018 Indones. Assoc. Pattern Recognit. Int. Conf. Ina. 2018 - Proc.*, no. July 2019, pp. 1–5, 2019, doi: 10.1109/INAPR.2018.8627014.

[13] O. Koller, S. Zargaran, H. Ney, and R. Bowden, "Deep sign: Hybrid CNN-HMM for continuous sign language recognition," in *British Machine Vision Conference 2016, BMVC 2016*, 2016, vol. 2016-Septe, pp. 136.1-136.12, doi: 10.5244/C.30.136.

[14] R. Cui, H. Liu, and C. Zhang, "Recurrent convolutional neural networks for continuous sign language recognition by staged optimization," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1610–1618, 2017, doi: 10.1109/CVPR.2017.175.

[15] J. Huang, W. Zhou, Q. Zhang, H. Li, and W. Li, "Video-based sign language recognition without temporal segmentation," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 2257–2264, 2018.

[16] J. Singha and K. Das, "Indian Sign Language Recognition Using Eigen Value Weighted Euclidean Distance Based Classification Technique," *arXiv Prepr. arXiv1303.0634*, vol. 4, no. 2, pp. 188–195, 2013, [Online]. Available: http://arxiv.org/abs/1303.0634.

[17] J. Forster, C. Schmidt, O. Koller, M. Bellgardt, and H. Ney, "Extensions of the sign language recognition and

translation corpus RWTH-PHOENIX-Weather," *Proc. 9th Int. Conf. Lang. Resour. Eval. Lr. 2014*, no. May, pp. 1911–1916, 2014.

[18] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 4489–4497, 2015, doi: 10.1109/ICCV.2015.510.

[19] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," *2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL HLT 2016 - Proc. Conf.*, pp. 1480–1489, 2016, doi: 10.18653/v1/n16-1174.

[20] Q. Zhang, G. Hua, W. Liu, Z. Liu, and Z. Zhang, "Auxiliary training information assisted visual recognition," *IPSJ Trans. Comput. Vis. Appl.*, vol. 7, pp. 138–150, 2015, doi: 10.2197/ipsjtcva.7.138.

[21] N. C. Camgoz, S. Hadfield, O. Koller, and R. Bowden, "SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 3075–3084, 2017, doi: 10.1109/ICCV.2017.332.

[22] A. Krizhevsky, I. Sutskever, and G. E, Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012, [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[23] R. Cui, H. Liu, and C. Zhang, "A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training," *IEEE Trans. Multimed.*, vol. 21, no. 7, pp. 1880–1891, 2019, doi: 10.1109/TMM.2018.2889563.

[24] O. Koller, J. Forster, and H. Ney, "Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers," *Comput. Vis. Image Underst.*, vol. 141, pp. 108–125, 2015, doi: 10.1016/j.cviu.2015.09.013.

[25] L. Pigou, A. van den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, "Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video," *Int. J. Comput. Vis.*, vol. 126, no. 2–4, pp. 430–439, 2018, doi: 10.1007/s11263-016-0957-7.

[26] V. Athitsos *et al.*, "The American Sign Language Lexicon Video Dataset," *2008 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. CVPR Work.*, 2008, doi: 10.1109/CVPRW.2008.4563181.

[27] M. Zikky and Z. F. Akbar, "Kamus Sistem Isyarat Bahasa Indonesia (KASIBI) dengan Voice Recognition sebagai Pendukung Belajar Bahasa Isyarat Berbasis Android," *JST (Jurnal Sains Ter.*, vol. 5, no. 2, 2019, doi: 10.32487/jst.v5i2.732.

[28] F. Ronchetti, F. Quiroga, and L. Lanzarini, "LSA64 : An Argentinian Sign Language Dataset," *Congr. Argentino Ciencias la Comput.*, pp. 794–803, 2016.

[29] R. Prabhavalkar *et al.*, "Minimum Word Error Rate Training for Attention-Based Sequence-to-Sequence Models," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2018-April, no. Cd, pp. 4839–4843, 2018, doi: 10.1109/ICASSP.2018.8461809.

[30] J. Ortiz Laguna, A. G. Olaya, and D. Borrajo, "A dynamic sliding window approach for activity recognition," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6787 LNCS, pp. 219–230, 2011, doi: 10.1007/978-3-642-22362-4_19.

[31] Y. Ye, Y. Tian, M. Huenerfauth, and J. Liu, "Recognizing american sign language gestures from within continuous videos," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2018-June, pp. 2145–2154, 2018, doi: 10.1109/CVPRW.2018.00280.

[32] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo, "TV-L1 Optical Flow Estimation," *Image Process. Line*, vol. 3, pp. 137–150, 2013, doi: 10.5201/ipol.2013.26.

[33] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof, "A duality based algorithm for TV-L1-optical-flow

image registration," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4792 LNCS, no. PART 2, pp. 511–518, 2007, doi: 10.1007/978-3-540-75759-7_62.

[34] G. Hacohen and D. Weinshall, "On the power of curriculum learning in training deep networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 4483–4496, 2019.

[35] J. Carreira and A. Zisserman, "Quo Vadis, action recognition? A new model and the kinetics dataset," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 4724–4733, 2017, doi: 10.1109/CVPR.2017.502.

[36] O. Koller, S. Zargaran, H. Ney, and R. Bowden, "Deep Sign: Enabling Robust Statistical Continuous Sign Language Recognition via Hybrid CNN-HMMs," *Int. J. Comput. Vis.*, vol. 126, no. 12, pp. 1311–1325, 2018, doi: 10.1007/s11263-018-1121-3.