BHUTANESE SIGN LANGUAGE HAND-SHAPED ALPHABETS AND DIGITS DETECTION AND RECOGNITION

KARMA WANGCHUK

A Thesis Submitted to the Graduate School of Naresuan University
in Partial Fulfillment of the Requirements
for the Master of Engineering in (Computer Engineering - (Type A 2))
2020

BHUTANESE SIGN LANGUAGE HAND-SHAPED ALPHABETS AND DIGITS DETECTION AND RECOGNITION

KARMA WANGCHUK

A Thesis Submitted to the Graduate School of Naresuan University
in Partial Fulfillment of the Requirements
for the Master of Engineering in (Computer Engineering - (Type A 2))
2020

Thesis entitled "Bhutanese Sign Language Hand-shaped Alphabets and Digits Detection and Recognition"

By KARMA WANGCHUK

has been approved by the Graduate School as partial fulfillment of the requirements for the Master of Engineering in Computer Engineering - (Type A 2) of Naresuan University

Oral Defense Committee

..... Chair
(Dr. Choochart Haruechaiyasak, Ph.D.)

..... Advisor
(Assistant Professor Dr. Panomkhawn Riyamongkol, Ph.D.)

..... Internal Examiner
(Associate Professor Dr. Panus Nattharith, Ph.D.)

..... Internal Examiner
(Associate Professor Dr. Phongphun Kijsanayothin, Ph.D.)

Approved

.....
(Paisarn Muneesawang, Ph.D.)

Dean of the Graduate School

Title BHUTANESE SIGN LANGUAGE HAND-SHAPED ALPHABETS AND DIGITS DETECTION AND RECOGNITION

Author KARMA WANGCHUK

Advisor Assistant Professor Dr. Panomkhawn Riyamongkol, Ph.D.

Academic Paper Thesis M.Eng. in Computer Engineering - (Type A 2), Naresuan University, 2020

Keywords Bhutanese Sign Language, BSL Dataset, Convolutional Neural Network, Visual Geometry Group, Image augmentation

ABSTRACT

The communication problem between the deaf and the public is an emerging concern for both parents and the government of Bhutan. The parents are not able to understand their children. The deaf students are not able to communicate with the general public. Therefore, deaf school and government is urging people to learn Bhutanese Sign Language (BSL) but learning Sign Language (SL) is not easy. However, Computer Vision and machine learning applications have been solving communication gaps. It has been easy to learn and understand SL with the help of signs' translation apps. The basics of all sign languages are alphabets and numbers. The purpose of this study is to develop a suitable machine learning model to detect and recognize the BSL alphabets and digits using BSL hand-shaped alphanumeric datasets.

In this study, the first BSL hand-shaped alphanumeric dataset was created with different augmentation techniques. Different SL models were evaluated with the dataset. However, the Convolutional Neural Network (CNN) based architecture outperformed them. Using six layers of CNN with the batch normalization and different dropout ratios, 20000 digits dataset, and 30000 alphabets dataset obtained better results compared to LeNet-5, SVM, KNN, and logistic regression. Furthermore, ResNet with 43 convolutional layers obtained the best training and validation accuracy of 100% and 98.38% respectively on 60,000 alphanumeric datasets. This research is the first of its

kind to study the possibility of machine learning integration with the BSL to detect and recognize hand-shaped alphabets and digits. It was found that machine learning models can be deployed to develop Computer Vision applications to make BSL learning easier and accessible to the general public. Further studies are needed to create a video-based dataset and study BSL dynamic gesture recognition for word translation.

ACKNOWLEDGEMENTS

This study would not have been possible without the scholarship from the Office of the Gyalpoi Zimpon and Naresuan University. Therefore, I would like to extend my sincere and profound gratitude to His Majesty the King of Bhutan and the Department of Electrical and Computer Engineering for allowing me to pursue a master's studies. And thanks to the College of Science and Technology and the Royal University of Bhutan for believing in me and allowing me to pursue higher studies. Moreover, I am indebted to the following people, without whom I would not have been able to complete this thesis and master's degree. I would like to thank my advisor Assistant Professor Dr. Panomkhawn Riyamongkol and Lecturer Mr. Rattapoom Waranusast for their timely guaranteed help and support, critical feedback, patience, and encouragement. Additionally, I would like to extend my heartfelt gratitude to the department and faculty secretaries, who have supported me relentlessly for the smooth completion of the study.

Furthermore, I would like to extend special appreciation to international students at Naresuan University, who allowed me to capture images and record videos patiently, and without whom I would not have collected enough BSL dataset. Besides, I would also like to thank lab members for their unwavering help and support especially Mr. Yonten Jamtsho (GCIT) and Mr. Sittisak. And my roommates, Mr. Thongley (JNEC) and Mr. Ongpo Lepcha Sukommu (CNR) for sharing happiness, stress, and excitement. Above all, withstanding our morning alarm was the toughest of all.

Lastly, I would like to extend my biggest gratitude to my wife Rinchen T. Peldon for giving me much-needed moral support and upbringing our sons to the best of her ability. I understand the difficulty she has gone through all these years. She has been amazing. For my sons, Tandin Gawa Wangyal and Sonam Norbu Tshering, I am sorry for not being able to video call often lately. I will see you soon and spend quality time with you all. Promised!

KARMA WANGCHUK

TABLE OF CONTENTS

Page

List of tables

Page

List of figures

Page

CHAPTER I
BACKGROUND OF THE STUDY

1.1 Introduction

Bhutan is one of the smallest Himalayan kingdoms, landlocked and sandwiched between two gigantic countries namely China and India in the North and South respectively. Bhutan remained happily self-imposed isolation for centuries untouched and unexplored. She has 71% of the country under forest cover and the constitution of Bhutan mandates 60% of the land under forest cover for all times to come. She is the only carbon-negative country in the world. However, to keep on par with modernization and globalization, Bhutan gradually started to open to the outside world by joining the Universal Postal Union and United Nations in 1961 and 1971 respectively.

Bhutan is enhancing both regional and international cooperation by establishing foreign relationships and cross-border connectivity. In addition, with the introduction of the Internet and television in 1999, Bhutan is developing and modernizing at a faster rate while keeping customs and traditions intact. Surprisingly, in the era of technology, Bhutan is the only country in the world that does not have a single traffic light even in the capital city and is directed by policemen (Walcott, 2009). Further, to accelerate socioeconomic in the country, modern infrastructures such as international airports, roads, bridges, and schools were constructed.

In 1914, Gongzim Ugyen Dorji supported by the first King of Bhutan, Gongsa Ugyen Wangchuck, established the first school in Haa (Dorji, 2008). This marked the beginning of modern education in Bhutan. Some of these students were selected and sent to schools in Kalimpong and Darjeeling in India on the government's scholarship to pursue higher studies. The first batch of 46 boys was sent to India to joined at Graham's Homes, a Scottish mission school in Kalimpong to study (Wangmo & Choden, 2010). These scholars returned to Bhutan and served the King and the people. They worked tirelessly, in their different capacities, to help the government in planning and developing modern Bhutan. The tremendous progress in the education system, over the years, saw the increased number of schools and colleges in the country with the best teaching and learning facilities. The schools are being upgraded to higher secondary

schools and colleges. Besides, the Royal Government of Bhutan has given due importance to the Deaf Community by establishing the Deaf School and urge people to learn the sign language.

The Deaf Education Unit (DEU) is the only deaf school in Bhutan located about 12km from Paro. The institution was established in 2003 with Drukgyel Lower Secondary School but in 2014, DEU became a standalone school and named it Wangsel Institute for the Deaf. The institute has students with moderate to severe to profound hearing loss. The mode of communication and teaching is through Bhutanese Sign Language (BSL). The institute provides an opportunity for both deaf children and parents to study BSL.

Table 1 Total number of teaching staff in 2020

Teachers	Instructors	Volunteers	Linguist	Contract Teacher	Total
26	6	2	1	1	36

The Research Team at the institute is working since 2003 with support from the Thailand linguistics team to study and document sign language. There are around 2800 signs formally documented and persistently working on sign language to standardize BSL. The documentation of sign language would assist teaching staff to improve teaching and learning pedagogies. Currently, there is 36 teaching staff which includes 26 teachers, six instructors, two volunteers, a linguist, and a teacher on contract as shown in Table 1. The number of students in the institute is 103 (55 boys and 48 girls) as shown in Table 2. The classes start from preparatory to class XI. Students are taught in both Dzongkha and English like any other students in the school but the medium of instruction is in sign language.

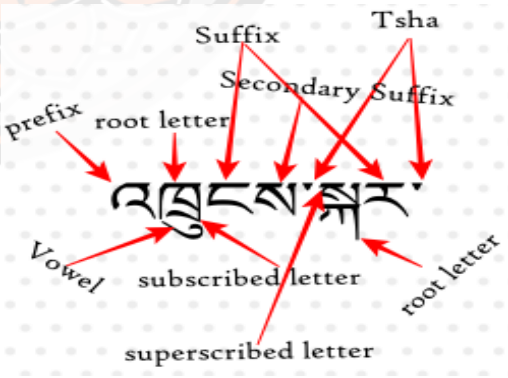
Table 2 Total number of students in 2020

Class	Male	Female	Total
Preparatory	2	0	2
PP	1	2	3
I	3	5	8
II	0	5	5

III	5	3	8
IV	3	5	8
V	7	9	16
VI	5	4	9
VII	7	3	10
VIII	6	3	9
IX	7	4	11
X	5	2	7
XI	4	3	7
XII	0	0	0
Total	**55**	**48**	**103**

The Dzongkha hand-shaped alphabets and digits are illustrated in Figure 1 and Figure 2 respectively. There are 30 Dzongkha Alphabets (Wangchuk, Riyamongkol, & Waranusast, 2020a) and 4 Vowels as shown in Table 3. The hand-shaped vowels are shown in Figure 3. Different syllables are formed by adding alphabets and vowels either on top (superscribed) or bottom (subscribed) of the root letter to form a word. Words are separated by a delimiter called ***Tsha*** (period).

Table 3 Dzongkha Alphabets Transcription, vowels, and word formation

Alphabets						Vowels	Words
ཀ	ཁ	ག	ང	ཅ	ཆ	ི ུ ེ ོ	
ka	kha	ga	nga	ca	cha		
ཇ	ཉ	ཏ	ཐ	ད	ན		
ja	nya	ta	tha	da	na		
པ	ཕ	བ	མ	ཙ	ཚ		
pa	pha	ba	ma	tsa	tsha		
ཛ	ཝ	ཞ	ཟ	འ	ཡ		
dza	wa	zha	za	a`	ya		
ར	ལ	ཤ	ས	ཧ	ཨ		
ra	la	sha	sa	ha	a		

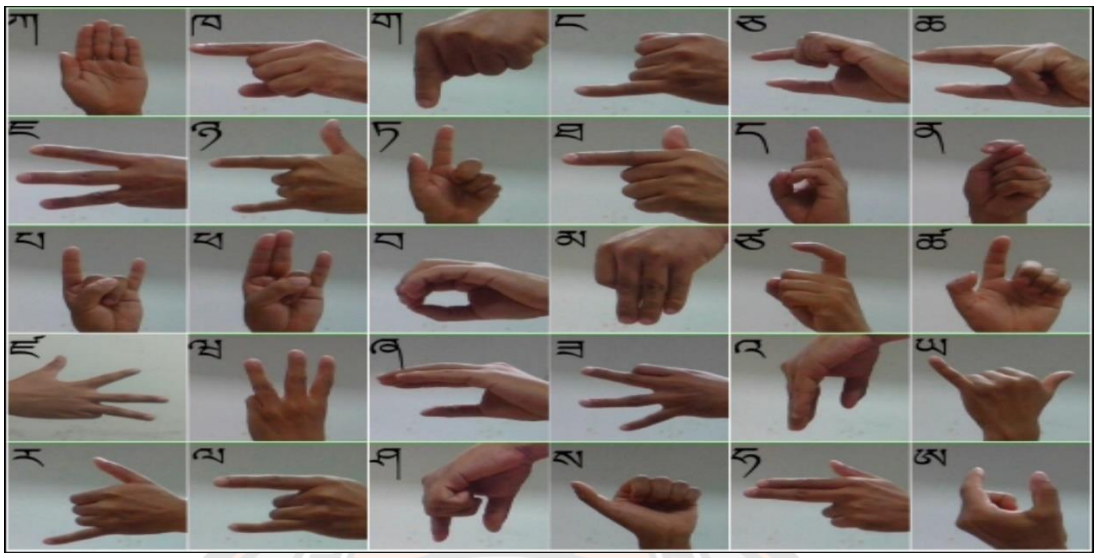


Figure 1 BSL Dzongkha hand alphabets

Source: Wangchuk et al. (2020a)

