Sign Language Recognition

Inflated 3D ConvNet

Nguyen To

Master's Thesis

Faculty of Science and Forestry
School of Computer Science

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu School of Computing
School of Computer Science

Student, Nguyen To : Sign Language Recognition
Master's Thesis , 42 p.
Supervisors of the Master's Thesis : Profressor Xiao-Zhi Gao
August 2021

Abstract:

Effective communication is considered as the foremost fundamental of human skills. However, more than 5% of the world's population is suffering from disabling hearing loss, as indicated by the World Health Organization (WHO). There is hence a communication gap between the hearing-impaired community, whose primary means of communication is sign language, and others who are not privy with this language. To this end, Sign Language Recognition could be an essential instrument which utilizing vision-based technology and helps the hearing-impaired communicate with the society with ease, thereby diminishing the verbal exchange barrier. The first step in interpreting and analyzing communication via gestures is word-level sign language recognition (WSLR). Recognizing signs from recordings maybe a tough challenge due to the fact that the meaning of a word is determined by a combination of subtle body movements, hand gestures, and other actions. Be that as it may, with the significant advancement of technology, notably Convolutional Neural Network (CNN), in this paper, an Inflated 3D Networks (I3D), a 3D video categorization solution are used to method as an answer of WSLR.

Keywords: Sign Language, Continuous Sign Language, Sign Language Recognition, Classification, Vidieo Classification, Recognition, Convolutional Neural Network, 3D Convolutional Neural Network

CR Categories (ACM Computing Classification System, 1998 version): A.m, K.3.2

Preface

The basis for the project initially originated from my ardor for developing better methods of sign language recognition. The hearing-impaired community's life has changed considerably over the past half-century as a result of policy adjustments and latest technological tendencies, yet the road to find a viable answer for word-level sign language recognition (WSLR) keeps on being a drawn out circumstance. It is my passion to not solely find out, however to develop tools to bridge the communication gap between the deaf community and the society. This project follows the reference and citation guidelines of the "Quo Valdis, Action Recognition? A New Model and the Kinetics Datasets" by a group of João Carreira and Andrew Zisserman.

In truth, I could not have achieved my current level of success without a strong support group. To begin with, I wish to express my sincere thanks to my supervisor, Profressor Xiao-Zhi Gao, for his excellent guidance, valuable input and support throughout the entire period. Furthermore, I would also like to thank Li Dongxu for his enormously valued assistance in collecting data for this study. Especially with respect Cong Phan, a Phd candidate at Griffith University who gave a great help by offering several useful insights and recommendations. And finally, I am grateful to Mai Khanh Nguyen Ngoc. She stood by my side and provided me with the support I needed to complete this thesis.

List of Abbreviations

ACM	Association for Computing Machinery
ISY	Itä-Suomen yliopisto
UEF	University of Eastern Finland
WSLR	Word-level Sign Language Recognition
I3D	Two -Stream Inflated 3D Convolutional Network
CNN	Convolutional Neural Network
LSTM	Long-short Term Memory
TGCN	Temporal Graph Convolutional Network

Contents

List of Tables

List of Figures

List of Equations

1 Introduction

Sign Language, any methods of communicating by bodily motions, particularly with hands and arms, that is utilized when verbal communication is either difficult or undesirable. Sign language can consist of a series of overly-exaggerated facial expressions, shrugs, or hand gestures; or it can be a fine and delicate mix of hand signals that are complemented by facial expressions and words spelt out using a manual alphabet. When a deaf person or someone speaking a different language is communicating with someone who is hearing, using sign language can help connect the parties. (Britannica, 2020, November 12). The public has neither the time nor the patience to learn sign language, which is complicated and time-consuming to learn and practice. Additionally, there are also many language and culture-specific (Holtz, 2014) (e.g Germany, Japanese) constraints which will hinder the widespread adoption of sign language. Significant advances in deep learning (DL) and improvements in device capabilities, such as computation power, memory capacity, power usage, sensor resolution, and optics, have improved the performance and cost-effectiveness of vision-based applications, allowing them to spread more quickly in the market place. For this reason, it is interesting to examine sign language recognition (SLR), which automatically translates sign language and aids deaf-mute individuals in communicating with others in their life

Back to the history of 90s, Yann LeCun et al. published "Gradient-Based Learning Applied to Document Recognition", which is widely considered to be the most popular AI article from the era. This paper was the first modern application of convolutional neural networks to be developed. Since then, more and more sophisticated models trained on ever-larger datasets have been built using the convenient approach of convolutional neural networks. Especially in the field of Computer Vision - Human-based activity recognition, there are many methods can be applied to solve the problem, from traditional convolutional neural networks such as CNN-RNN, CNN-LSTM to RestNetCRNN, Conv3D and state-of-the-art networks e.g Pose-TGCN, I3D. Inheriting the idea of using two-stream I3D network, which is based 2D ConvNet (Carreira & Zisserman, 2017), presented by Carreira and Zisserman, this project is re-implement the model with a slightly modification inside. It might not be better when comparing with other models, however during the project, I have got many experience and broaden my

knowledge on the field of Deep Learning.

1.1 Problem

As same as other human-based activity recognition, SRL also shares some common problems such as background clutter, lightning or lightning changing in a video, motion blur, angle of camera, changing scale. SRL, on the other hand, is a more difficult task than ordinary action recognition. Firstly, sign language relies on a combination of global body movement and subtle hand/arm gesture. Additionally, depending on how many times they are repeated, same gestures might have different meanings. SRL might be more difficult to examine because of different states of motions and signers such as localism, gesture speed, preferred hand or physical form. Finally, it is also expensive to collect additional data from many signers even though it is desirable (Jiang et al., 2021).

As described above, the datasets that uses for training SLR are limited, even the number of samples inside each dataset. The table below describes some datasets that normally use for researching.

Table 1: Sign Language Datasets.

Dataset	**Language**	**Classes**	**Samples**	**Data Type**	**Language Level**
CSL Dataset I	Chinese	500	125,000	Video & Depth from Kinect	Isolated
CSL Dataset II	Chinese	100	25,000	Videos & Depth from Kinect	Continuous
RWTH-PHOENIX-Weather 2014	German	1,081	6,841	Videos	Continuous
RWTH-PHOENIX-Weather 2014 T	German	1,066	8,257	Videos	Continuous
ASLLVD	American	3,300	9,800	Videos(multiple angles)	Isolated
ASLLVD-Skeleton	American	3,300	9,800	Skeleton	Isolated
SIGNUM	German	450	33,210	Videos	Continuous
DGS Kinect 40	German	40	3,000	Videos(multiple angles)	Isolated
DEVISIGN-G	Chinese	36	432	Videos	Isolated
DEVISIGN-D	Chinese	500	6,000	Videos	Isolated
DEVISIGN-L	Chinese	2000	24,000	Videos	Isolated
LSA64	Argentinian	64	3,200	Videos	Isolated
GSL isol.	Greek	310	40,785	Videos & Depth from RealSense	Isolated
GSL SD	Greek	310	10,290	Videos & Depth from RealSense	Continuous
GSL SI	Greek	310	10,290	Videos & Depth from RealSense	Continuous
IIITA -ROBITA	Indian	23	605	Videos	Isolated
PSL Kinect	Polish	30	300	Videos & Depth from Kinect	Isolated
PSL ToF	Polish	84	1,680	Videos & Depth from ToF camera	Isolated
BUHMAP-DB	Turkish	8	440	Videos	Isolated
LSE-Sign	Spanish	2,400	2,400	Videos	Isolated
Purdue RVL-SLLL	American	39	546	Videos	Isolated
RWTH-BOSTON-50	American	50	483	Videos(multiple angles)	Isolated
RWTH-BOSTON-104	American	104	201	Videos(multiple angles)	Continuous
RWTH-BOSTON-400	American	400	843	Videos	Continuous
WLASL	American	2,000	21,083	Videos	Isolated

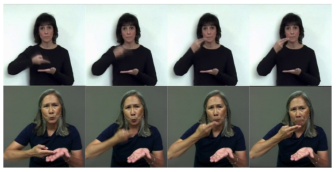
Time segmentation is another issue for SLR as it is difficult to distinguish different kinds of sign language while signers make gestures continuously to describe a

phrase or a sentence (Xiao et al., 2020). Word-level sign recognition, an integral part of comprehending sign language phrases and sentences, is also extremely difficult task itself:

- The meaning of signals is primarily determined by the mix of bhand actions, body motions and head positions, and small variations in these elements can result in a variety of interpretations.
- With the same gesture, depend on the natural languages and context, they might have different meaning. It is also possible for nouns and verbs from the same lemma to share the same sign. These nuances are not effectively reflected by the small-scale datasets that are currently available (Figure 1) (D. Li et al., 2020).
- The number of signs that are used on daily basis is enormous, it could be thousands. In comparison, tasks such as gesture and action recognition have just had a few hundred categories. The scalability of recognition algorithms is significantly hampered as a result.

(a) **"Scream"** is performed slightly different by different signers

(b) The verb **"Wish"** (top) and the adjective **"Hungry"** (bottom) have same gesture



(c) **"Rice"** (top) and **"Soup"** (bottom) have same gesture

Figure 1: Common occurrence of ambiguity and variation signing

1.2 Thesis scope and target

In fact, the researching of SLR, including isolated (word-level) and continuous (phrase or sentences), has yielded significant results and developed rapidly in recent years (Lim et al., 2019; Y. Li et al., 2018; Mercanoglu Sincan et al., 2021). So this project aims to understand the architecture of I3D model, focuses on the methodology of training, evaluating the network with given dataset. A further objective of the project is getting familiar with libraries and frameworks of Deep

Learning which is built for Python environment, and also other related topics such as image processing, data labelling, development tools e.g Google Colab, Anaconda, Jupyter Notebook. Regarding data and dataset, the project gave a comprehensive overview of organizing data, loading, splitting data for training and testing purpose.

Finally, in terms of the topic's aim, this thesis primarily focuses on enhancing the accuracy of the dataset with the two-stream I3D network. Because of the limitation of facilities, such as hardware, and time-consuming of training process, the project only used the WLASL and ASL dataset. In order to reduce the consumption of training time, as well as to test the effectiveness of I3D network, the dataset is divided into many sub-datasets which have 100 classes, 300 classes, 1000 classes and 2000 classes which is applied for WLASL. Regarding ASL dataset, because the number of class is small (64 words) so this is not splitted.

2 Model

2.1 Methodology of video classification and CNNs

A video is a collection of multi images which are represented by sequential. Therefore, the main idea of clarifying an action or object inside a video is analyzing frame by frame. In details, the general procedure of video recognition (Sivic & Zisserman, 2003; Niebles et al., 2010) contains three key stages. The video is first divided into regions (Liu et al., 2009) based on the places that are easy to characterize in visual terms. This is done by either sampling regions densely (Wang & Schmid, 2013) or, if there are few areas of interest (Laptev, 2005), by using a sparse sampling technique. In the next step, the characteristics are merged into a video level description with defined size. One common method is to train a K-means dictionary and then evaluate all the features. This allows to gather visual words during the duration of the video and then arrange them into histograms of various spatio-temporal positions and extents (Karpathy et al., 2014; Laptev et al., 2008). Finally, a classifier (such as an SVM) is trained to discriminate between the classes of interest with the resulting "bag of words.".

CNN, a single neural network which emulates the process of human brain (LeCun

et al., 1998) offer an approach that combines 3 phases into an end-to-end training from the raw value of pixels to the output classifiers. By using restricted connection across layers (local filters), parameter sharing (convolutions), and specific local invariance-building neurons (max pooling), the spatial structure of pictures is specifically exploited. Recently with the outstanding of GPU, CNNs can now scale the networks with millions variables, resulting in significant developments in object recognition (Girshick et al., 2014), image classification (Krizhevsky et al., 2017), scene annotation. The use of CNNs has piqued the interest of many in the computer vision research community (Zha et al., 2015), and it has been demonstrated that CNN-based methods can reach state-of-the-art performance on various of complex image datasets (Sharif Razavian et al., 2014).

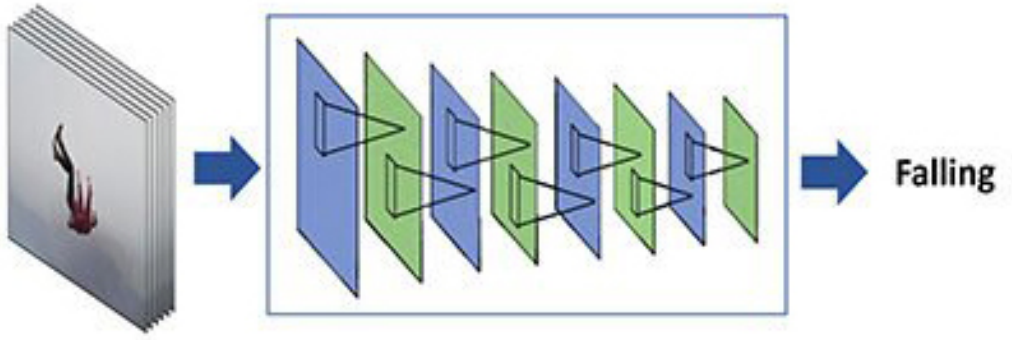


Figure 2: An example of video classification

2.2 Some models to solve video recognition problems

2.2.1 ConvNet-LSTM

LSTM network and CNNs have been researched widely but independently previously. While CNNs are able to give spatially specific information, LSTM networks are good at producing temporally comprehensive results (Mutegeki & Han, 2020). As a result, the use of CNN-LSTM networks is extensively employed for time series data, and is particularly useful for video datasets that have a time dimension.

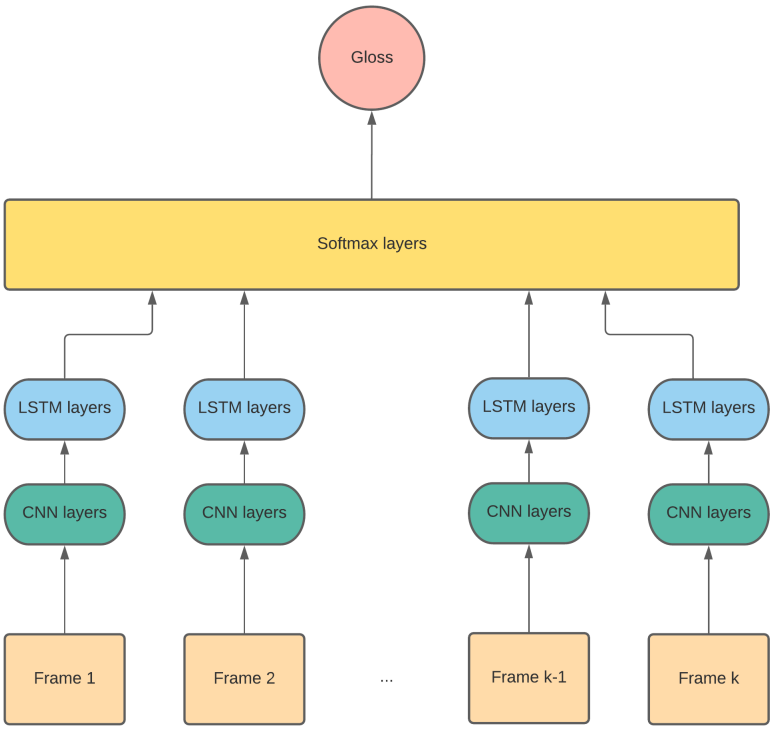


Figure 3: CNN-LSTM model

In the model of ConvNet-LSTM, features are extracted separately from each frame of a video through a CNN network (Karpathy et al., 2014). Then they are feeded into a LSTM layer (Yue-Hei Ng et al., 2015; Donahue et al., 2015), which is capable of providing stage encoding and temporal ordering. Finally, a fully connected layer is put on the top for classifer.

2.2.2 3D ConvNet

Extended from 2D ConvNet, in which convolutions exclusively generate features based on spatial dimensions, 3D ConvNet computes features in the spatial as well as temporal dimensions. Convolution using a 3D kernel is done by convolving a cube of several frames to get a 3D result. This architecture has the additional advantage of making the feature maps in the convolution layer linked to several contiguous frames in the preceding layer, which aids in the capture of motion information (Ji et al., 2013). The use of 3dConvNet have been exploited in many researchs (Tran et al., 2015; Taylor et al., 2010).

With $c \times l \times h \times w$ size of a video (c is a number of channel, l is a number of frames, h and w are corresponding to height and width of each frame), kernel size for 3D convolution and pooling are denoted by the notation $d \times k \times k$, where d is the kernel temporal depth and k is kernel spatial size, 3D ConvNet networks are programmed to accept video clips as inputs and predict the class label that correlate to n classes based on datasets (Tran et al., 2015).

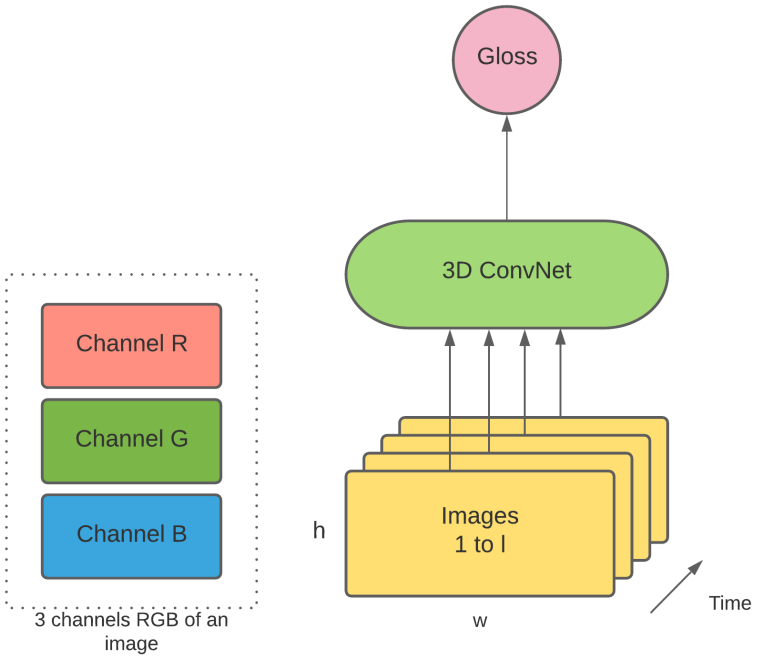


Figure 4: An example of 3D ConvNet with RGB video as input

2.2.3 Two-stream network

Introduced by Simonyan and Zisserman et al, a model which uses an individual RGB frame and an external n-frame optical flow, was shown extremely high performance on benchmarks, while also being efficient to train and test (Simonyan & Zisserman, 2014). In details, a video data is separeted into two streams, spatial stream and temporal stream. The spatial stream can spot motion in static frames, whereas the temporal stream is trained to perceive motion in images as optical flows, both of them are developed with ConvNet (Simonyan & Zisserman, 2014).

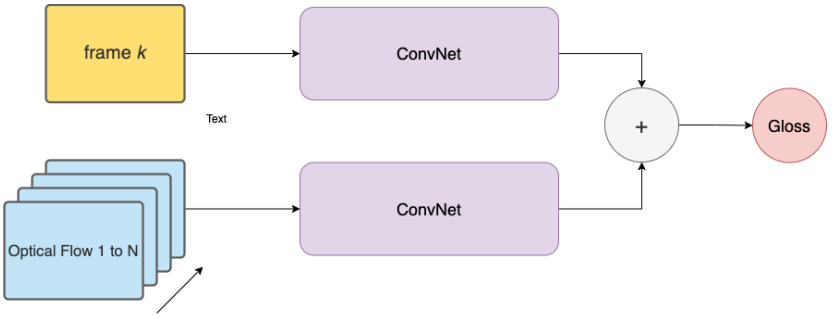


Figure 5: Two-stream 3D ConvNet model

The result shown by Simonyan and Zisserman indicates that using optical flow for training a temporal dimension is outperform than training on static frames (Karpathy et al., 2014).

An extended version of two-stream network, 3D-Fused Two-Stream, is using a 3D ConvNet which can learn patterns related to temporal directly (Carreira & Zisserman, 2017). The model of 3d-Fused Two-Stream is as Figure 6.