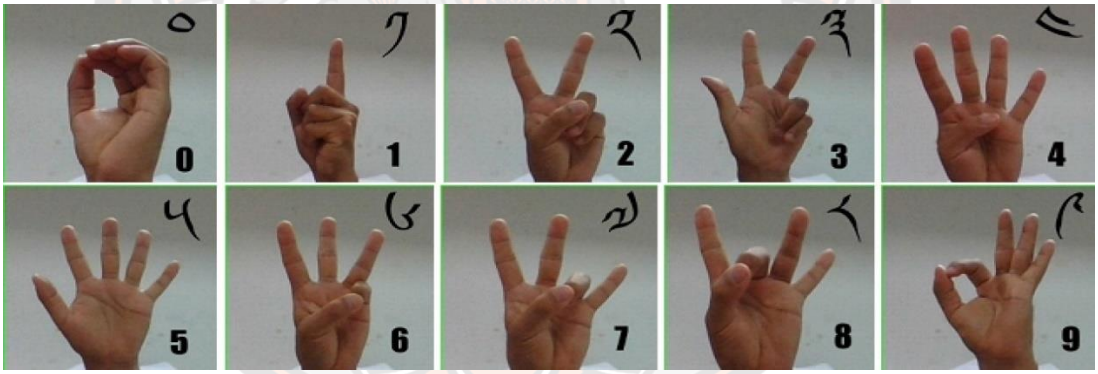


Figure 2 BSL Digits 0-9

Source: Wangchuk, Riyamongkol, and Waranusast (2020b)

Figure 3 BSL four vowels

Table 4 A single syllable formation by the addition of superscribed and subscribed alphabets to the root letter

Type of Joins	Joining Letters	Alphabets	Syllabus formation
Superscribed	ར	ཀ་ག་ང་ད་ཨ་ཀ་ད་ན་བ་མ་ཙ་ཛ།	ཀ་ཁ་ང་ཇ་ཉ་ཏ་ད་ན་ཐ་མ་ཙ་ཛ།
	ལ	ཀ་ག་ང་ཅ་ཨ་ཀ་ད་པ་བ་ཀ།	ཁ་ཁ་ཎ་ཐ་ཕ་ཉ་ཎ་ཕ་ཁ་ཁ།
	ས	ཀ་ག་ང་ཨ་ཀ་ད་ན་པ་བ་མ་ཙ།	ཁ་ཁ་ཚ་ཁ་ཉ་ཚ་ཐ་ཕ་ཁ་ཁ།
Subscribed	ཡ	ཀ་ཁ་ག་པ་ཕ་བ་མ།	ཀྱ་ཁྱ་ཀྱ་ཏྱ་ཐྱ་ཁྱ།
	ར	ཀ་ཁ་ག་ཀ་ཐ་ད་ན་ཐ་བ་མ་ཤ་ས་ཀ།	ཀྲ་ཁྲ་ཀྲ་ཐྲ་དྲ་ནྲ་ཐྲ་ཐྲ་ཁྲ་ཤྲ་སྲ་ཀྲ།
	ལ	ཀ་ག་བ་ཟ་ར་ས།	ཀླ་ཁླ་ཁླ་ཟླ་ཆླ་སླ།

There are six different ways to add alphabets as superscribed or subscribed with root letters to form a single syllable as shown in Table 4. The alphabets **ར, ལ,** and **ས** are used as superscribed to join on top and **ཡ, ར** and **ལ** as subscribed to join beneath the root letter. There are 12 different ways in which letter **ར** is added as superscribed to ཀ་ག་ང་ཇ་ཉ་ཏ་ད་ན་བ་མ་ཙ་ཛ alphabets to form various syllable root letter ཀ་ཁ་ང་ཇ་ཉ་ཏ་ད་ན་ཐ་མ་ཙ་ཛ as demonstrated in Figure 4.

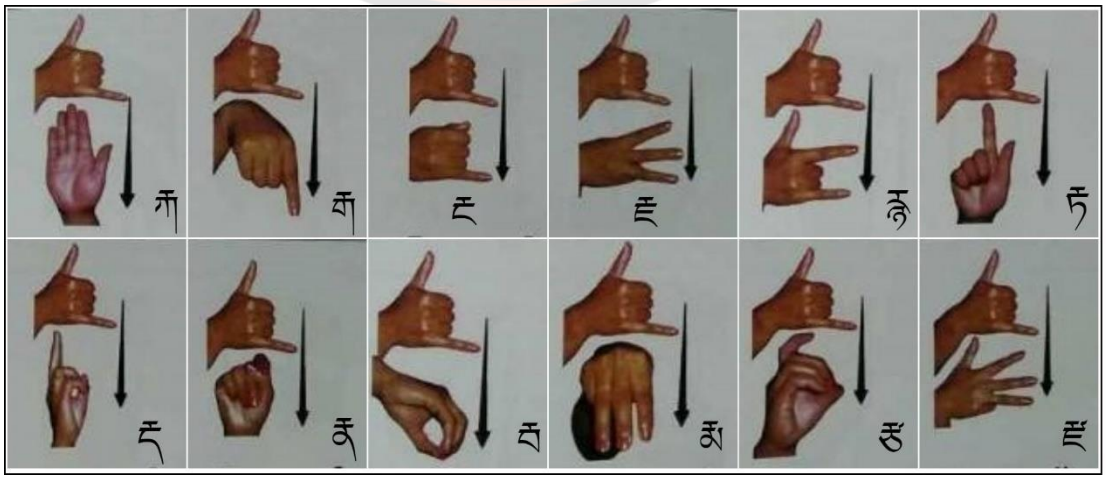


Figure 4 Rago Chunyi

Similarly, the letter ཤ is superscribed with 10 different alphabets ཀ་ཁ་ང་ཅ་ཉ་ཏ་ཐ་ད་ན་པ་བ་ཏ to form ཤྐ་ཤྑ་ཤྒ་ཤྔ་ཤྕ་ཤྙ་ཤྟ་ཤྠ་ཤྡ་ཤྣ as shown in Figure 5.

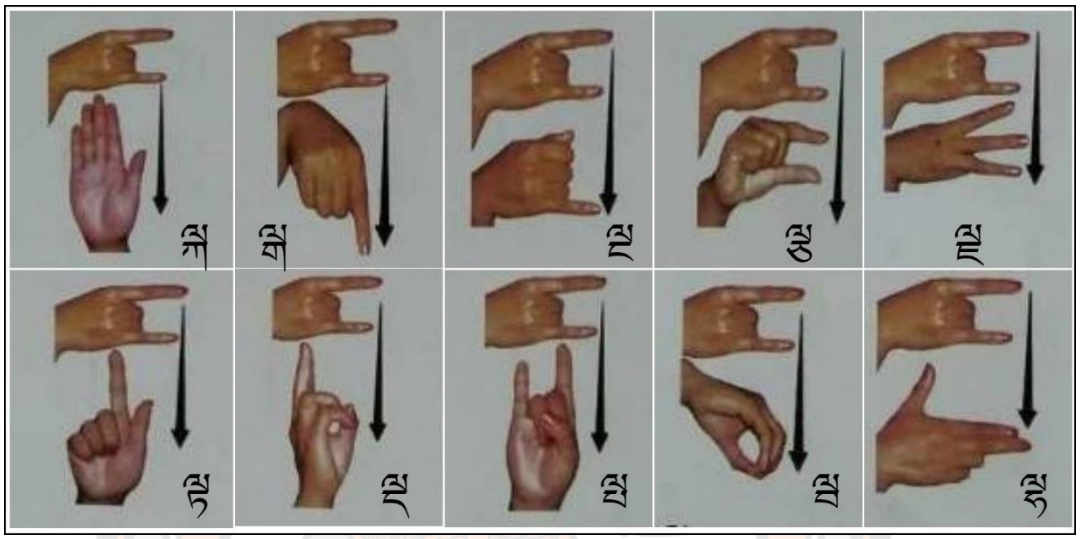


Figure 5 Lago Chu

The letter ས is superscribed with 11 numerous alphabets ཀ་ཁ་ང་ཅ་ཏ་ད་ན་པ་བ་མ་ཙ and form various syllable སྐ་སྒ་སྔ་སྙ་སྟ་སྡ་སྣ་སྤ་སྦ་སྨ་སྩ as illustrated in Figure 6.

Figure 6 Sago chuchi

Similar to superscribed syllables, there are three different subscribed combination. There are 7 different ways in which letter ཡ is added as subscribed to ཀ་ཁ་ག་ད་ཕ་བ་མ to form ཀྱ་ཁྱ་གྱ་ཟྱ་ཕྱ་བྱ་མྱ as demonstrated in Figure 7.

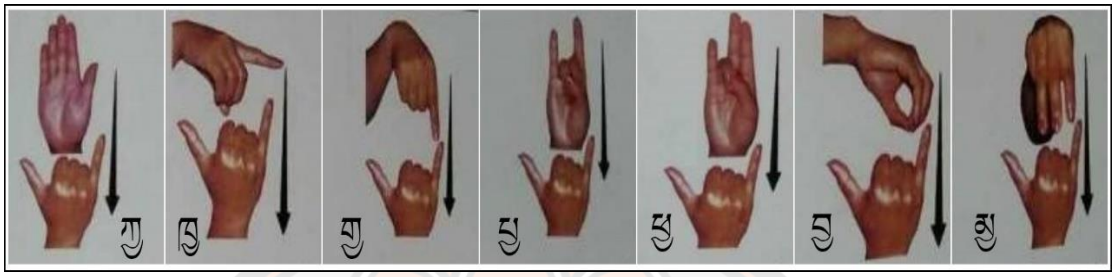


Figure 7 Yata dhuen

Similarly, there are 14 different ways in which letter ར is added as subscribed to ཀ་ཁ་ག་ཏ་ཐ་ད་ན་ཕ་བ་མ་ཤ་ས་ཧ to produce root letters ཀྲ་ཁྲ་གྲ་ཏྲ་ཐྲ་དྲ་ནྲ་པྲ་ཕྲ་བྲ་མྲ་ཤྲ་སྲ་ཧྲ as demonstrated in Figure 8.

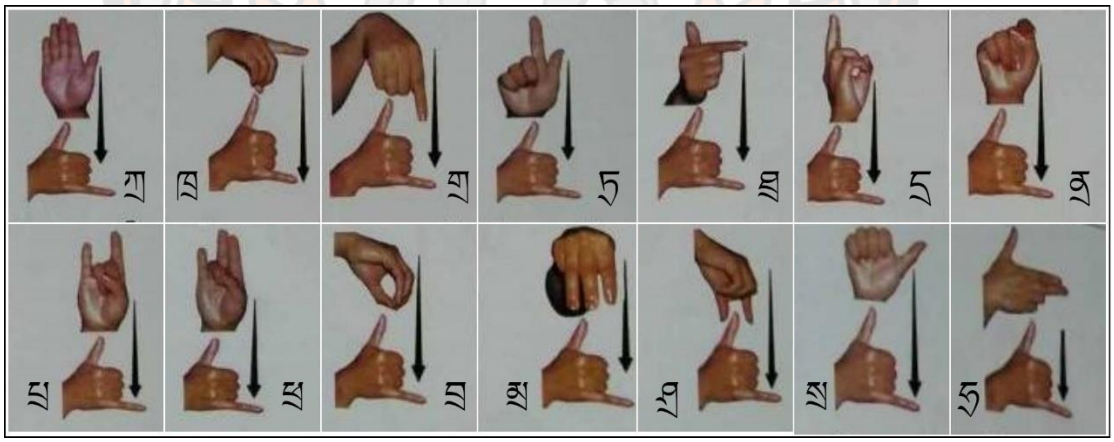


Figure 8 Rata chuzhi

There are 6 different ways in which letter ལ is combined as subscribed with alphabets ཀ་གཱ་ད་ཟ་ར་ས to form the root letters ཀླ་གླ་ཟླ་ཟླ་རླ་སླ as demonstrated in Figure 9.

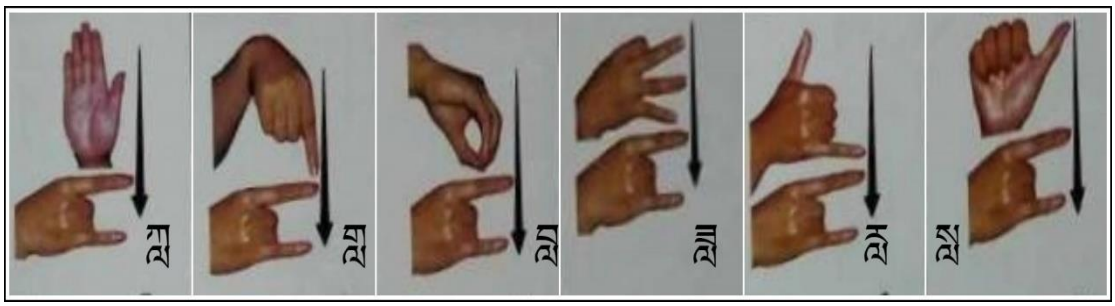


Figure 9 Lata dru

Teaching and learning environment would be livelier by using technology in classrooms. Gestures can be digitized visually on the screen with the written text while teaching signs in the classroom. Apart from teaching and learning in the classroom, it will also address the communication gap within society. For instance, the English language is translated into several languages and vice-versa. Tremendous works have been done for Natural Language Processing such as Text to Speech, Speech to Text, Spell Checker, Optical Character Recognition, etc. Similarly, these techniques can be applied to sign languages as well. The popular sign languages such as American and Australian Sign Languages have implemented technology to recognize signs and Bhutan is no exception now. There is a communication gap between the deaf community and the public in Bhutan. The possibility of addressing this issue can be achieved with the help of technologies. BSL can be translated into text or speech in real-time.