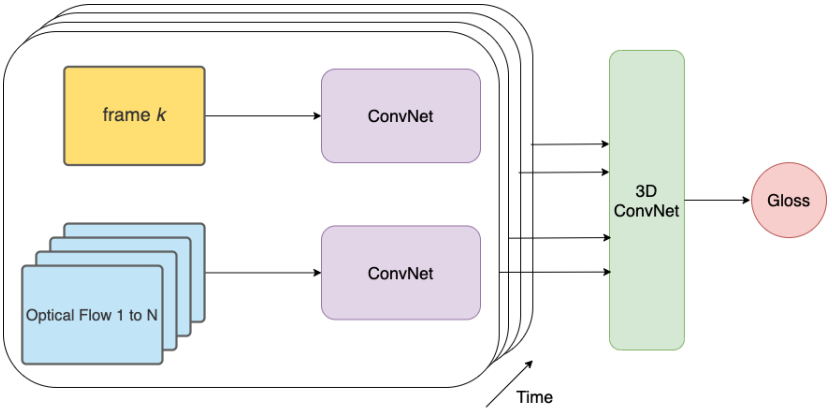


Figure 6: 3D-Fused Two-Stream

2.3 Model in this study

With the extendable 2D ConvNet to 3D ConvNet where time dimensions can be added to perform the result, a coming up model is that use two streams of raw images and also optical flow frames. Thus, the temporal patterns is not only studied during the optical flow but also with RGB inputs. As a result, the performance is improved significantly compared with using only RGB stream (Carreira & Zisserman, 2017).

In the method that Zisserman and Carreira proposed, 2 streams are trained independently, Therefore, with an input video, each stream will provide a prediction according to RGB frames and optical flow frames of it which are fed into two different I3D networks. Then next step is that the predictions of RGB stream and Optical Flow stream are computed averaged to give a final prediction.

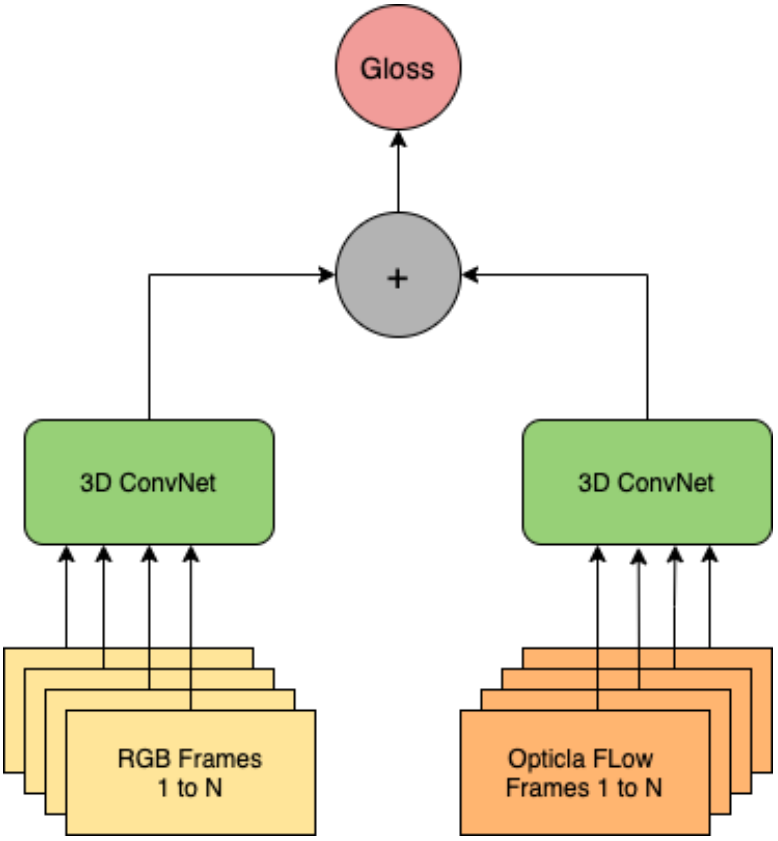


Figure 7: 2-stream 3D ConvNet by Zisserman and Carreira (Carreira & Zisserman, 2017)

The result of combining the predictions from RGB stream and optical flow stream is better that the prediction of each stream separately and also better than the results from other models (Carreira & Zisserman, 2017). In the study, the group of author examined on several datasets for human recognition, such as UCF-101, HMDB-51, Kinitecs, the comparision if shown as the table below.

Architecture	UCF-101			HMDB-51			Kinetics		
	RGB	Flow	RGB + Flow	RGB	Flow	RGB + Flow	RGB	Flow	RGB + Flow
LSTM	81.1	-	-	36.0	-	-	63.3	-	-
3D ConvNet	51.6	-	-	24.3	-	-	56.1	-	-
2-Stream	83.6	85.6	91.2	43.2	56.3	58.3	62.2	52.4	65.6
3D-Fused	83.2	85.8	89.3	49.2	55.5	56.8	-	-	67.2
2-Stream I3D	84.5	90.6	93.4	49.8	61.9	66.4	71.1	63.4	74.2

Table 2: Comparision betwen models on several human activity datasets (Carreira & Zisserman, 2017)

With an amazing result compared to others, this study use the architecture of 2-stream I3D network to see how good it is when apply to recognize sign languages instead of human activies.

3 Convolution Neural Network in image classification

3.1 Introduction of CNNs

The heart of Deep Learning algorithms rely on Artificial Neural Networks (ANNs) what immitate the behaviors of human braind in which signals are send between neurons (Education, n.d.-b). Various neural network types are employed for certain use cases and data types and CNNs is a specific netwwork which are primarily focus to solve the problem of classification and computer vision tasks. Before the use of CNNs, to identify an object in images, feature extraction approaches were applied, however it were time-consuming and manual methods. With the introduction of convolutional neural networks, a more scalable technique to image classification and object identification has been made available. This methodology, which leverages linear algebra concepts, relies on matrix multiplication to find patterns inside an image. Most of CNNs have expensive computation, therefore to train model effectively, graphical processing units (GPUs) are required.

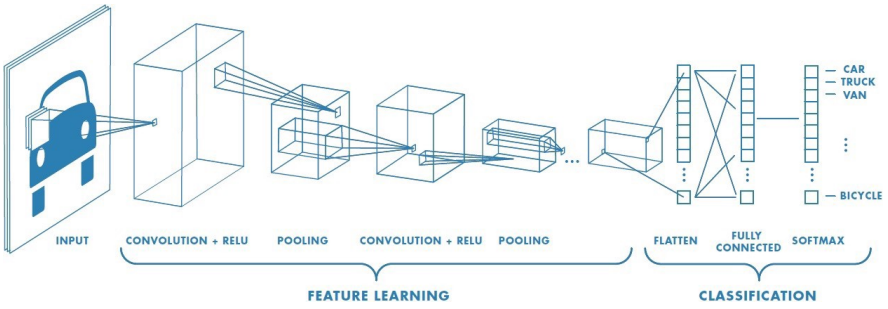
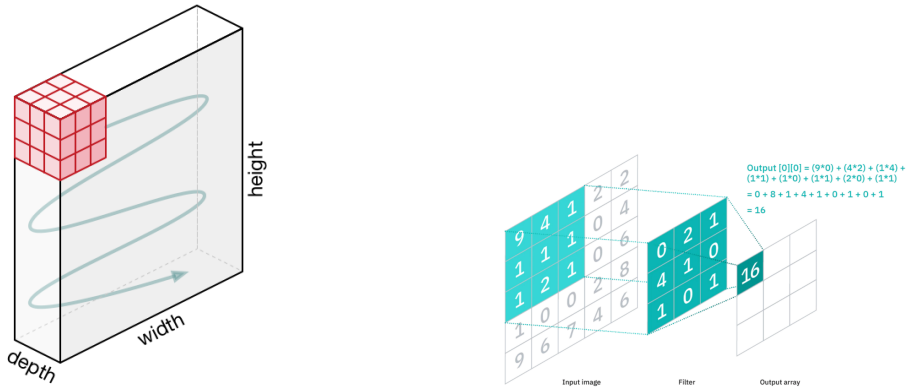


Figure 8: Example of CNN for classification

A key difference between convolutional neural networks and other neural networks is their strength ability to handle voice, audio or image inputs. In the architecture of CNNs, there are 3 primary types of layers: convolutional layers, pooling layers and fully connected layers (Education, n.d.-a).

3.1.1 Convolution layer - the kernel

The convolutional layer is the fundamental building element of a CNN, where most of computation occurs. The main component of convolution layers are input data, a filter and feature mapping. For instance, an image can be consider as a matrix of $width \times high \times channel$, a video is matrix of $width \times high \times channel \times times$. Then a kernel or filter, named as feature detector, will evaluate for the presence of a specific feature by moving across the fields of images. The final output of this processing is called as an activation map, feature map or convoluted feature (Dumoulin & Visin, 2016).



(a) Kernel Movement

(b) Convolution Processing

Figure 9: Kernel movement and Convolution Processing

A kernel is a matrix of weights and the dimensions of the kernel is as same as the dimensions of input parameter of a convolutional layer, for example, with 2D convolutions, 2D matrix is used as kernel matrix. On the other hand, the filter is set of kernels, and each kernel is applied to a specific channel. Therefore, a filter have one more dimension compared to the dimension of kernel. If size of kernel is $h \times w$, and with k channels, filter's dimension is $k \times h \times w$ (Albawi et al., 2017a). Through the convolution operation, the high-level feature such as edges are extracted. Convolutional layer is not restricted to only one. Normally, the first layer of convulutional captures low-level features such as color, gradient orientation, edges. Adding more layers allows the architecture to be able to adapt high-level characteristics which gives us a network that fully understanding image in the dataset.

However, there are problems with convolutions to be considered. As the example in 9b, with the 5×5 matrix input, after applying the filter, the size of output matrix is 3×3. Thus, the image size is shrinked every convolution operation. Because of multi convolutional layers in the network, the original image gets smaller; however it is not expected to be shrinked every time (Szegedy et al., 2015). Another matter is that every time the kernel moves accross the images, the edges of objects inside the picture have less number of touches than the middle area of the objects, and the middle areas are also overlapped. So the output does not reflect much about the information of corner features and edges in an image. Padding is a solution to tackle these issues by keep the size of the images preserve.