This study will be beneficial to the deaf community and the general public in the future. The study aims to explore the possibility of integrating technology and introduce the machine learning foundation with the BSL. The proposed system translates the BSL Dzongkha hand-shaped sign alphabets and digits to text in real-time. This is the basic building block of Natural Language Processing (NLP) for the development of advanced systems such as signs to text to speech and vise-Versa. The research in BSL is gaining momentum with the government and Ministry of Education prioritizing the Deaf community. According to (Rinzin, 2019b), former president of the World Federation of the Deaf, Collin James Allen, said that Bhutan is giving importance to the sign language by developing BSL and must be given equal significance as the national language.

However, no one has attempted to use Machine Learning techniques for BSL. The integration of Artificial Intelligence with the BSL would address the issue of the communication gap in society. The deaf community is not able to communicate with the public and at times with their parents as well. (Rinzin, 2019a) discussed that Wangsel Institute for the Deaf provides sign language workshop to the parents for one hour on Fridays. To include the wider section of the society, it is time to integrate and harness the power of technology in sign language.

1.2 Purposes of the Study

The main objective of this study is to develop a machine learning model for Bhutanese Sign Language hand-shaped alphabets and digits detection and recognition. This research discusses the following purposes:

1. Bhutanese Sign Language hand-shaped alphabets and digits (BSL) dataset preparation.
2. Selection and modification of Machine learning models to train with the BSL dataset.
3. Identify suitable algorithm for real-time detection, recognition, and translation of BSL Hand-shaped alphanumeric into Dzongkha text using the trained model with webcam.

1.3 Statement of the Problems

The emerging communication problem between the deaf and the public is a concern for both parents and the government of Bhutan. One of the priorities of the government is education and it is provided free. The number of schools is upgraded to higher secondary schools and colleges. Bhutan also drafted the national policy for disability and established a school for the deaf. The government urges people to study BSL but learning sign language is difficult. The deaf institute in collaboration with the Ministry of Education is prioritizing the issue by conducting BSL research and workshops.

According to (Rinzin, 2019b), the development of a standard BSL and the documentation of sign language tasks have been progressing very well for the last three years by the research team comprising of a consultant, two teachers, and eight deaf

adults at the Wangsel Institute. The author described the research environment set up with cameras, lightings, tripods, and a green board in the classroom for written and video documentation of the BSL. However, no one has attempted the develop a BSL recognition system. In the era of technology, the communication gap can be resolved by translating sign language into either text or speech. Having such a system would allow deaf people to learn visually and interact with the public more easily.

In response to this problem, the study proposes to study the feasibility of technology integration with BSL for designing a basic system that can detect, recognize, and translate Dzongkha hand-shaped alphabets and digits to text in real-time. The study plans to investigate several machine learning algorithms that are suitable to implement in BSL translation, choose the best model for BSL, and lay the foundation for artificial intelligence in BSL.

1.4 Scope of the Study

The scope of the study is to develop the machine learning model for Bhutanese Sign Language which can detect, recognize, and translate both hand-shaped Dzongkha alphabets and digits in real-time using the webcam. The aspects of the research look into the preparation of the BSL dataset, testing different algorithms with this dataset, and identification of the best machine learning model that is suitable for the BSL.

However, the study does not cover BSL word detection and recognition. The word recognition requires motion captures and video processing approaches that require intensive computing resources and experts to record dynamic sign videos dataset.

Key Words

Bhutanese Sign Language, BSL Dataset, Convolutional Neural Network, Visual Geometry Group, Image augmentation.

CHAPTER II
LITERATURE REVIEW

2.1 Introduction

The advancement of technology has been helping humanity for decades and is improving every day. Machine learning algorithms have the ability to learn from the data without requiring explicit human intervention. The learning improves automatically with the experiences and makes a better decision. Different algorithms are developed by the emulation of body sensory responses such as hearing, speech, and vision (Khan, Sohail, Zahoora, & Qureshi, 2019). Computer vision and image processing are focusing on the vision. The various techniques are implemented to recognize gestures, image segmentation, classification, detection, object recognition, and video processing.

The development of computer vision applications is challenging. This challenge has been addressed by the introduction of a new neural network architecture called the Convolutional neural network (CNN). In the late 1980s, Y LeCun et al. proposed CNN and was implemented for image recognition but it has limitations such as lack of computing power and the scarcity of labeled data. The CNN remained dormant but the researchers used Support Vector Machine (SVM), Principle Component Analysis (PCA), Hu's moment, K-Nearest Neighbours (KNN), and Hidden Markov Model (HMM) to study computer vision. However, CNN revived in 2012 with the usage of the Graphics Processing Unit (GPU), Tensor Processing Unit (TPU), and the availability of huge labeled datasets (Rawat & Wang, 2017). According to Krizhevsky, Sutskever, and Hinton (2012), deep learning, by increasing the number of layers of the networks, enhances algorithms with state-of-the-art performance on the ImageNet dataset. CNN is considered one of the best algorithms for understanding image content.

2.2 Early Gesture Recognition Techniques

The researchers are using different algorithms for their studies. The choice of the algorithms is dependent on the nature of the studies, the availability of the system requirements, and the dataset. The early gesture recognition methods used by the early

researchers are based on the four utmost important phases: Image Acquisition, Segmentation, Feature Extraction, and Gesture Recognition.

2.2.1 Image Acquisition

Sign language recognition uses sensor-based and vision-based approaches to collect data and develop sign recognition systems. The latter approach has the potential to communicate naturally between the human and the computer by supporting intuitive and efficient interaction with powerful tools (Murthy & Jadon, 2009). There are lots of studies conducted on sensor-based approaches such as Accelerometer, Wi-Fi, Radar, and wearable sensors .), implemented Data Glove for hand gesture recognition using a learning vector quantization classifier. They have collected 7800 right-hand gestures from people of different gender and physique. The proposed model obtained 99.31% accuracy on the testing dataset. The development of the Kinect sensor gave researchers renewed opportunities for Human-Computer Interaction (HCI) studies. Ren et al. explained Finger-Earth Mover's Distance, a novel distance metric, to handle the noisy hand shapes. The Kinect sensor was used to collect data for robust part-based hand gesture recognition. They have obtained 1000 gestures for the first 10 natural numbers from 10 different people. The accuracy of the model was 93.20%. The gestures are recognized with the measurement of electrical pulses. According to Camastra and De Felice (2013), the car was controlled by gestures performed with one hand. The authors developed an electromyogram (EMG) based real-time bio-signal interface for hand gesture recognition. The system was evaluated by 30 subjects in 40 test sessions using an RC Car. The proposed model achieved 94% accuracy. The development of different sensor technologies facilitated researchers to develop HCI systems, however, the disadvantages of having to wear sensors continuously are not feasible in real-time applications and the sensors are expensive (Jonghwa Kim, Mastnik, & André, 2008; Verma, Srivastava, & Kumar, 2015). Therefore, researchers are shifting their focus and concentration on vision-based approaches.

The images, in vision-based, are collected using cameras, smartphones, Leap Motion Controller (LMC), Microsoft Kinect devices, and body marker gloves. The images are extracted from videos that are recorded either by camera or smartphones

and used for image recognition. The prior studies on gesture recognition used LMC (Nikam & Ambekar, 2016). The LMC device can generate 200 frames per second for the hand movement (Koul, Patil, Nandurkar, & Patil, 2016). The low-cost Microsoft Kinect provides synchronized color and depth images. It is initially used for the Xbox gaming console but later extended for computer vision applications development (Mohandes, Aliyu, & Deriche, 2014). J. Han, Shao, Xu, and Shotton (2013), used LMC for Arabic sign language recognition. They have collected 2800 frames from a person. The signer posed 10 samples for every 28 Arabic alphabets. The LMC provided 23 values, but only 12 frames were chosen for the representation. The authors combined the proposed method with Nave Bayes Classifier and Multilayer Perceptron (MLP). The proposed method with MPL obtained more than 99% accuracy. The Microsoft Kinect generated 3D depth information from hand motions that were used to recognize hand gestures by applying a hierarchical conditional random field (Mohandes et al., 2014). The proposed model yielded a success rate of 90.40%.

2.2.2 Image Segmentation

Image segmentation is a vital phase of image processing and pattern recognition. It partitions an image into several segments based on some criteria to locate boundaries and objects (Yang, 2015). The partitions provide small but meaningful areas in the image for information extraction. According to Cheng, Jiang, Sun, and Wang (2001), there are four techniques for image segmentation: thresholding, boundary-based, region-based, and hybrid techniques. Thresholding is based on the intensity and variance of the pixel. It is a tool used for separating objects from the backgrounds (Nimbarte & Mushrif, 2010). Sezgin and Sankur (2004), explained that the threshold-based image segmentation uses the histogram of the image to detect thresholds. Naidu, Kumar, and Chiranjeevi (2018), grouped various thresholding methods according to information exploitation such as histogram shape-based, clustering-based, entropy-based, object attributed-based, spatial, and local methods. The histogram shape-based method has different thresholding techniques namely convex hull, peak-and-valley, and shape-modeling. Similarly, clustering-based thresholding types are Iterative, Clustering, Minimum error, and Fuzzy clustering thresholding. Different types of entropy-based thresholding are Entropic, Cross-

entropic, and Fuzzy entropic thresholding. Moment preserving, Edge field matching, Fuzzy similarity, Topological stable-state, maximum information, and Enhancement of Fuzzy compactness are various types of object attributed-based thresholding. Spatial thresholding methods are Cooccurrence, Higher-order entropy, Random sets, 2-D fuzzy partitioning. Similarly, different types of local adaptive thresholding are Local variance, Local contrast, Center-surround schemes, Surface-fitting, and Kriging thresholding.

Boundary-based thresholding is based on the sudden change in pixel properties such as intensity, color, and texture between different regions in an image, but region-based has similar neighbouring pixels value in the same region (Sezgin & Sankur, 2004). Shih and Cheng (2005), explained that the edge-based finds edges in the image or active contours for segmentation. The region-based segmentation depends on the thresholding values and the edges of the input images. The optimum thresholding is achieved by a combination of these thresholding methods. The combination of boundary detection and region growth results in better segmentation (Iannizzotto & Vita, 2000).

2.2.3 Feature Extraction

Image features are of paramount importance for the analysis. The features can be contour, shape, rotation, angle, coordinates, movements, and background data of an image (Deng & Manjunath, 2001). Detection of these features plays a vital role in pattern recognition. According to Mingqiang, Kidiyo, and Joseph (2008), the main aim of the feature extraction is to get the most relevant information from the data and represent that into designated classes. The widely used feature extraction methods are Contour profiles, Deformable templates, Fourier descriptors, Geometric moment invariants, Gradient features, and Gabor features, Graph description, Unitary image transform, Template matching, Projection histograms, Zoning, Zernike moments, Spline curve approximation (Kumar & Bhatia, 2014).

In addition, one of the prior extraction methods used was the Scale Invariant Feature Transform (SIFT) (Cheung & Hamarneh, 2009; Kumar & Bhatia, 2014; Lindeberg, 2012). There were limitations for the prior methodologies such as hand detection because of low resolution and different light intensity. The development

of new and hybrid algorithms addressed these limitations. The neural network algorithms are increasingly used for pattern recognition. The multiscale convolutional neural network solves prior hand detection problem (Lowe, 2004). However, Yan, Xia, Smith, Lu, and Zhang (2017) explained that SIFT was robust against rotation, translation, or scaling variation, and best for the collection of large local feature vectors.

2.2.4 Gesture Recognition

The final phase of image processing is gesture recognition. The decision of class detection and recognition, after analysis of features, is made by assigning a label. Gurjal and Kunnur (2012) described two types of recognition: recognition using extracted features and machine learning classifiers. The template matching is one of the methods which uses extracted features to recognize gestures (Alvi et al., 2004; Ariesta, Wiryana, & Kusuma, 2018; Yun, Lifeng, & Shujun, 2012). The geometric moment also uses features to recognize gestures. Mahbub, Imtiaz, Roy, Rahman, and Ahad (2013), used skin color as the feature of the hand to detect gesture using geometric moments. The graphs are used for hand gesture recognition as well. The graphs are formed based on a group of template images. The features of the images are assigned to the nodes of the graph for recognition (Priyal & Bora, 2013). Fourier descriptors and Hu moments are implemented for hand gesture recognition based on features extracted from the hand region. Li and Wachs (2014) compared Four descriptors and Hu moments for hand posture recognition.

In recent times, the trend for image processing and pattern recognition are using machine learning classifiers such as Convolutional Neural Network and Hidden Markov Model for the development of applications. The detail about these algorithms and their hybrids would be explained in the following sections.

2.3 Sign Language Recognition Algorithms

The system hardware has been improving daily, and the market availability encourages researchers to pursue continuous studies on images. The evolution of sign language recognition algorithms has been tremendous. The progress has been started from supervised learning to unsupervised recognition. The learning in supervised classification is based on labelled training data but unsupervised learning does not need

labelling on training data (Conseil, Bourennane, & Martin, 2007). The algorithms such as SVM, KNN, PCA, HMM, ANN, and CNN are used in Computer Vision for detection and recognition.

2.3.1 Supervised Classification Algorithms

The supervised algorithm teaches the system to detect patterns from the input data and predicts unseen future data. The labels of the training dataset are assigned properly to train the mapping function. The input is fed to the mapping function to generate the desired output. Some of the supervised algorithms used for sign language recognition are SVM, KNN, and Artificial Neural Network.

Sathya and Abraham (2013) performed dynamic hand gesture recognition of Indian Sign Language by combining the SVM and the Hu's Moments. The video frames of the gestures were converted to HSV color space and segmented based on skin pixels. The proposed model obtained 97.50% accuracy. Similarly, Raheja, Mishra, and Chaudhary (2016) described hand gestures tracking of American Sign Language (ASL) using SVM. The static gestures of six letters (A, W, H, O, L, I) were trained by providing their coordinates. The classifier gave an accuracy of 92.13%. Furthermore, Bag-of-Features and multiple SVM were used for real-time hand gesture detection and recognition (Sinith et al., 2012). The authors presented a novel system for interaction with videogame via hand gestures. The classification accuracy of the proposed model was 96.23%.

The adaptive network-based fuzzy inference system and KNN were implemented for ASL hand gesture recognition (Dardas & Georganas, 2011). The authors separated 26 gestures into 3 groups: fingers gripped, fingers facing upward, and fingers facing sideways. The classification of these gestures by the KNN classifier obtained 80.77% accuracy with 10 epochs. Similarly, Mufarroha and Utaminingrum (2017), classified Indian Sign Language gestures based on the correlated neighbour. The features of single-handed and double-handed gestures are extracted using Histogram of Oriented Gradients (HOG) and Scale Invariant Feature Transform (SIFT). They have combined extracted feature matrices into a single matrix and trained using KNN. The accuracy was 97.50% for single-handed gestures and 91.11% for double-handed gestures respectively. Bengali Sign Language recognition system in real-time

was presented using the KNN classifier (Gupta, Shukla, & Mittal, 2016). They have collected 3600 images from 10 signers for 6 vowels and 30 alphabets. The recognition accuracies for vowels and alphabets were 98.17% and 94.75% respectively.

Artificial Neural Network (ANN) processes information similar to biological neurons. It has three prominent steps: input layers, hidden layers, and output layers. The input neurons receive data and process it to hidden layers. The weights are calculated and passed to the activation function in the hidden layer followed by giving expected output. Rahaman, Jasim, Ali, and Hasanuzzaman (2014), implemented a single feed-forward ANN to detect New Zealand sign language. They used 7392 gesture signals to train 13 gestures with 45 inputs and 14 outputs using two hidden layers. The accuracy of 96.02% was obtained. Similarly, ANN was used for recognition of accelerometer and electromyography-based hand gestures, recognize virtual reality driving with hand gestures, and hand directions using gestures (Ahsan, Ibrahimy, & Khalifa, 2011; Akmeliawati et al., 2009; R. Xie & Cao, 2016; Xu, 2006).

2.3.2 Unsupervised Clustering Algorithms

The unsupervised algorithms are similar to supervised algorithms but formal does not require labelled data. The output is predicted based on learning features from the input data and no prior set of categories is needed. Some of the unsupervised clustering algorithms are K-mean, PCA, and HMM.

According to Murthy and Jadon (2010), K-mean clustering is commonly used for gesture recognition. Cheok, Omar, and Jaward (2019), described the recognition of hand gestures using K-mean clustering based on features of the hand shape such as orientation, centroid, and finger status (raised or folded). The author used 450 images and achieved a training accuracy of 94%. Similarly, Panwar (2012), applied K-means for object classification and localization based on features extracted from the images.

There are limitations to the algorithms. These limitations can be solved by combining different algorithms. The hybrids algorithms perform better and can be applied in the development of various applications involving different datasets. The dynamic gestures are recognized with the motion trajectory of a single hand using HMM (Schmitt & McCoy, 2011). They have recorded 30 video gestures from the

number 0 to 9. The accuracy of 200 training and 98 testing sequences achieved 94.29%. Similarly, Elmezain, Al-Hamadi, Appenrodt, and Michaelis (2009) described a real-time tracking method and HMM for hand gesture recognition. The proposed model tracked the moving hand and extracted the hand region. The spatial and temporal features are characterized by using Fourier Descriptor. They have collected 60 images from 20 people and 1200 images are used for training. The HMM classifier obtained 98.50% accuracy. Furthermore, Chen, Fu, and Huang (2003) used PCA and HMM to recognize gestures. The PCA described spatial shape variations and HMM portrayed temporal shape variations. It has been found that the hybrid algorithm recognized 18 different continuous gestures.

2.3.3 Convolutional Neural Networks

The active research in images and gesture recognition has been promoting different models' architecture design and the contributing evolution of existing algorithms. CNN is considered one of the best algorithms used for computer vision applications (C.-l. Huang & Jeng, 2001). However, the initial phase of the inception was not popular due to various limitations. Khan et al. (1989) implemented CNN but required more computing resources and humongous labeled data. Therefore, it remained dormant. However, in the late 2000s, researchers used the Graphics Processing Unit (GPU) and parallel processing. In 2012, CNN revived with the implementation of GPU, Tensor Processing Unit (TPU) and availability of huge labelled data (Y LeCun et al.). The deep CNN on the ImageNet dataset outperforms previous performances (Rawat & Wang, 2017).

The revival of CNN garnered the development of computer vision applications based on object detection, image recognition, classification, segmentation, and robotic vision. Frontrunner companies such as Google, Facebook, and Microsoft are using computer vision for application developments. Some of the machine vision applications in the market are IBM Watson, Google Cloud Vision, AWS Rekognition, Microsoft Computer vision, Kairos, EmoVu, ImageVision, and Animetrics. The Tech companies motivate and encourage researchers to take part in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which is conducted annually. This competition evaluates object detection and image classification on a large scale.

The CNN based architectures won series of ILSVRC competitions from 2012 onwards. The researchers have been taking inspiration from the LeNet, CNN-based architecture, to design their model (Krizhevsky et al., 2012). Yann LeCun et al. (1989) implemented a network similar to LeNet but was deeper with more convolutional layers and GPU accelerated hardware. The network was called AlexNet and won the 2012 ILSVRC. The top 5 test error of AlexNet was 15.3% compared to 26.2% of the runner-up which brought limelight and popularized CNN in Computer Vision. Krizhevsky et al. (2012), won ILSVRC 2013 competition and the network was known as ZFNet. The architecture of the ZFNet was the improvement of AlexNet by fine-tuning hyperparameters. The authors observed better performance compared to AlexNet on the ImageNet benchmark. The winner of the ILSVRC 2014 was GoogLeNet (Zeiler & Fergus, 2014). The network reduced the number of parameters to 4 million from 60 million with the development of an Inception Module. There are different versions of GoogLeNet (Inception: V2, V3, V4, and Inception-ResNet) (Szegedy et al., 2015). The runner-up of ILSVRC 2014 was VGGNet (Ariesta et al., 2018). The network was made of 16 convolutional layers. The VGGNet demonstrated that the depth of the network was critical for better accuracy. In 2015, ResNet was the winner of ILSVRC (Simonyan & Zisserman, 2014). The network uses skip connections and batch normalizations. The Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) are used for temporal sequence analysis.

The dynamic hand gesture recognition needs both spatial and temporal information of the image sequences (He, Zhang, Ren, & Sun, 2016). CNN analyses the spatial features of the images and RNN for the temporal relation of the image sequences. However, long sequences cannot be handled by RNN. It suffers from short term memory due to vanishing gradient. Therefore, LSTM is used for learning long term dependencies. Shin and Sung (2016) described the detection of key points in the human body for sign language recognition. They have extracted key points from the face, hand, and body parts. These features are trained using RNN and achieved the classification accuracy of 89.5% for 100 sentences. Similarly, Ko, Son, and Jung (2018) recognized large-scale continuous gestures using two-stream RNN (2S-RNN). The continuous gesture is segmented into individual separate gestures based on hand position using Faster R-CNN. Then each isolated gesture is detected and recognized

using 2S-RNN based on fuse multi-modal features such as RGB and depth channels. Furthermore, many researchers have been working on dynamic sign language recognition using a hybrid of these algorithms (Chai, Liu, Yin, Liu, & Chen, 2016; Edel & Köppe, 2016; Hu et al., 2018; Lefebvre, Berlemont, Mamalet, & Garcia, 2013).

2.4 Recent Trends in Sign Language Recognition

In recent years, the studies are based on dynamic action recognition using videos. The evolution of algorithms and ability to extract spatiotemporal information from the videos allowed researchers (Haseeb & Parasuraman, 2017; Ji, Xu, Yang, & Yu, 2012; Kar, Rai, Sikka, & Sharma, 2017) to move from static to dynamic gestures detection and recognition. However, video processing required huge resources than static images.

Baccouche, Mamalet, Wolf, Garcia, and Baskurt (2011) described trajectory-based Persian sign language recognition using HMM. They used a white glove to record 1200 videos from 12 persons for 20 signs. The time-varying features trajectories of hands were extracted. The proposed model obtained 97.48% average accuracy. Similarly, the novel deep learning-based architecture using Single Shot Detector (SSD), 2D and 3D CNN, and LSTM were proposed by Azar and Seyedarabi (2020). The model was trained on 10,000 videos of 100 Persian sign words and achieved an accuracy of 99.90%. Furthermore, two-stream attention and LSTM architecture are used for human activity recognition. Rastgoo, Kiani, and Escalera (2020) presented two-stream attention-based LSTM by training on different datasets. The UCF11 dataset has 1600 videos, UCFSports consist of different sports actions featured on BBC and ESPN television channels, and Joint-annotated Human Motion Database (JHMDB) has 923 videos. The proposed method obtained an accuracy of 96.90% for UCF11, 98.60% for UCFSports, and 76.30% for JHMDB.

CHAPTER III
RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, the detailed modification of the neural network architecture for the Bhutanese Sign Language alphanumeric detection and recognition is discussed. There are four prominent gesture recognition phases such as Data Acquisition, Data Pre-processing, Feature Extraction, and Gesture Recognition as shown in Figure 10. Each phase is significant for robust gesture detection and prediction systems. The system requirement is presented first followed by the overall system flowchart. The following sections discuss each phase with a precise block diagram.

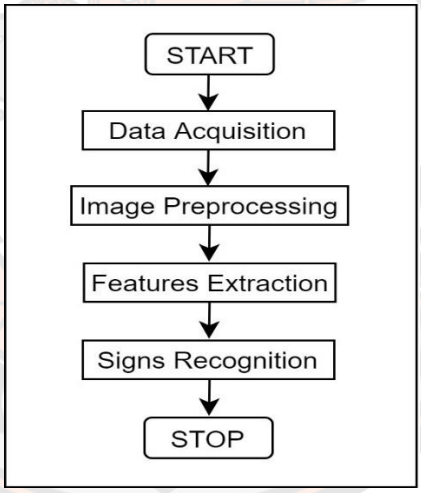


Figure 10 Four phases of sign recognition

3.2 System requirements

The system requirements of the BSL alphanumeric recognition is illustrated in Table 5. In this research, both hardware and software were used. The hardware used was a laptop and camera, whereas software such as PhotoScape X, Visual Studio (VS) Code, high level (TensorFlow), and low level (Keras) training APIs were used. Python 3.7 is used as the programming language.

The model was trained using a free cloud service called Google Colab. It is the virtual machine operated on the NVIDIA Tesla K80 GPU using the Intel Xeon processor. Colab provides 12 hours of free uninterrupted training online.