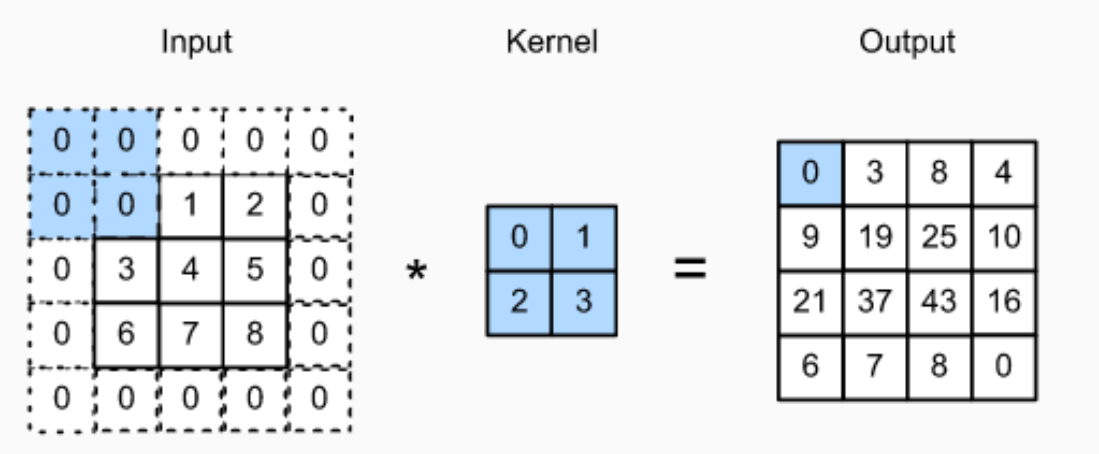


Figure 10: Example of padding

So if the input matrix has size $n \times n$, filter matrix is $f \times f$ with the padding p, then the matrix output will have the size of $(n + 2p - f + 1) \times (n + 2p - f + 1)$. In the exxample in 10, p is 1 while orginal input size is 3×3 and 2×2 is the size of filter matrix.

Another hyperparameter related to convolution neural networks is stride. As definition, stride is number of pixels that filer matrix moves each step across the horizontal or vertical position. The size of convolution output is depened on padding and stride hyperparameters. With padding p, $f \times f$ filter matrix, $n \times n$ input matrix, and stride s, the output dimension will be $[(n + 2p - f + 1)/s + 1] \times [(n + 2p - f + 1)/s + 1]$

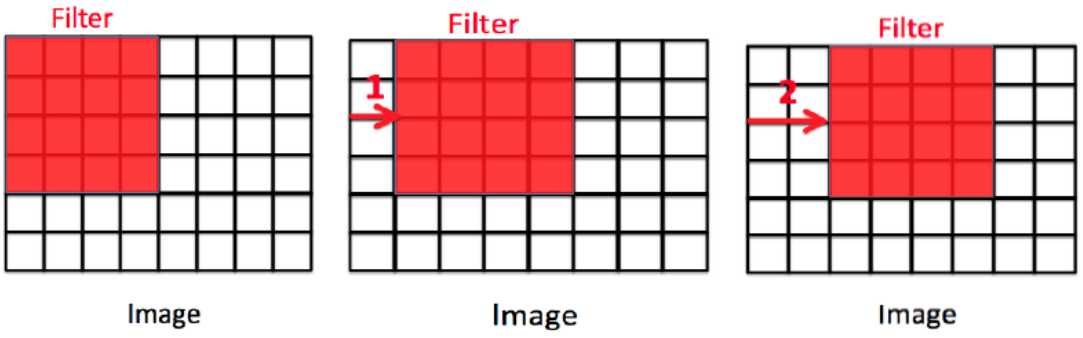


Figure 11: Example of stride = 0, 1, 2 respectively.

3.1.2 Pooling layer

Besides convolution computations, pooling operations is responsible to build up essential blocks in CNNs by decreasing the spatial size of convovled feature, or reducing the number of parameter input (Dumoulin & Visin, 2016). By this, computational power required also decrease because of dimensionality reduction. Additionally, it is beneficial for extracting dominant features which are positional and rotational invariant, thus allows the model to properly trained. The pooling operation works similarly as the convolutional layer in that it sweeps a filter across the input, except that the filter does not contain any weights (Albawi et al., 2017a). There are two types of pooling (Education, n.d.-a):

- Max Pooling: with each pass of the filter over the input, the highest value is selected to the output array. This technique is more commonly implemented

compared with avarage pooling.

- Average Pooling: instead of choosing a max value of input, tt computes the average value inside the receptive field, then send to output array.

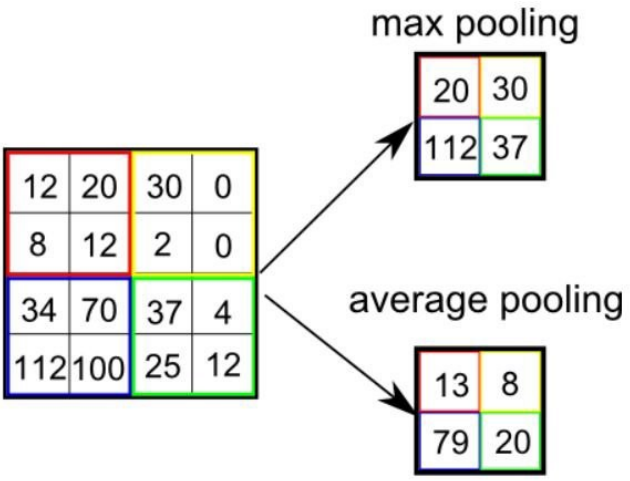


Figure 12: Pooling

The pooling layer has a few drawbacks, but is also advantageous to the CNN. They simplify, optimize, and minimize the danger of overfitting.

3.1.3 Fully Connected (FC) Layers - Classification

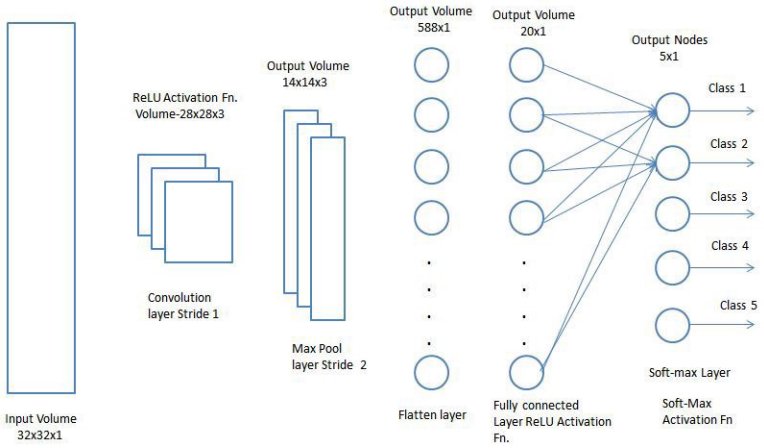


Figure 13: Fully Connected layer (Saha, n.d.)

It is typically inexpensive to add a Fully Connected Layer after convolutional layers to learn non-linear combinations of the features represented as output.

The name of the full-connected layer accurately reflects its characteristics. A good description of the layer is that each node in the output layer connects to all nodes of the previous layer (Albawi et al., 2017b). Using the characteristics gathered from the preceding layers and their various filters, this layer conducts the duty of categorization on the data. Contrary to convolutional and pooling layers, which often employ ReLu functions to categorize inputs properly, FC layers typically employ a softmax activation function to give results in a probability ranging from 0 to 1 (Education, n.d.-a)

3.2 1D, 2D, 3D Convolutions

When it comes to time series data processing, 1D Convolutions are widely used because the input in such cases is one dimension. Mentioned previously, multi channels can stack with the 1D data input. As the filter can travel in one direction only, then the output only have 1D.

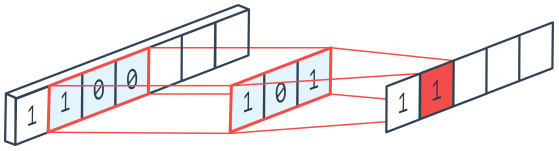
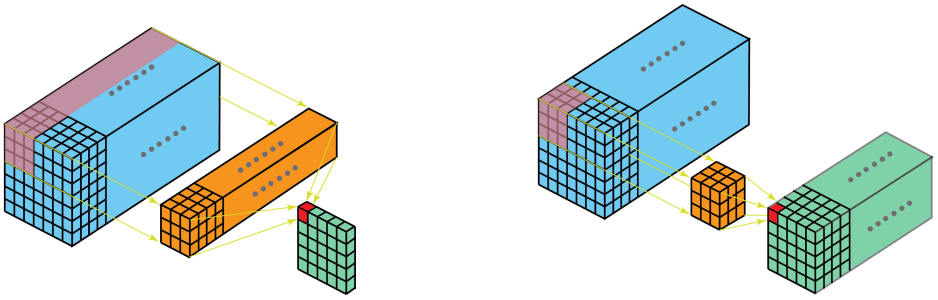


Figure 14: 1D Convolution

The diagram below visualize how 2D convolutions is. As mentioned before, 2D data also have multiple channels, so the filter can move in 2 directions and lead to the final 2D output. In computer vison, 2D convolutions are the most common convolutions. In case of 3D convolutions, it is hard to visualize the filter because of 4 dimensions the filter has, so only the 3D convolutions with single channels is visualized as the figure below. Like others, in 3D convolutions, the filter can move in 3 directions therefore the output is 3 dimentions.



(a) 2D Convolutions (b) 3D Convolutions with single channel

Figure 15: 2D and 3D Convolutions

4 Techniques for Inflated 3D ConvNet

4.1 Inflating 2D ConvNets into 3D ConvNets

In order to prevent spending to much time in the process of training spatio-temporal models, a simply approach is applied to utilize the successful of 2D (image) classification architectures into 3D ConvNet when 2D classified models have been achieved many significant improvements in years. As a result, the filters and pooling kernels that are coming from the 2D architectures are inflated with one more dimension, temporal dimension. So that they are from a $N \times N$ matrix to a $N \times N \times N$ cubic (Carreira & Zisserman, 2017).