Table 5 System requirements

System	Name	Specs/version
Hardware	Laptop	Dell, 8GB RAM, Intel Core i7-7500U CPU @ 2.70GHz 2.90GHz
Cameras	Canon/Phone	Canon M50/Redmi Note 4
Programming	Python	3.7
Editor	VS Code	1.45.1
Backend	TensorFlow	2.x
ML Library	Keras	2.3.0
Photo Editor	PhotoScape X	4.0.2.0
Cloud Service	Google Colab	online
Drawing Software	DrawIO	Online

3.3 System overview

The overview of the proposed system is shown in Figure 11. The digital camera and smartphone were used to capture images and videos. The number of frames was extracted from the recorded videos. In addition, images were further augmented to add variations to the BSL dataset. The augmentation generates more images from a single image with variations such as the addition of pixels, colors, top and black hat, and so on. Different image classification algorithms were evaluated on BSL data and the suitable model was selected for further hyperparameter fine-tuning. The model was trained until the desired accuracy was obtained. During the training stage, prominent features of the images were extracted and accordingly classified every image into one of the classes.

The model was saved and deployed with the laptop using the VS Code, OpenCV, and TensorFlow as the backend. The webcam was used to feed real-time BSL alphanumeric data to detect and predict signs with the deployed model. In the following sections, data acquisition, pre-processing, feature extraction, and model deployment are discussed.

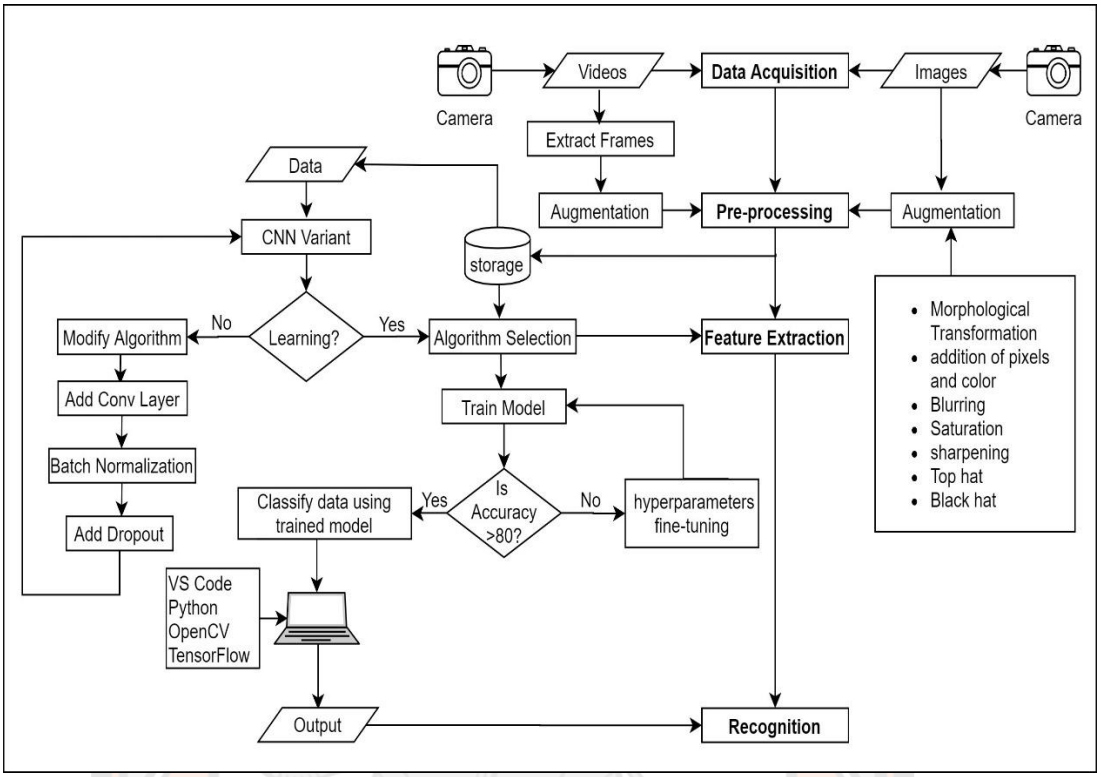


Figure 11 System overview

3.3.1 Data acquisition

Data is the most significant part of Machine Learning and Data Analytics. Artificial Intelligence (AI) systems are consistently learning, inferring information, and making decisions to help humanity. Data mining and analytics further enhance AI systems. A careful and substantial selection of data can significantly improve the generalization of machine learning performances (Dai, Liu, & Lai, 2020) for information extraction, prediction, and pattern recognition. According to Attenberg, Melville, Provost, and Saar-Tsechansky (2011), the analysis of data allows knowledge discovery and support decision making. There are dataset repositories online such as Kaggle, UCI Machine Learning Repository, Github, Google Public Datasets, and many more. However, the BSL dataset is not available for the study.

The BSL dataset was created from international students based at Naresuan University. Figure 12 shows the detailed procedure used for data collection. In the initial stage, the International Student Office was contacted for international

student information. The volunteers were requested from different countries as listed in Table 6. The images and videos dataset were collected.

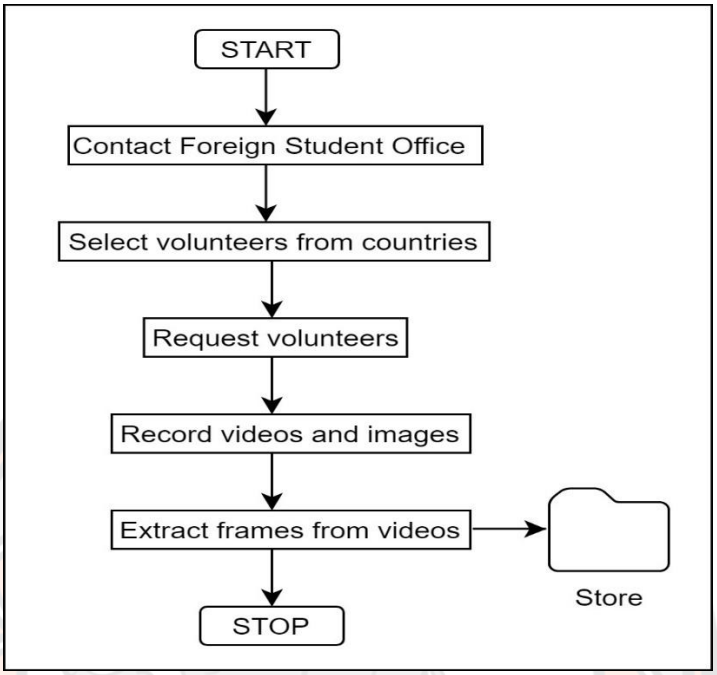


Figure 12 Data acquisition

Table 6 Volunteer signers from different countries

Country	Male	Female	Total
Bhutan	14	3	17
Burundi	1	0	1
Cambodia	7	1	8
Nepal	2	0	2
Thailand	4	8	12
Total	**28**	**12**	**40**

Source: Ge, Song, Ding, and Huang (2017)

The videos were recorded ranging from 18-25 seconds for every 40 alphanumeric classes from 40 volunteers. The images and videos were recorded with different angles, positions, various backgrounds, camera distances, and lighting conditions to add variations to the dataset. Therefore, 1200 videos from 30 actors with 40 classes were recorded as shown in Figure 13. The frames were extracted from videos at the rate of 10 fps. The images were pre-processed by augmentation techniques.

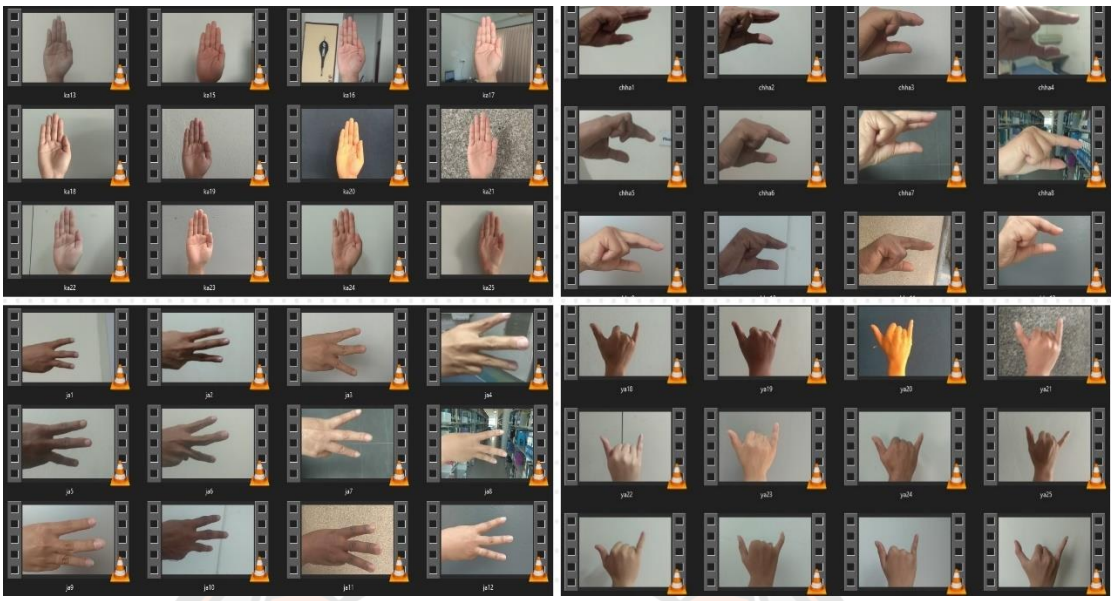


Figure 13 Sample of BSL videos data

3.3.2 Data Preprocessing

Data preprocessing is an important phase of machine learning. It transforms raw data into a machine-understandable format. Usually, real-world data are inconsistent, noisy, incomplete, and in different formats (Wangchuk et al., 2020a). Figure 14 demonstrates the BSL data preprocessing steps. The images were both manually and programmatically pre-processed using a photo editor called PhotoScape X and python respectively. The images were resized to 200x200 pixels and horizontally flipped 50% of the images. After rescaling, 40 images from each class were randomly selected for the augmentation.

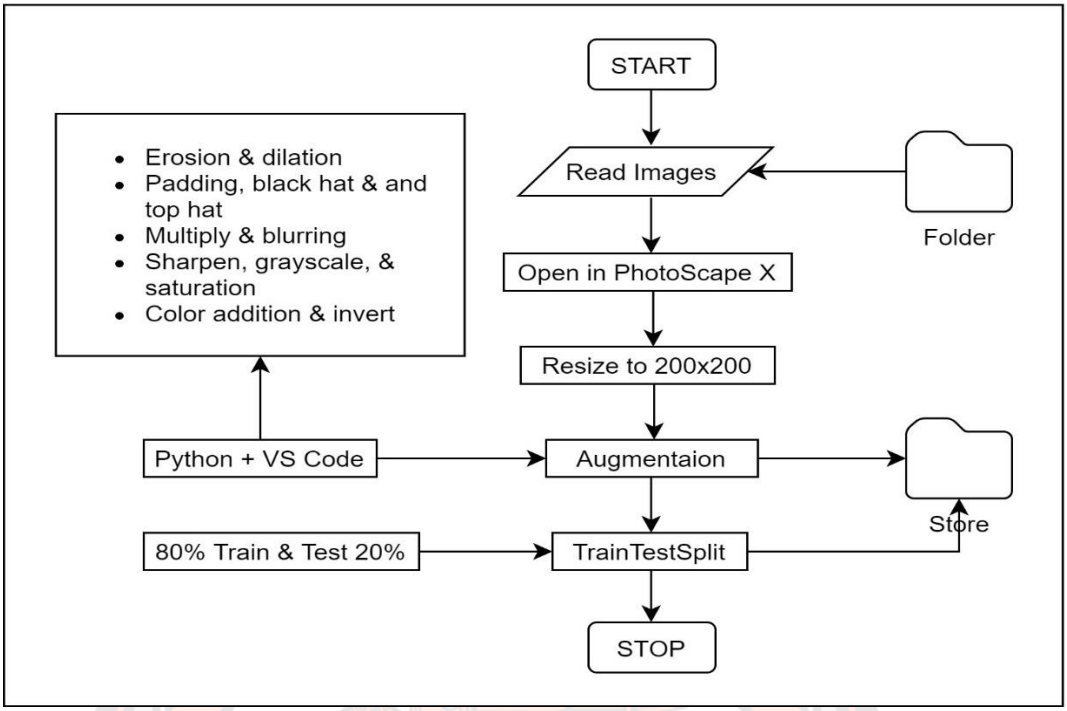


Figure 14 Data pre-processing steps

The main purpose of augmentation is to increase images and add variations to the dataset. The augmentation techniques used were erosion and dilation, padding, black and top hat, blurring, sharpening, color and light addition, saturation, rescaling, multiplying pixels, flipping, and inverting. The augmentation on a single image created 50 images by using the aforementioned methods as shown in Figure 15. Therefore, 2000 (40 classes x 50 images) augmented images were generated after performing augmentation.