4.2 2D filters to 3D filters

In additional, the input parameters can also be boostrapted by adopting pre-trained ImageNet models. As can be seen, a video (boring video (Carreira & Zisserman, 2017)) can be made from an individual image by cloning the image many times into video scenes. Thus following the linearity, 2D filters weights are repeated N times according to time dimension and then rescaling by dividing by N. In this way, 3D ConvNet models can be implicitly train on Imagenet in which the pooled activation of boring video are the same with the input of single image. It means that the computation outputs of non-linearity layers, max and average pooling layers are as same as the 2D case (Mansimov et al., 2015).

4.3 Receptive field

To comparing with 2D models, another change to take into account is the receptive field of pooling and convolutional layers. As described before, an input vector for a convolutional neural network is an area of an image which is accessible to a filter per time and as more layers it is more increasing. 2D convolutions and pooling are symmetrical because they concentrate on image width and height (for example 7x7 kernel is accetable because it is symmetrical while a kernel 7x5 is not). When the temporal dimension is taken into consideration, it is necessary to determine the optimum receptive field, which is dependent on the frame rate and the picture dimensions of the image. (Schiappa, n.d.). According to Zisserman and Carreira, the problem of symmetric receptive field is that if it develops too rapidly in time compared to space, it may confuse edges from different objects, leading to poor feature detection; on the other hand, if it grows too slowly, dinamically scence might be fail to be captured (Carreira & Zisserman, 2017). In the final, I3D kernels are not symmetrical due to the inclusion of temporal dimension.

4.4 Network Architecture

The network diagram is as the figure below Carreira & Zisserman (2017)

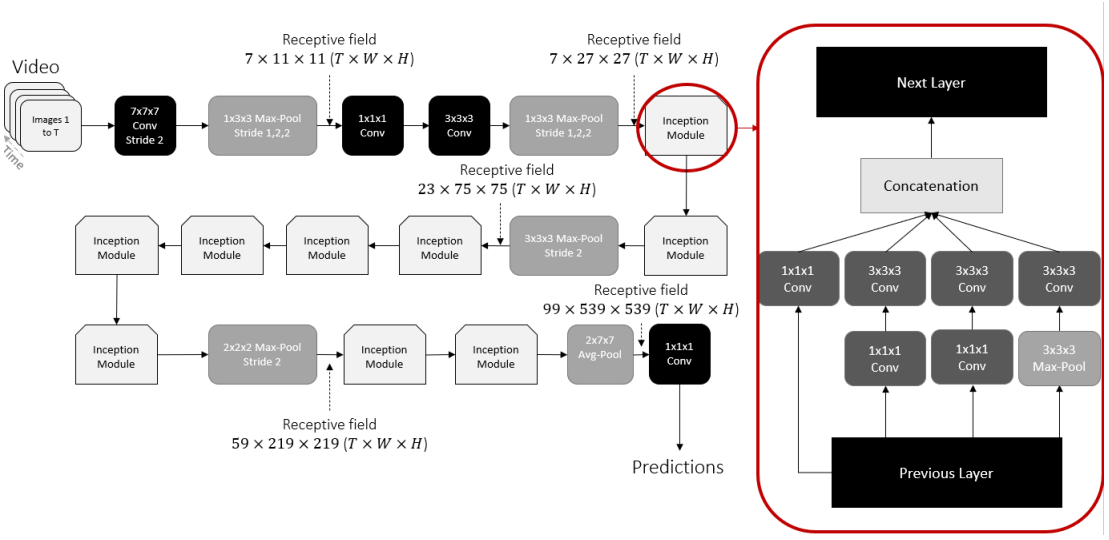


Figure 16: Network architect (Carreira & Zisserman, 2017)

To be seen in the diagram, as the begining an asymmetrical filters is used for

max-pooling, keep time retains while doing pooled operations over image width and height dimensions. So far, the convolutions and pooling which related to time dimension are performed in the later stage. In this paper, the Inception module is not described in details, however it is used widely in 2D classification networks. The Inception module aggregates the results after evaluating spatial (and time in case of classifying video) information at multiple scales. In the proposal of Inception model, after going through the $1 \times 1 \times 1$ convolution, the number of input channels is reduced before passing into the larger convolution $3 \times 3 \times 3$, so the computations are less expensive compared to the alternative.

5 Model evaluating

The objective of Machine Learning/Deep Learning model is trying to minimize the difference between the prediction and actual result (or called as *ground truth*). The process of calculating the difference is named lost/cost function. The classification problem generally can be divided into two types: multi-class classification and multi-label classification (Gomez, n.d.). The difference between multi-class and multi-label classification is that the prediction of multi-class is just belonged to only one class but the output of multi-label might be belonged to more than one classes (17).

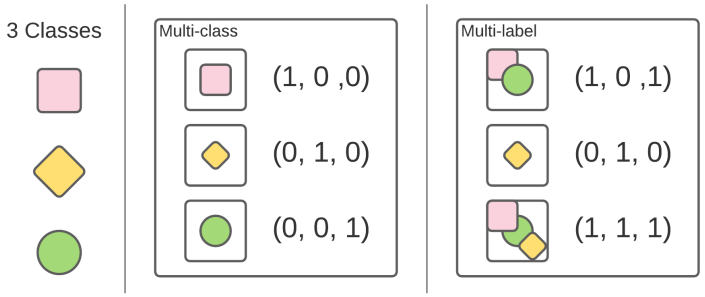


Figure 17: Multi-class vs Mult-label classification

Generally, the output vectors of CNNs are transformed to compute loss value by Sigmoid or Softmax functions. When applied Sigmoid function, the vector is scaled in range of 0 to 1, and for each element, it is calculated independently. Softmax function, it is not a loss, is used to produce scores s, in which an element

is a probability of the prediction belongs to the class s_i. The Softmax function depends on all elements of s. The Sigmoid (1) and Softmax (2) function can be calculated as equations below (Gomez, n.d.).

$$f(s_i) = \frac{1}{1 + e^{-s_i}} \tag{1}$$

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{2}$$

To measure the difference between to probability distributions of set of data, cross-entropy is applied. The cross-entropy loss is defined:

$$CE = -\sum_i^C t_i log(s_i) \tag{3}$$

Where t_i is the groundtruth, s_i is the score for each class C_i. Normally, before calculating the CE loss, an activation function (Softmax or Sigmoid) is applied to produce a score. Activation function is denoted as $f(s_i)$.

5.1 Categorical Cross-Entropy Loss

To evaluate the model of multi-class classification, Softmax activaton combined with Cross-Entropy loss is used. It means the output of CNNs is a probability over the C classes for each input item.

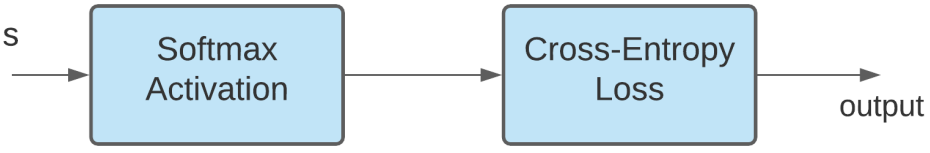


Figure 18: Softmax Cross-Entroy Loss

$$CE = -\sum_{i}^{C} t_i log(\frac{e^{s_i}}{\sum_j^C e^{s_j}}) \tag{4}$$

As normally, in case of multi-class problems, labels are reprented as one-hot vectors, thus only the class C_p (positive class) keeps its term in the loss. So far, only one element in the target vector is non-zero, $t_i = t_p$. Because $t_j = 0$ when $j \neq i$ and $t_i = 1$ when $i = p$, therefore the equation (4) can be written as

$$CE = -log(\frac{e^{s_p}}{\sum_j^C e^{s_j}}) \tag{5}$$

The next step is find out the optimum values for parameters in the network by calculating the gradient of loss function with respect to the output of CNN, then applying the back-propagation algorithm to optimize the loss function. So the gradient of CE loss respect each class score in s needs to be computed. Despite the groundtruth C_p, the gradient for other classes are the same, the derivative respect to the C_p class (eq. (6)) and other classes (eq. (7)) are defined (Koech, n.d.; Bendersky, n.d.):

$$\frac{\delta}{\delta s_p}(-log(\frac{e^{s_p}}{\sum_j^C e^{s_j}})) = (\frac{e^{s_p}}{\sum_j^C e^{s_j}} - 1) \tag{6}$$

$$\frac{\delta}{\delta s_n}(-log(\frac{e^{s_p}}{\sum_j^C e^{s_j}})) = (\frac{e^{s_n}}{\sum_j^C e^{s_j}}) \tag{7}$$

s_n is the score of negative classes differ from C_p

5.2 Binary Cross-Entropy loss

Instead of using a combiation between Softmax activation and cross-entropy, this loss function is using Sigmoid activation. Therefore it is also called as Sigmoid

Cross-entropy. While in Softmax activation a score of an elements depends on all elements, the output with Sigmoid activation are dependently to each others so that the loss estimated value for each output vector component is unaffected by the value of other component elements. This is the reason why Sigmoid Cross-Entropy is used in multi-label classification problems where prediction of an element belonging to a specific class should not affect the judgement for another class.