The final BSL image dataset was prepared with 1500 images per class that comprised of 1000 randomly selected images from each class and 500 augmented images. Therefore, the BSL dataset consisted of 60,000 images from 40 alphanumeric classes. The dataset was partitioned into 70% training and 30% validation sets. These sets were uploaded in Google Drive and using Google's Colab, the model was trained. The feature extraction and model training are explained in the next section.

Figure 15 Augmented Images

3.3.3 Features Extraction and Classification

3.3.3.1 Introduction

In image processing, features of the image play important roles in classification and recognition tasks. According to Clark and Niblett (1987), the main purpose of feature extraction is to extract the most relevant information from the image and represent it in lower-dimensional space. The authors described two types of feature extractions: Local and Global Features. Local features are based on geometry such as concave/convex, endpoints, branches, joints, and shapes. However, topological features are based on lines, edges, curves, connectivity, and the number of holes. These features from the images are extracted using deep learning algorithms such as Convolutional Neural Network (CNN), Visual Geometry Group (VGG), ResNet, and LeNet-5. The features are also extracted by using supervised machine learning algorithms such as SVM, KNN, and logistic regression. In the following sections, aforementioned algorithms are discussed in detail.

3.3.3.2 Convolutional Neural Network

CNN is one of the deep learning algorithms used for image content analysis. It captures both spatial and temporal dependencies of the images using the relevant filters. Deep CNN consists of two phases: feedforward and

backpropagation. In the feed-forward phase, all the data are feed from the input layer to the output layer and the loss is calculated. The loss/error needs to be reduced for better prediction accuracy. Therefore, the loss is minimized with back-propagation by updating the bias and weight (Kumar & Bhatia, 2014) using different optimizers. The detailed discussion is described in the following sections.

3.3.3.2.1 Feedforward Neural Network

Artificial Neural Network (ANN) is inspired by the human brain. Tripathy and Anuradha (2017) presented a computational ANN model that was capable of learning, recognizing, and solving complex problems. According to McCulloch and Pitts (1943), Feedforward neural network (FNN) is based on ANN and capable of speech and signal processing, pattern recognition, clustering, and classification. The architecture of FNN comprises several neurons as input layers, hidden layers, and output layers as shown in Figure 16. The neurons (processing unit) have connections (weights) with each layer and bias is added in each neuron in the hidden nodes.

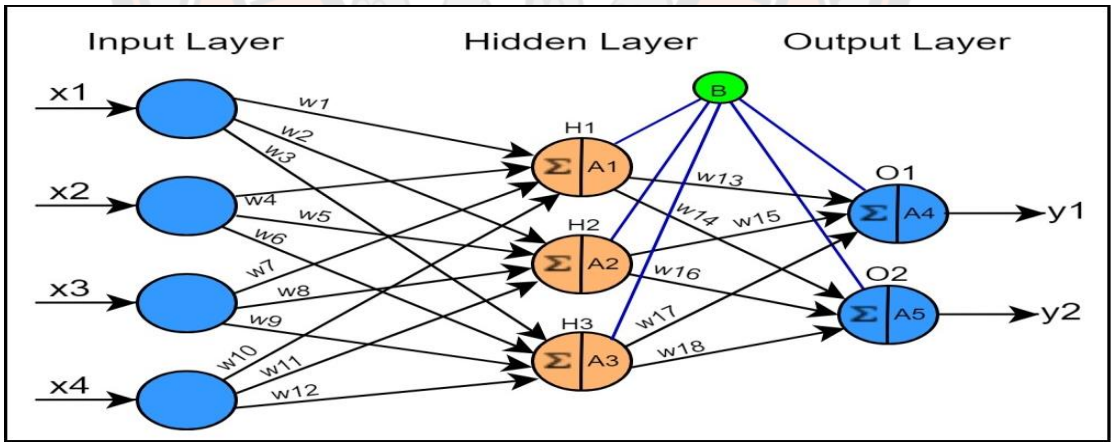


Figure 16 Feedforward neural network with one hidden layer

In the FNN, the features (x_1, x_2, x_3, x_4) and weights from input nodes are fed to the neurons in the hidden layer. The neurons are the processing unit of the network and perform two important operations: weight calculation and activation. The weights are calculated in every neuron by the summation of an element-wise product as derived in Equation 2 and pass to the

threshold function called the activation function. The final output is the predicted value from the input data. The predicted output may have higher losses. FNN aims to estimate some function *f(x)* to predict *y* output from the given *x* input, and defined as,

$$y = f(x) \tag{1}$$

Equation 1 maps the input *x* with the output category *y*. Initially, the random weights are initialized to all weights *w1* to *w18* from the normal distribution (~N(0,1)) and biases to 1.0. Calculation of summation of weights for hidden nodes *H1*, *H2*, and *H3* are given below:

$$H1 = (w1.x1 + w4.x2 + w7.x3 + w10.x4) + b1$$
$$H2 = (w2.x1 + w5.x2 + w8.x3 + w11.x4) + b2$$
$$H3 = (w3.x1 + w6.x2 + w9.x3 + w12.x4) + b3$$

Therefore, the general equation for the weight calculation is defined as,

$$H_n = \sum_{i=1}^{n} (w_i \cdot x_i) + b \tag{2}$$

Where *n* is the number of data points, w_i is weights, x_i is features, and *b* is bias. After the summation of weights in each hidden node, the calculated weights are passed to the threshold function called the activation function. The detailed of activation functions is discussed in the following sections but for illustration purpose, the sigmoid function is used to squash values between 0 and 1, and defined as,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Now, calculated weights *H1*, *H2*, and *H3* are passed to the sigmoid activation function as,

$$A_1 = activation(H1)$$
$$A_1 = \frac{1}{1+e^{-H1}}$$

$$A_2 = activation(H2)$$
$$A_2 = \frac{1}{1+e^{-H2}}$$

$$A_3 = activation(H3)$$
$$A_3 = \frac{1}{1+e^{-H3}}$$

The output of hidden nodes A_1, A_2 , and A_3 become the inputs of output nodes 0_1 and 0_2. The weight of the output nodes is calculated as,

$$O_1 = (w13.A_1 + w15.A_2 + w17.A_3) + b4$$
$$O_2 = (w14.A_1 + w16.A_2 + w18.A_3) + b5$$

Now, selecting the linear output activation function from Equation 1

$$y = f(x) = x$$

Therefore, output 0_1 and 0_2 are passed to the activation function as,

$$A_4 = O_1$$
$$A_5 = O_2$$

Therefore, A_4 and A_5 are the predicted outputs. Now, let A_i be the predicted output and y_i be the desired output from Equation 1. The error is measured using mean square error (MSE) but the selection of loss function depends on researchers. So, the total error is defined as,

$$MSE = \frac{1}{n}\sum_{i=1}^{2}(y_i - A_i)^2 \quad (4)$$

The general equation is defined as,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (5)$$

Where n is the number of data points, y_i is the actual observed values and $\hat{y}_i$ is the predicted value. In the first forward pass, the error is higher. However, this error can be reduced by using the backpropagation with optimizer that updates the weights and trains the model iteratively.

3.3.3.2.2 Back Propagation Algorithm

One of the most important algorithms in machine learning is Back Propagation (BP). In the 1970s and 1980s, researchers independently studied BP learning with the multilayer perceptron (Ojha, Abraham, & Snášel, 2017) and tried to modify the learning equation. However, the publication of the paper by Kariniotakis (2017) about back-propagating errors to represent learning revitalized the implementation of BP and was popular.

The main purpose of the BP is to reduce the error by adjusting the weights from the training data until the difference between the actual observed value and the predicted output is minimum. The difference between the predicted output and the actual value is calculated by using the loss function. During the training, the error should be minimize using an optimizer. To reduce the error, the network should backpropagate and update the weights and biases as shown in Figure 17. After the updating of parameters, the second phase of feedforward training begins with the updated values and computes the error again. Therefore, the model is trained iteratively until the error is minimum over a fixed number of iterations. The loss function and optimizer play an important role in backpropagation.

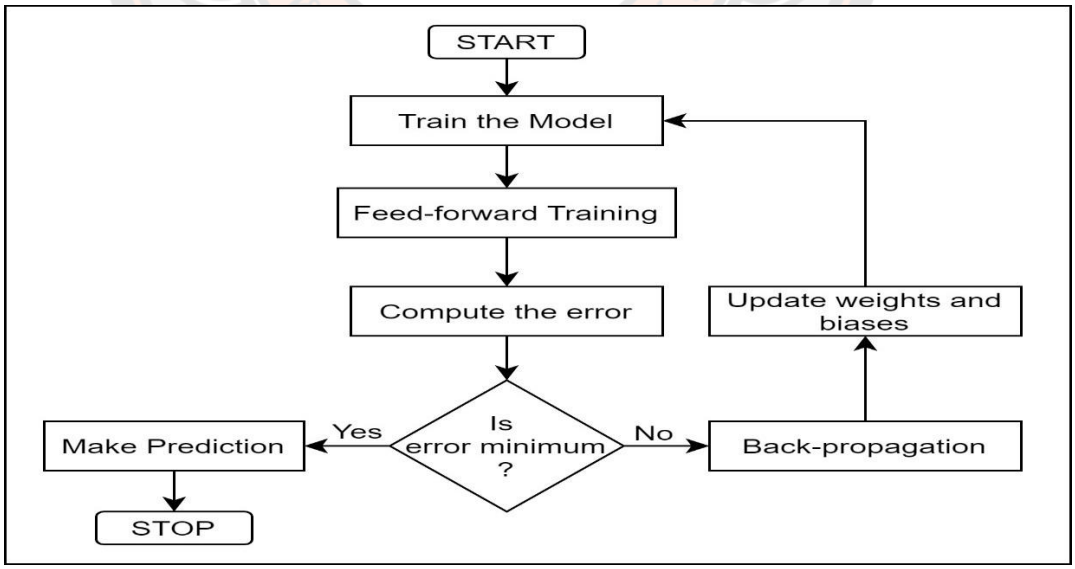


Figure 17 Backpropagation algorithm flowchart

3.3.3.2.3 Loss Functions

The loss function evaluates the performance of the algorithm on the dataset by comparing the prediction and the true value. The difference between predicted and actual value will be either higher or lower depending on whether the learning of the model is bad or good. Different types of loss functions are Mean Absolute Error (MAE) also known as L1 loss, Hinge, Huber, and Kullback-Leibler. The most popular loss functions used for the image classification are Mean Squared Error (MSE) and Cross Entropy. MSE is also known as L2 loss. The MSE equation is given in Equation 5 and cross-entropy is defined in Equation 6 below,

$$H(p,q) = -\sum_{x} p(x) log q(x) \tag{6}$$

Where p is the true label and q is the predicted label. The difference between p and q is reduced using one of the optimizers in backpropagation during the training to update the parameters of the network such as weights and biases.

3.3.3.2.4 Optimizers

An optimizer is an algorithm that calculates the minimum value of the function. It reduces the loss by fine-tuning the parameters of the neural network such as weights, biases, and learning rates during the backpropagation. There are different types of optimizers such as Gradient Descent, Stochastic Gradient Descent (SGD), Adagrad, RMSprop, Adam, Adadelta, Adamax, Nadam, and Ftrl. Many researchers have been describing Gradient Descent as the mother of machine learning. It is the main workhorse of the features of learning. Figures 18 and 19 show the loss calculation and weights update using gradient descent respectively.

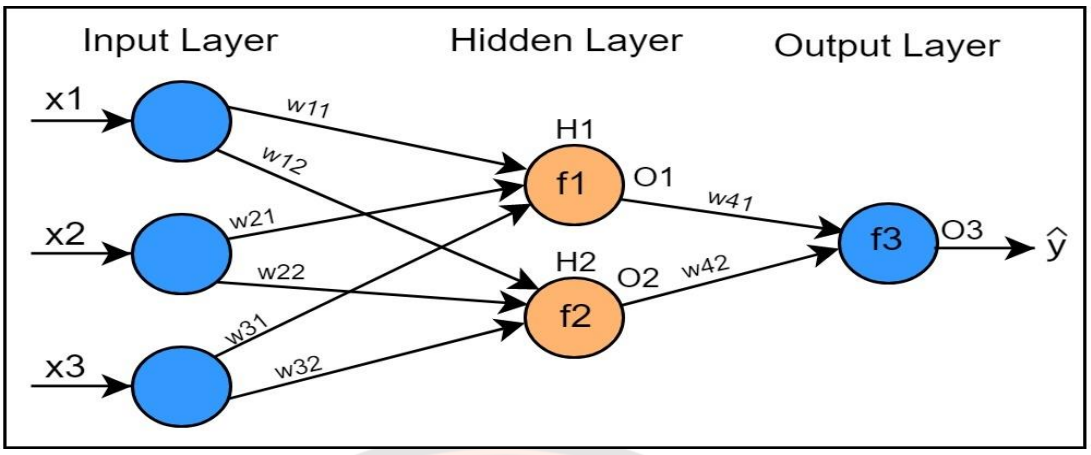


Figure 18 Loss computation with feedforward network

The loss is calculated by finding the difference between the true value (y) and the predicted output $(\hat{y})$. The predicted output is computed by the hidden neurons in two important steps. In the first step, all the inputs and weights which are connected to the neuron are multiplied and a bias is added as defined in Equation 2 as,

$$H = \sum_{i=1}^{n}(w_i \cdot x_i) + b$$

Therefore, the summation of weights at hidden neurons H1 and H2 are:

$$f_1 = (w_{11}.x_1 + w_{21}.x_2 + w_{31}.x_3) + b$$
$$f_2 = (w_{12}.x_1 + w_{22}.x_2 + w_{32}.x_3) + b$$
$$f_3 = (w_{41}.o_1 + w_{42}.o_2) + b$$

In the second step, these calculated weights are passed to the activation function (sigmoid) and defined in Equation 3 as,

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

Therefore, passing f_1 and f_2 in the sigmoid activation:

$$f(x) = \frac{1}{1+e^{-f_1}}$$

Output from H1 $o_1 = \sigma(f_1)$

$$f(x) = \frac{1}{1+e^{-f_2}}$$

Output from H2 $o_2 = \sigma(f_2)$

$$f(x) = \frac{1}{1+e^{-f3}}$$

Output from f3 $o_3 = \sigma(f_3)$

Therefore, the final predicted output is $o_3 = \hat{y}$.

The loss is calculated by using MSE and defined in Equation 5 as,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Therefore, $loss(L) = (y - \hat{y})^2$. This loss needs to be minimized by one of the optimizers such as Gradient Descent or Adam.

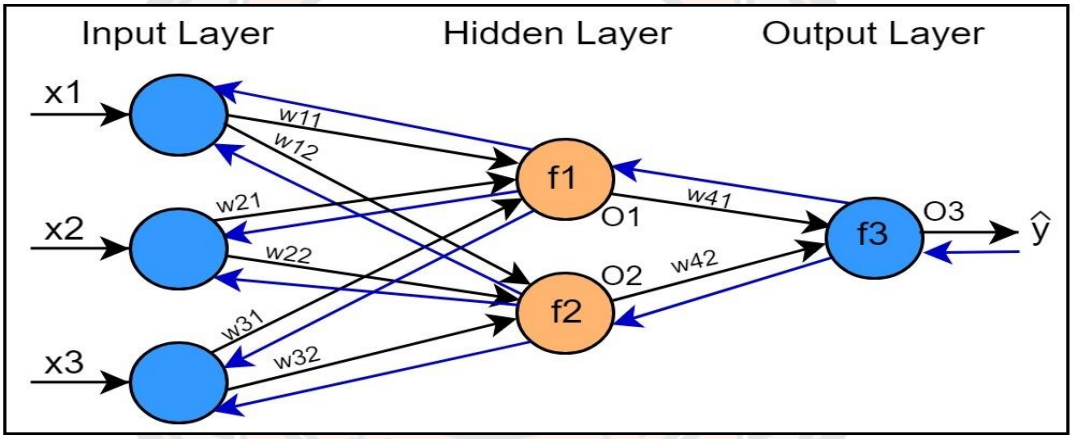


Figure 19 Weights update with Back Propagation using Gradient Descent

The weight update formula is defined as,

$$w_{(t+1)} = w_{(t)} - \eta \frac{\partial L}{\partial w(t)} \quad (7)$$

Where $w_{(t+1)}$ is a new weight, $w_{(t)}$ is the old weight, η is the learning rate which decides the rate of convergence to the global minima, and $\frac{\partial L}{\partial w(t)}$ is the derivative of loss (slope) to the old weight that decides the direction of the descent to the global minima. All the weights are updated in the back propagation. For the weight update illustration purpose, let's take weights *w41* to replace in Equation 7 as,

$$w_{41_{new}} = w_{41} - \eta \frac{\partial L}{\partial w41} \quad (8)$$

The values of w_{41} and η are known in the feedforward training. Therefore, $\frac{\partial L}{\partial w41}$ needs to be computed to calculate the new weight $w_{41_{new}}$. The loss (L) is affected by the output *O3* and O3 in turn is impacted by w41 as shown in Figure 16. It is calculated by applying the chain rule as,

$$\frac{\partial L}{\partial w41} = \frac{\partial L}{\partial O3} \cdot \frac{\partial O3}{\partial w41}$$

Similarly, the weight *w11* is updated by the following equation,

$$w_{11_{new}} = w_{11} - \eta \frac{\partial L}{\partial w11} \quad (9)$$

From the impact analysis, the loss (L) is affected by *03*, and *O3* is affected by *O1,* and *O1* in turn is affected by *w11* as shown in Figure 16. Therefore, using the chain rule $\frac{\partial L}{\partial w11}$ is computed as

$$\frac{\partial L}{\partial w11} = \frac{\partial L}{\partial O3} \cdot \frac{\partial O3}{\partial O1} \cdot \frac{\partial O1}{\partial w11}$$

Therefore, other weights are updated similarly as shown in Equations 8 and 9. When all the weights are updated, the second iteration starts. The number of weight updates is as same as the number of epochs of the training. In every epoch, the loss is reduced. However, the loss can remain the same or become larger. The vanishing gradient (VG) problem makes the network hard to learn and error reduction remains almost the same or a little change but the exploding gradient (EG) problem makes a large error by updating large weights. The VG problem can be solved by using different activation functions other than sigmoid or tanh and the EG is solved by appropriately initializing the weights during the training.

3.3.3.3 Components of CNN

The computer and human being see images differently. The computers view images as the numbers having rows and columns of pixels as

shown in Figure 20. The images are divided into shades of colours ranging from 0-255 pixels values. A grayscale image has 256 different shades of color where 0 is black and 255 is the white. However, the RGB image has three channels in which each channel has a color ranging from 0-255 pixels. The grayscale image matrix is written as 6x6x1 and RGB as 6x6x3 assuming that the image size is 6x6. The image matrix (6x6x3) is flattened into a 108 vector and fed into CNN as the input as shown in Figure 21. Therefore, there are 108 input neurons in the input layer and these data are passed through different components of CNN.

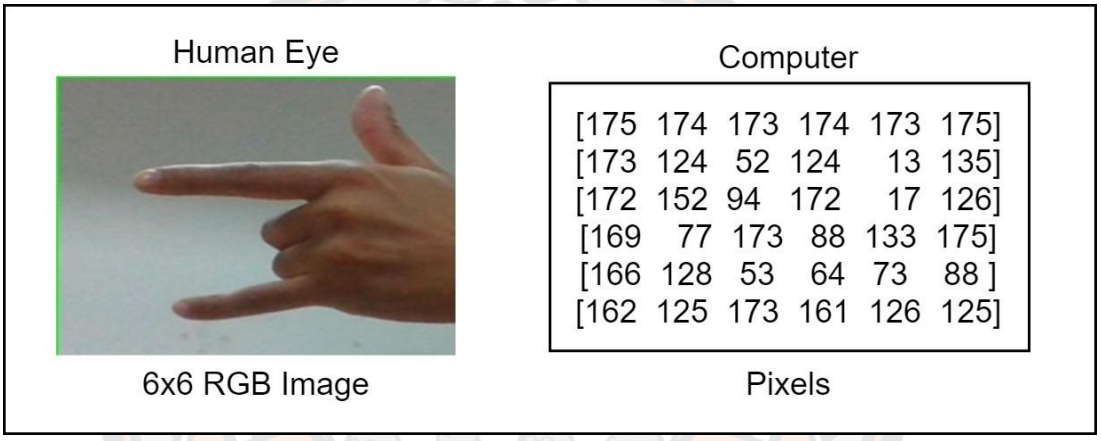


Figure 20 Different perceptions of images by human and computer

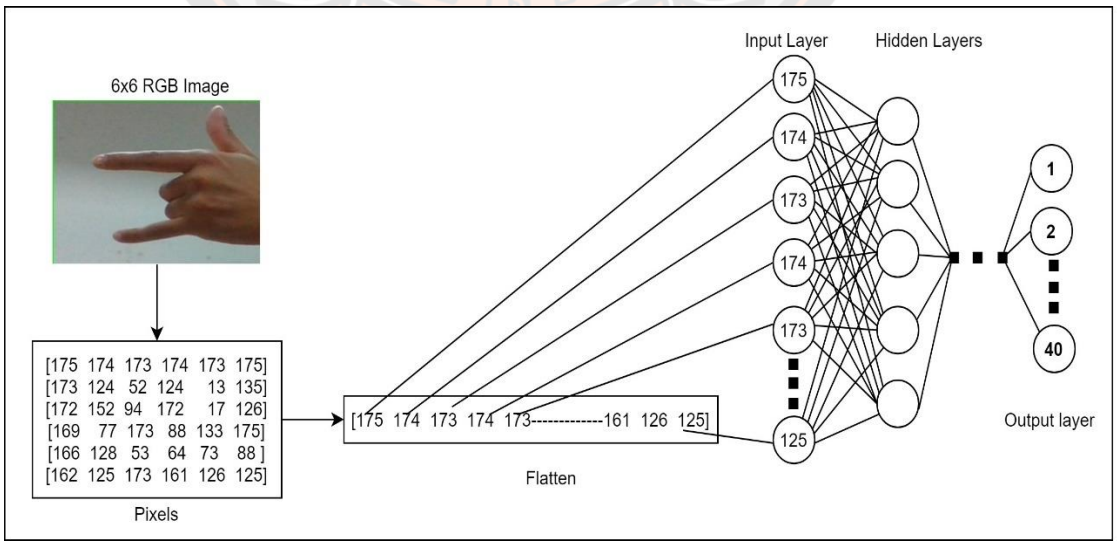


Figure 21 Image matrix is flattened and fed into the network

Different components of CNN are Convolution layers, Activation functions, Pooling layer, Flatten Layer, Fully Connected (FC) layer, and Loss function layer with the softmax function to classify one of the classes. The different arrangements of these components give a new architectural design (Rumelhart, Hinton, & Williams, 1986). CNN takes images as the input and processes their pixel values through a series of layers for features extraction and class classification as shown in Figure 22. These layers are discussed in the following sections.

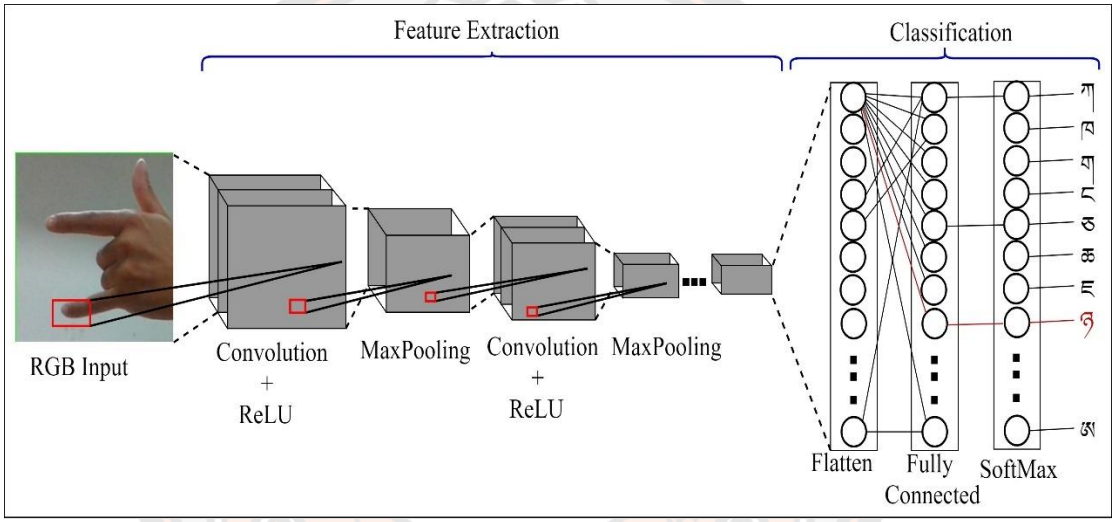


Figure 22 Components of Convolutional Neural Network

3.3.3.3.1 Convolution Layer

The convolution layer is the first component of CNN. It extracts features (lines, edges, corners, curves) from the input image with the number of filters. The image matrix (H x W x D) is multiplied (convolved) with filter matrix (H_f x W_f x D_f) to produce the feature map (*[H-Hf+1] x [W-Wf+1] x 1*) as shown in Figure 23. For example, a square image that has H=W=50 is convolved with 10 filters of size 3x3x3. The feature map of (50-3+1) x (50-3+1) x 10 = 23040 are generated.