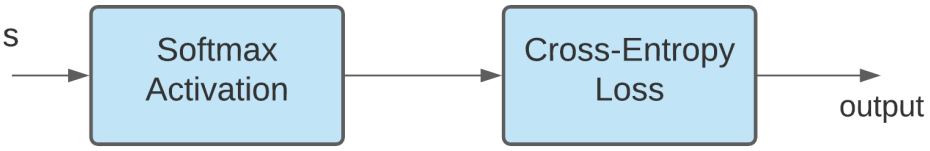


Figure 19: Binary Cross-Entropy Loss

In this case, an output is evaluated that it is belonging to the class C_i (for all classes C) or not, tha's why it is called *"binary"*. The local loss for each class C_i is defined (**??**)

$$\begin{aligned} CE &= -\sum_{i=1}^{C'=2} t_i log(f(s_i)) \\ &= -t_1 log(f(s_1)) - (1-t_1) log(1-f(s_1)) \end{aligned} \tag{8}$$

Where $f(s_i)$ is sigmoid function at (1). The problem of multi-label classification can be considered as multi *small* binary problems: is the output belonged to the class C_i for every class C? Therefore each C_i class, t_1 and s_1 is the groundtruth and the score of the setup binary problem within the class. It is known that the value of t_1 is binary (only 0 or 1, positive or negative) so the equation (8) can re-write as

$$f(x) = \begin{cases} -log(f(s_1)) \text{if } t_1 = 1 \\ -log(1-f(s_1)) \text{if } t_1 = 0 \end{cases} \tag{9}$$

As saying before, by using sigmoid activation, the score of each class is independent with other class, only effect by the loss of binary problem itself. Thus the gradient respect to the score s_i is

$$\begin{aligned}\frac{\delta}{\delta s_i}(CE(f(s_i))) &= t_1(f(s_1)-1)+(1-t_1)(f(s_1)) \\ &= \begin{cases} f(s_1)-1 \text{if } t_1 = 1 \\ f(s_1) \text{if } t_1 = 0 \end{cases}\end{aligned} \tag{10}$$

For the global loss of the model, it is calculated by sum up local loss of each class C_i. This is the way to evaluate the thesis model.

6 Optical flow

6.1 Introduction

Optical flow is a representation of the mobility of a scene environment in relation to an observer's position. In the problem of recognizing human gesture, a static scence in a single frame can lead to an ambiguity in interpreting gesture class. The movement of human bodies must be taken advantage of in order to have a thorough grasp of the gesture class. There are a variety methods for calculation optical flow from frame to frame, including region-based, energy-based, differential-based, phase-based (Marco & Farinella, 2018). The differential technique is the most frequently used method, and it is based on the premise of picture brightness constancy (Horn & Schunck, 1981).

Figure 20: Example of an optical flow frame (NVIDIA, n.d.)

Assume that the constant brightness of a pixel *(x, y)* in an object does not change; it is not true in reality, however it is reasonable and natural when the time is small enough. Therefore, the image intensity of two continuous frames can expressed by the equation below (O'Donovan, 2005).

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{11}$$

Then, using Tayloy Series Approximation of the RHS:

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \Rightarrow \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \tag{12}$$

Next, dt is divided to derive optical flow equation

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{13}$$

Noted that $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$ and $\frac{dI}{dx}$, $\frac{dI}{dy}$, $\frac{dI}{dt}$ can be considered as the image gradients along to horizontal, vertical and time dimensions. So far, optical flow problem is that sovling $u(\frac{dx}{dt})$ and $v(\frac{dx}{dt})$ (O'Donovan, 2005). However, the issue could not be solved with only one equation with two unknow variables.

There are two type of optical flow: Sparse and Dense optical. In details, Sparse optical flow requires a pre-processing step in which some *"interesting features"*

(pixels of object's edges or corners) are detected and then only calculate flow vectors of those pixels (en Lin, n.d.). On the other hand, computation cost for Dense Optical Flow is expensive and slow performance (Wulff et al., 2017) because it provides flow vectors of entire frames, one flow vector per pixel.

6.2 Algorithm for converting RGB video into Optical Flow video

6.2.1 Lucas-Kanade: Sparse Optical Flow

As mentioned above, the initial step of calculating Sparse optical flow is extracting interesting features which are points in images and present rich content information (Harris et al., 1988). They are made up of two main things:

- **Interesting points:** The points which are immune when an image is rotated, translated, scaled or intensified. For instance, pixels which are related to edges, corners or blobs can be considered as interest points inside an image (Derpanis, 2004).
- **Feature Descriptors:** These findings outline the patch of images near the interest points in vectors, from raw pixel to complex Histogram of Gradients (HoG) (Derpanis, 2004).

One of a popular method for corner detection is Harris Corner Detector which was proposed in 1988 by Chris Harris and Mike Stephens in the paper "A combined Corner and Edge Detector". The idea of them is that determine the intensity difference for a movement of *(u, v)* in all directions. It is expressed in the mathematic form as (Harris et al., 1988):

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} [\underbrace{I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2 \tag{14}$$

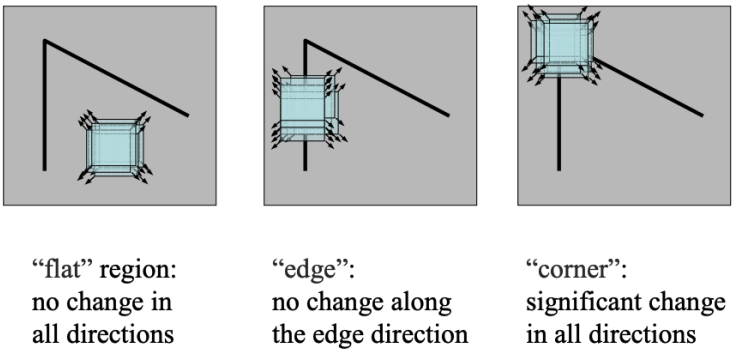


Figure 21: Harris Corner Dector basic idea

The Harris Corner Detector includes three primary steps (Ryu et al., 2011):

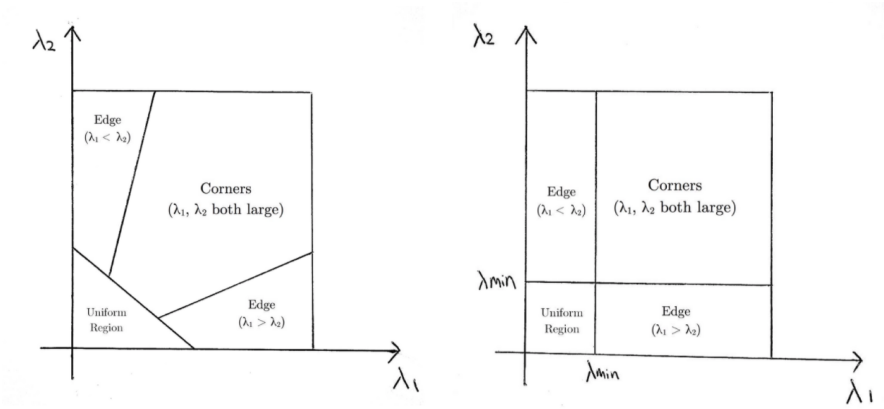
- It identifies which windows (small picture patches) generate extremely significant changes in intensity when moved in both the X and Y axes (i.e. gradients).
- Compute a score R for each window found.
- Important corners are selected and marked by applying an threshold to the score.

Another algorithm for corner dectected is Shi-Tomasi Corner Detector which was presented by Jianbo Shi and Carlo Tomasi in 1994 (Shi et al., 1994). The algorithm mostly as same as Harris Corner Detector, however the R score is calculated differently which makes the result better. Furthermore, Shi-Tomasi focuses only in the top N corners therefore it might be beneficial in case that each and every corner do not want to be detected.

Deep into the calcultion of R score to identify the difference between HDC and SCD.

Table 3: HDC vs SDC Scoring function

Harris Corner Detector	Shi-Tomasi Corner Detector
$R = detM - k(traceM)^2$ $detM = \mu_1\mu_2$ $traceM = \mu_1 + \mu_2$	$R = min(\mu_1, \mu_2)$



(a) Scoring function in HDC (b) Scoring function in SDC

Figure 22: Scoring function between HDC and SDC on μ_1 - μ_2 space

The pictures below show the result of Harris and Shi-Tomasi Corner Detector.

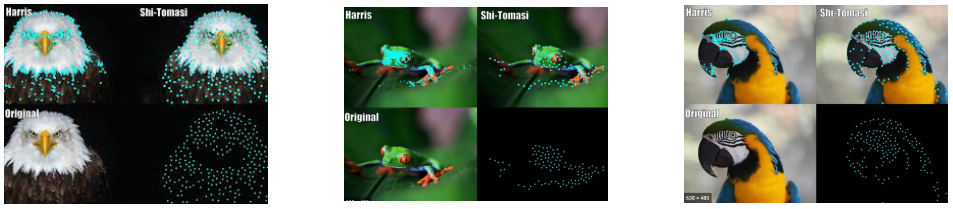


Figure 23: Result of HDC and SDC algorithm for corner detection

Then based on an assumption that in the very small period (δt), the object is not moving much and objects in a frames is changed smoothly in shade of gray. So with this assumption, Lucas - Kanade proposed an idea that a small neighborhood $n \times n$ window arround features detected (by Shi-Tomashi detector) have the same motion. Therefore the partial derivates of image I respect to positon (x, y) and time t for a pixel q inside the window is described as equation 15. It is Optical Flow Equation that described earlier.

$$I_x(q_n)V_x + I_y(q_n)V_y = -It(q_n) \tag{15}$$

Then in a form of matrix $Av = b$ where

$$A = \begin{bmatrix} I_x(q_n) & I_y(q_n) \end{bmatrix}$$

$$v = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

$$b = \begin{bmatrix} -I_t(q_n) \end{bmatrix}$$

Now the issue is that solving two unknown variables v_x and v_y with n equations, so it is over-determined. To deal with this problem, least squares fitting is used as follow:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix} \tag{16}$$

Where $v_x = u = \frac{dx}{dt}$, $v_y = v = \frac{dy}{dt}$ indicate the movement of x and y over the time. The optical flow problem is completed when the two variable is sovled. So the algorithm identifies some key points and then calculate the optical flow vectors of these points. However, becauses of the assumption above, Lucas-Kaneda approach is only suitable for tracking object with slightly movements and not successful for large motion. Then the idea comes up compute optical flow from difference resolutions is offered with Open CV library. It is called pyramids method. In details, small motions can be ignored and the large motions are scaled to be small motions and then computed the optical flow based on the scaled motions.

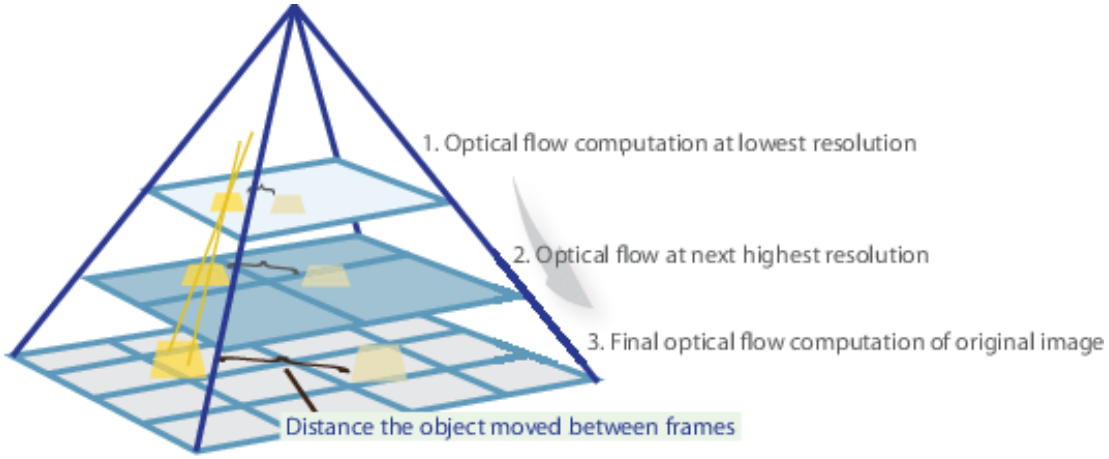


Figure 24: Lucas - Kaneda pyramid method

Below pseudo-code show how to convert RGB frames into optical flow by using Lucas - Kanade algorithm with supported by OpenCV library.

Convert to OF Convert video to optical flow with Lucas - Kanade

Require: *video_input*

feature_params ← *params for corner detection*
lk_params ← *params for Lucas − Kanade optical flow*
color ← *create some random colors*
old_frame ← *the first frame of video input*
old_gray ← *convert to gray image*
$p0$ ← *find interesting points* ▷ using corner detection
mask ← *create a mask image*
while reading video frames **do**
    frame ← *current frame*
    frame_gray ← *convert into gray image*
    $p1, st, err$ ← *calculate optical flow* ▷ using cv2.calcOpticalFlowPyrLK
    ▷ Select good points
    good_new_points ← *good points based on* $p1$
    good_old_points ← *good points based on* $p0$
    for $(new\_point, old\_point) \in (good\_new\_points, good\_old\_points)$ **do**
        (a, b) ← *coordinate of new point*
        (c, d) ← *coordinate of old points*
        mask ← *draw line on mask from (a, b) to (c, d)random color color*
        frame ← *draw circle center (a, b), radius r*
    end for
    image_optical_flow ← *image by (frame, mask)*
    video_optical_flow ← *append image*
    ▷ Update previous frame and points
    old_gray ← *current frame_gray*
    $p0$ ← *curren list of new good points*
end while

6.2.2 Farneback: Dense Optical Flow

Unlikely Sparse Optical Flow which only focuses on some reliable points in the frame, in Dense Optical Flow all the pixels are involved to detect change in pixel intensity between two frames. At first, a quadratic polynomial 17 is used to approximate pixel neighborhood

$$f_1(x) = x^T A_1 x + b_1^T x + c_1 \tag{17}$$

And with the global displacement factor $\mathbf{d}$, the signal f_2 is created following (18)

$$\begin{aligned} f_2(x) &= f_1(x - d) \\ &= (x - d)^T A_1 (x - d) + b_1^T (x - d) + c_1 \\ &= x^T A_1 x + (b_1 - 2A_1 d)^T x + (d^T A_1 d - b_1^T d + c1) \\ &= x^T A_2 x + b_2^T x + c_2 \end{aligned} \tag{18}$$

Thus

$$A_2 = A_1 \tag{19}$$

$$b_2 = b_1 - 2A_1 d \tag{20}$$

$$c_2 = d^T A_1 d - b_1^T d + c1 \tag{21}$$

It can be seen that the translation $\mathbf{d}$ can be solved at least A_1 is non-singular by the equation (20) (Farnebäck, 2003)

$$2A_1 d = -(b_2 - b_1) \Rightarrow d = -\frac{1}{2} A_1^{-1}(b_2 - b_1) \tag{22}$$

The Farneback optical flow algorithm can be explained following the steps:

- The windows of image frames is approximated by quadratic polynomials follow polynomial expansion transform as above.
- Based on the polynomial transform under motion, estimation of displacement fields is defined from polynomial expansion coefficients. Then dense optical flow is calculated after some refinements (in the scope of this thesis, the details of calculation steps are not referred).

To visualize the optical flow with OpenCV, from the 2 channels array of flow vectors ($\frac{dx}{dt} and \frac{dy}{dt}$) (from optical flow problem (13)), the magnitude and direction (angle) of flow vectors are computed (2D vector) (CV, n.d.).

$$magnitude(I) = \sqrt{x(I)^2 + y(I)^2} \tag{23}$$

$$angle(I) = \arctan2\left(y(I), x(I)\right) \frac{180}{\pi} \tag{24}$$

The below pseudocode is show how conduct an optical frames from a RGB video.

Convert to OF Convert into dense optical flow video

Require: *video_input*

cur_frame ← *first video frame*
cur_frame ← *convert frame into grayscale frame*
hsv_mask ← *matrix* 0 *with size of frame*
hsv_mask ← *set image saturation to a maximum value* 255
while reading video frames **do**
    prev_frame ← *get previous grayscale frame*
    cur_frame ← *current frame*
    cur_frame ← *convert cur_frame into grayscale*
    flow ← *calculate optical flow based on prev_frame and cur_frame* ▷ usingcv2.calcOpticalFlowFarneback
    magnitude, *angle* ← *calculate flow vectors magnitude and angle* ▷ using cv2.cartToPolar
    ▷ Set value as per the normalized magnitude of optical flow
    hsv_mask ← *angle* ∗ 180 / π / 2
    ▷ Set value as per the normalized magnitude of optical flow
    hsv_mask ← *nomalized magnitude*
    rgb_representation ← *convert hsv_mask from HSV to RGB*
end while

This algortithm is used in the thesis. However, for the flow system, it is not needed to represent flow frames therefore to generate flow tensor input, the process is stopped at generating flow vectors *(u, v)* by using function *calcOpticalFlowFarneback* which is supported by OpenCV. So the size of flow tensonr is $(num_frames, heigh, width, 2)$, the number 2 indicates 2 channels of flow image, while the input size of RGB tensor is $(num_frames, heigh, width, 3)$, 3 represents 3 channel RGB of an image.

(a) RGB Frame (b) Flow Frame

Figure 25: The result of optical flow frame after converting from RGB Video frame

7 Experiment

7.1 Datasets

As mentioned above, in this thesis, the two datasets, ASL64 and WLASL, are examined. The group of authors provide both raw and pre-processed videos. To reduce time and complexicity of the project, only pre-processed videos are used for training and testing purposed.

7.1.1 Argentina Sign Language Dataset

The dataset was created for providing a dictonary for LSA and training purpose of automatic sign language recognition. This dataset contain 3200 videos related to 64 signs indepently executed by non-experts. These are frequently signs used by LSA (Ronchetti et al., 2016).

With LSA64 datasets, authors provide 3 versions: raw version, cut/temporary segmented version and extracted version. The raw and cut/temporary version are the same, however some begining and ending scenes in cut version are removed to make the hands are static when videos start or end. On the other hands, the extracted version is that the authors extracted the position of hands and head for each frame, only hand in black background are kept with segmented image of each frame.

(a) Scene in extract version of LSA64 dataset

(b) Scene in cut version of LSA64 dataset

Figure 26: LSA64 dataset

The project is utilized the cut version of this dataset. Because of using dense flow vectors for flow stream, the motions of the whole image's pixels are captured instead of some interesting features such as left-right hand or head.

7.1.2 Word Level American Sign Language Dataset

The dataset of WLASL are collected from the internet with two main reliable sources, education sites of sign language such as ASL-LEX and ASLU (D. Li et al., 2020), and the most of video is downloaded from ASL tutorial video on Youtube. The videos uploaded in educational websites are verified by experts (University, n.d.; Caselli et al., 2017); on the other hands, selected ASL tutorial videos have clear tilte, between the gloss and signal, and only one sign (might be the sign is repeated multiple times in the video) with different backgrounds (D. Li et al., 2020). For the next step, the authors guarantee that for each gloss in the dataset the number of videos is at least seven videos so that it is possibly to spit for training and validating sets.

At pre-processing step, YOLO 3 networks is employed to train to identify human in a video and as a result, it provices a largest boundary box of person during time frames. After that, the videos are cropped based on the human bounded box from YOLO training. By doing this, the model is only forcus on signer's

action and remove unrelated objects in the background, increasing accuracy of the model. They also provide temporal boundary whereas the starting and ending frames of the signs are defined in each video.

Also from the group of authors, the shortest video is around 0.36 seconds and the longest video is 8.12 seconds, with the avarage time of the whole dataset is around 2.41 seconds. The intra-class standard deviation is 0.85 seconds on avarage. At the end, the authors split the dataset into 4 subsets depends on the sorting of sample size for each gloss, and the top K glosses are selected (K is 100, 300, 1000, 2000) (D. Li et al., 2020).

Dataset	**#Gloss**	**#Videos**	**Mean**	**#Signers**
WLASL100	100	2038	20.4	97
WLASL300	300	5117	17.1	109
WLASL1000	1000	13168	13.2	116
WLASL2000	2000	21083	10.5	119

Table 4: Statistic with WLASL dataset

7.1.3 Data Preprocessing and Augmentation

To reduce the heavily computations, firstly the solution of original videos in the datasets is reduced as 256 x 256, secondly the whole video frames are randomly crop as 224 x 244, and with 50% of probability of flipping horizontal. Lastly, only 64 consecutive frames are extracted to be feed into the network (both streams). This procedure lead to increasing of model's accuracy (Karpathy et al., 2014).

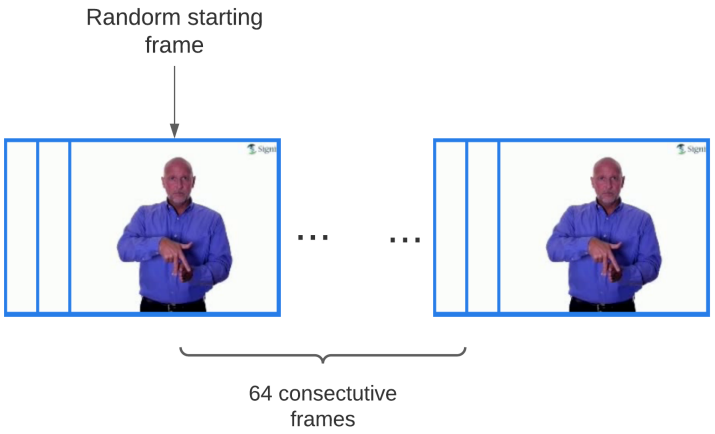


Figure 27: Video pre-processing

7.2 Setup environment

7.2.1 Python

The whole project is implemented with Python programming language. The reason is that Python is a high level programing language which is commonly used in Data Science, Machine Learning or Deep Learning project. Moreover, there are many libraries are well-supported for building, producing deep learning networks and algorithm, namely Keras, Pytorch, Tensor Flow, Numpy, Pandas and also for data visualization such as Seaborn, Matplotlib.

Besides, OpenCV is a most favourite framework for image and video processing. OpenCV supports most of commonly functions: crop, flip, rotate, reading video frame, translate into grayscale or other format as well as many complex functions like corner detection (Harris Corner Detector, Shi-Tomashi Corner Detector), flow calculation (sparse and dense optical flow).

7.2.2 Goolge Colaboratory

Because of limitation of personal machine, the project is runing on cloud, in details, Google Colaboratory (or Google Colab in short term). Google Colab is a product from Google Research whereas python code can be executed arbitrary

through the web browser. The advantage of Google Colab is that we can utilize the strength of GPU on Google Server to perform training and validating processes. However, there are some side effects of using Colab product. Firstly, from colab environment, the data which is stored locally is inaccessible, it is required to move the whole datasets into cloud storage (in this case, using Google Drive), and with large datasets, it takes long time to be uploaded and also storage-consuming. Secondly, Google only allows 24-hour session, it means after 24 hours training the session is terminated. Lastly, because of sharing resources, sometimes GPUs are occupied by others.

7.2.3 Project details

The project is split into multi parts:

- Utilities: all utilities functions that support data processing, included video processing as metioned above (cropping, flipping, converting RGB to Flow Vectors), and also function for converting an input array into tensor which is used in the network.
- Network model: this indicates the structure of each layer in I3D network 16. The code is following strictly the structure of I3D network model trained on Kinetics (Szegedy et al., 2015)
- Data processing: this is used to make up data which is used in the network by utilizing Dataset and DataLoader which supported by Pytorch library. By using these, data can be easily loaded for training and testing, also it is easy splitted into multi batches for training purpose.
- Training: in this section, the training process trains and evaluate the set of training data; also the network stage is kept track for every epoch in case that the session is terminated (as mentioned above), the weigths can be loaded to train continue, avoid for training everything from begining.
- Testing: with the test set, the model tries to predict the gloss of videos.

So far, the model trains ASL and WLASL independently to see how good it is for separated datasets.

7.2.4 Hyperparameter

For the initial configuration, the training parameters are setup with:

- Batch size: the batch size is 8
- Input size is $(64 \times 256 \times 256 \times 3)$ and $(64 \times 256 \times 256 \times 2)$ for RGB stream and flow stream respectively.
- Number of class: it is depend on the dataset. We have trained with WLASL100, WLASL300, WLASL1000 and ASL64.
- Learning rate: initial value is 0.001, and during the training process, I used Adam optimizer with Pytorch library to optimze learning rate for every 8 steps.
- Number of epoch: using early stopping technique.

7.3 Results

8 Conclusion

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017a). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (icet)* (pp. 1–6).

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017b). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (icet)* (p. 1-6). doi: 10.1109/ICEngTechnol.2017.8308186

Bendersky, E. (n.d.). The softmax function and its derivative [Computer software manual]. Retrieved from `https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/`

Britannica, T. E. o. E. (2020, November 12). Sign language. encyclopedia britannica [Computer software manual]. Retrieved from `https://www.britannica.com/topic/sign-language`

Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6299–6308).

Caselli, N. K., Sehyr, Z. S., Cohen-Goldberg, A. M., & Emmorey, K. (2017). Asllex: A lexical database of american sign language. *Behavior research methods*, *49*(2), 784–801.

CV, O. (n.d.). Operations on arrays [Computer software manual]. Retrieved from `https://docs.opencv.org/master/d2/de8/group__core__array.html`

Derpanis, K. G. (2004). The harris corner detector. *York University*, *2*.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2625–2634).

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Education, I. C. (n.d.-a). Convolutional neural networks [Computer software manual]. Retrieved from https://www.ibm.com/cloud/learn/convolutional-neural-networks#toc-what-are-c-MWGVhUiG

Education, I. C. (n.d.-b). Neural networks [Computer software manual]. Retrieved from https://www.ibm.com/cloud/learn/neural-networks

en Lin, C. (n.d.). Introduction to motion estimation with optical flow [Computer software manual]. Retrieved from https://nanonets.com/blog/optical-flow/#opticalflow

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on image analysis* (pp. 363–370).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

Gomez, R. (n.d.). Understanding categorical cross-entropy loss, binary cross-entropy loss, softmax loss, logistic loss, focal loss and all those confusing names [Computer software manual]. Retrieved from https://gombru.github.io/2018/05/23/cross_entropy_loss/

Harris, C., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, pp. 10–5244).

Holtz, R. (2014). Reading between the signs: Intercultural communication for sign language interpreters. *Journal of International Students*, *4*(1), 106–108.

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, *17*(1-3), 185–203.

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 221-231.

Jiang, S., Sun, B., Wang, L., Bai, Y., Li, K., & Fu, Y. (2021). Skeleton aware multi-modal sign language recognition. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 3413–3423).

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1725–1732).

Koech, K. E. (n.d.). Cross-entropy loss function [Computer software manual]. Retrieved from `https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e`

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Laptev, I. (2005). On space-time interest points. *International journal of computer vision*, *64*(2), 107–123.

Laptev, I., Marszalek, M., Schmid, C., & Rozenfeld, B. (2008). Learning realistic human actions from movies. In *2008 ieee conference on computer vision and pattern recognition* (pp. 1–8).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, D., Rodriguez, C., Yu, X., & Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 1459–1469).

Li, Y., Wang, X., Liu, W., & Feng, B. (2018). Deep attention network for joint hand gesture localization and recognition using static rgb-d images. *Information Sciences*, *441*, 66–78.

Lim, K. M., Tan, A. W. C., Lee, C. P., & Tan, S. C. (2019). Isolated sign language recognition using convolutional neural network hand modelling and hand energy image. *Multimedia Tools and Applications*, *78*(14), 19917–19944.

Liu, J., Luo, J., & Shah, M. (2009). Recognizing realistic actions from videos "in the wild". In *2009 ieee conference on computer vision and pattern recognition* (pp. 1996–2003).

Mansimov, E., Srivastava, N., & Salakhutdinov, R. (2015). Initialization strategies of spatio-temporal convolutional neural networks. *arXiv preprint arXiv:1503.07274*.

Marco, L., & Farinella, G. M. (2018). *Computer vision for assistive healthcare*. Academic Press.

Mercanoglu Sincan, O., Junior, J., CS, J., Escalera, S., & Yalim Keles, H. (2021). Chalearn lap large scale signer independent isolated sign language recognition challenge: Design, results and future research. *arXiv e-prints*, arXiv–2105.

Mutegeki, R., & Han, D. S. (2020). A cnn-lstm approach to human activity recognition. In *2020 international conference on artificial intelligence in information and communication (icaiic)* (pp. 362–366).

Niebles, J. C., Chen, C.-W., & Fei-Fei, L. (2010). Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision* (pp. 392–405).

NVIDIA. (n.d.). An introduction to the nvidia optical flow sdk [Computer software manual]. Retrieved from `https://developer.nvidia.com/blog/an-introduction-to-the-nvidia-optical-flow-sdk`

O'Donovan, P. (2005). Optical flow: Techniques and applications. *International Journal of Computer Vision*, 1–26.

Ronchetti, F., Quiroga, F., Estrebou, C., Lanzarini, L., & Rosete, A. (2016). Lsa64: A dataset of argentinian sign language. *XX II Congreso Argentino de Ciencias de la Computación (CACIC)*.

Ryu, J.-B., Lee, C.-G., & Park, H.-H. (2011). Formula for harris corner detector. *Electronics letters*, *47*(3), 180–181.

Saha, S. (n.d.). A comprehensive guide to convolutional neural networks — the eli5 way [Computer software manual]. Retrieved from `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`

Schiappa, M. (n.d.). Understanding the backbone of video classification: The i3d architecture [Computer software manual]. Retrieved

from https://towardsdatascience.com/understanding-the-backbone-of-video-classification-the-i3d-architecture-d4011391692

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 806–813).

Shi, J., et al. (1994). Good features to track. In *1994 proceedings of ieee conference on computer vision and pattern recognition* (pp. 593–600).

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*.

Sivic, J., & Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Computer vision, ieee international conference on* (Vol. 3, pp. 1470–1470).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Taylor, G. W., Fergus, R., LeCun, Y., & Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *European conference on computer vision* (pp. 140–153).

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the ieee international conference on computer vision* (pp. 4489–4497).

University, A. (n.d.). Asl dataset [Computer software manual]. Retrieved from http://asluniversity.com/

Wang, H., & Schmid, C. (2013). Action recognition with improved trajectories. In *Proceedings of the ieee international conference on computer vision* (pp. 3551–3558).

Wulff, J., Sevilla-Lara, L., & Black, M. J. (2017). Optical flow in mostly rigid scenes. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4671–4680).

Xiao, Q., Chang, X., Zhang, X., & Liu, X. (2020). Multi-information spatial–temporal lstm fusion continuous sign language neural machine translation. *IEEE Access*, *8*, 216718–216728.

Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4694–4702).

Zha, S., Luisier, F., Andrews, W., Srivastava, N., & Salakhutdinov, R. (2015). Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144* .