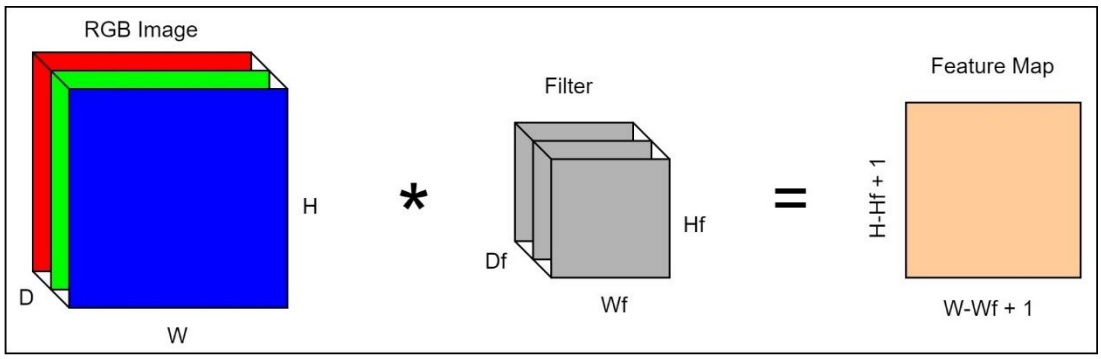


Figure 23 Convolution on an image with one filter to produce a feature map

The filter divides the image into smaller slices called receptive fields. Then the filter slides over these receptive fields by convolving with its specific weights with one unit of stride as shown in Figure 24. However, different strides are used to move filters on the image. During the convolution operation, the dimensions of the image are reduced and the filters do not exactly fit on the image. These problems could be solved by using padding. By applying zero-padding around the images, the same dimensions are maintained.

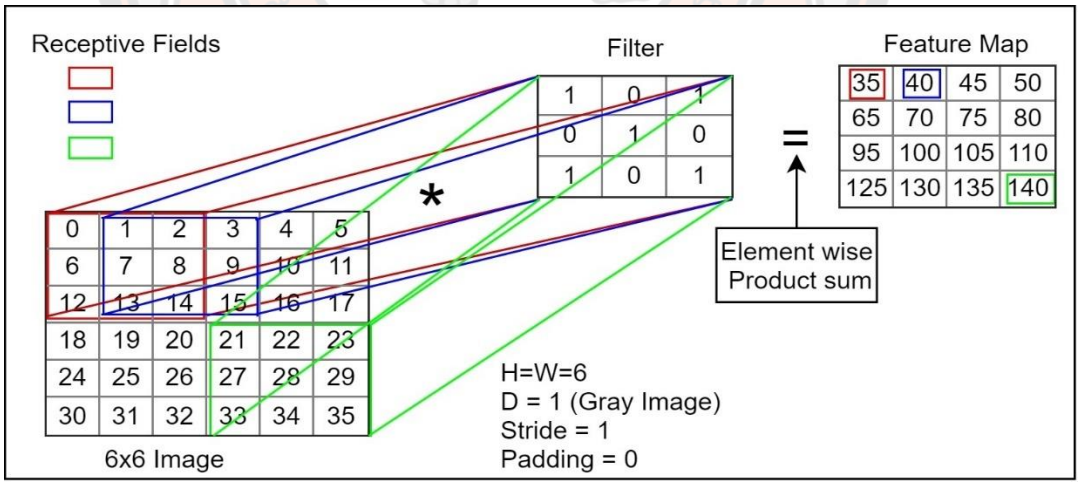


Figure 24 Receptive field and feature map

The convolution operation is defined as,

$$g\,[x, y] = (f * k)[x, y] \tag{10}$$

$$g\,[x,y] = \sum_i \sum_j f[x-i, y-j] * k[i,j] \tag{11}$$

Where input image and filter are denoted by *f* and *k* respectively. The rows and columns are represented by *x* and *y*. The output of a convolution is the feature map and is marked with *g*. The *i* and *j* are the indexes with which the image and filter convolve. The first cell of the feature map (*g[0,0]*) is calculated by the sum of the element-wise product as illustrated by Equations 9 and 10. The calculation of the feature map in Figure 23 is demonstrated below:

$g\,[0,0] = 0.1 + 1.0 + 2.1 + 6.0 + 7.1 + 8.0 + 12.1 + 13.0 + 14.1 = 35$

$g\,[0,1] = 1.1 + 2.0 + 3.1 + 7.0 + 8.1 + 9.0 + 13.1 + 14.0 + 15.1 = 40$

$g\,[0,2] = 2.1 + 3.0 + 4.1 + 8.0 + 9.1 + 10.0 + 14.1 + 15.0 + 16.1 = 45$

:

$g\,[3,3] = 21.1 + 22.0 + 23.1 + 27.0 + 28.1 + 29.0 + 33.1 + 34.0 + 35.1 = 140$

Filters play a very important role in image content analysis. Filters extract different features while convolving on the image such as edge detection, sharpening, and blurring. Figure 25 illustrates the convolutional operations performed with some of the filters.

Operation	Filter(k)	Feature Map g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Blur	$\begin{bmatrix} 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Horizontal Lines	$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$	
Vertical Lines	$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$	
Combination of Lines	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Figure 25 Convolution operation with different filters

3.3.3.3.2 Activation Functions

Activation functions determine the output of the neural network by activating (firing) the neuron based on the relevant information from input data. The neuron would not be fired (activated) if it does not contain features to predict desired outputs. At each neuron, the sum of products of inputs and the weights are calculated and then the activation function is applied to get the output which is then supplied to the next layer (Khan et al., 2019). According to the authors, there are 10 activation functions such as binary step function, linear, sigmoid, tanh, ReLU, leaky ReLU, parametrized ReLU, exponential linear unit, swish, and softmax which squashes the output to the finite value. However, the Rectified Linear Unit (ReLU) is the most popular activation function used in deep learning.

The sigmoid and Tanh functions suffer from vanishing gradient problem where neurons either do not update their weights or update weights very slowly. The sigmoid and Tanh take real numbers and squash between 0 to1 and -1 to 1 respectively as shown in Figure 26. The sigmoid activation function is defined in Equation 3 as,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The Tanh activation function is defined as,

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{11}$$

The ReLU takes real-valued numbers and squashes between 0 to the maximum number and is defined as,

$$f(x) = \max(0, x) \tag{12}$$

$$f(x) = \begin{cases} 1, if\ x > 0 \\ 0, if\ x < 0 \end{cases} \tag{13}$$

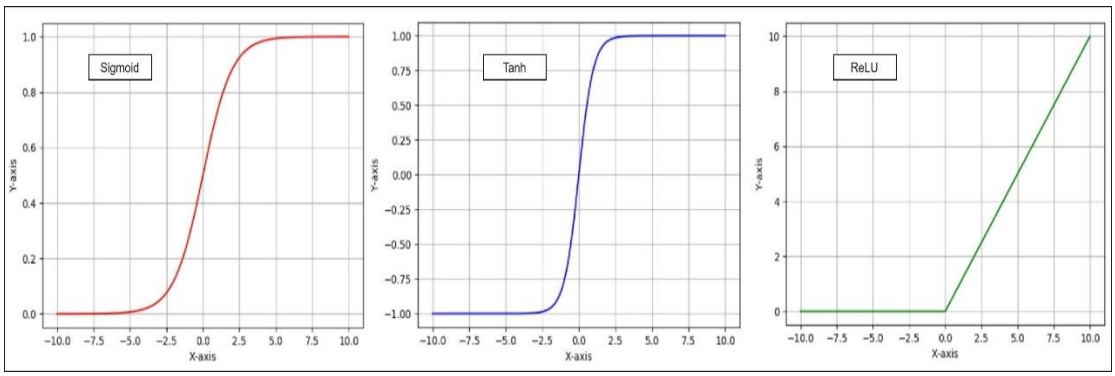


Figure 26 Graphs showing a respective range of activation functions

3.3.3.3.3 Pooling Layer

Pooling reduces the spatial dimensions of the input images (down sampling) and operates on each feature map independently. It reduces the number of parameters and makes computationally less expensive to process. The pooling aims to combine the most prominent features of an image by preserving significant information and discarding others (Sharma, 2017). There are three types of pooling: average pooling, max pooling, and global pooling.

The average pooling calculates the average of the pixels present on the feature map. The total number of pixels of the feature map on which average is computed depends on the size of the filter and stride as shown in Figure 27. If the filter size is 2x2, there would be 4 pixels on which average is computed. The average pooling is defined as,

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{14}$$

Where n is the total number of pixels and x is the pixel on the local pooling region.

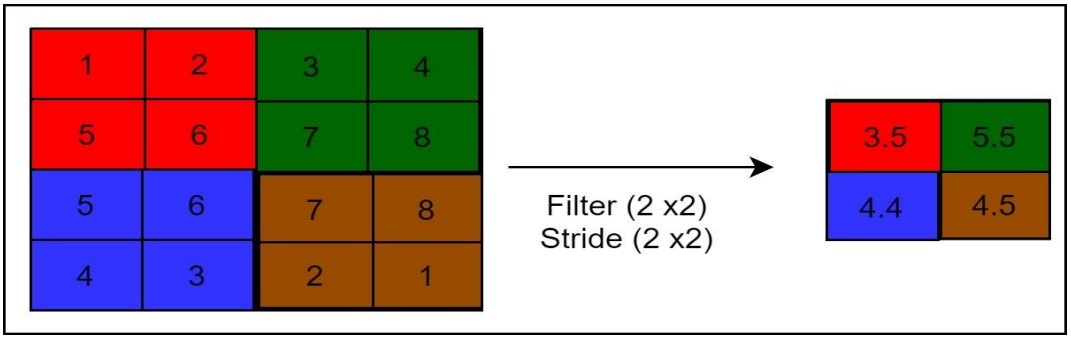


Figure 27 Average pooling

Similarly, max-pooling depends on the size of the filter and stride but takes the maximum number from the local region as shown in Figure 28. It is defined as,

$$f(x) = max_i . x_i \quad (15)$$

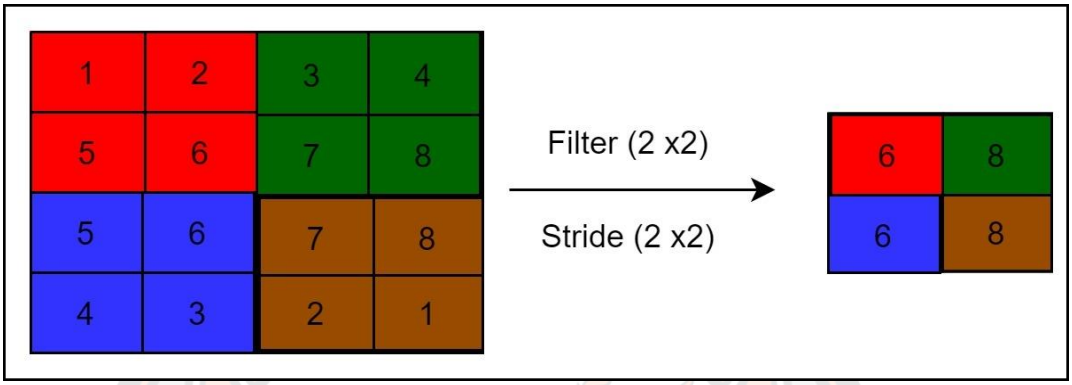


Figure 28 Max pooling

However, global pooling takes only a single value from the feature map. It drastically reduces the dimension. Therefore, the feature map having N_h x N_w x N_c dimension is reduced to the 1 x 1 x N_c dimension. The global pooling can be either global max pooling or global averaging pooling.

3.3.3.3.4 Batch Normalization

Batch normalization (BN) is a deep learning technique that speeds up the training process, reduces the covariance shift, stabilizes the network, and provides a regularization which is proposed by Yu, Wang, Chen, and Wei (2014). In the BN process, the input data are divided into a number of batches. These mini-batches are scaled using their mean and standard deviation which results in the introduction of noises in each layer providing a regularization effect to solve overfitting problems. The stability of the network is increased by normalizing the output of the previous activation layer by subtracting and dividing with the batch mean and standard deviation respectively as shown in Equations 15 and 16. BN increases the rate of convergence of the algorithm and reduces the covariance shift. Therefore, it improves accuracy and accelerates the training process (Ioffe & Szegedy, 2015).

During the network training, normalization of the input features can increase the speed of learning. The mean and variance are expressed by Equations 15 and 16 respectively. Then normalization is applied as shown by Equation 17. The mathematical equations of mean and variance are defined as,

$$\mu = \frac{1}{n}\sum_{i} z^{(i)} \tag{16}$$

$$\sigma^2 = \frac{1}{n}\sum_{i} (z_i - \mu)^2 \tag{17}$$

$$z_{norm}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{18}$$

Where $z^{(i)}$ is some hidden unit value of hidden layer (l) in the neural network, μ and σ^2 represent mean and variance respectively. The ε is added to gain numerical stability in case σ^2 becomes zero while normalizing. After normalization, every component of z has 0 mean and standard unit variance. However, hidden units do not want to have 0 mean and one variance rather hidden units have different distributions. The different distribution is defined as,

$$\tilde{z}^{(i)} = \gamma z_{norm}^{(i)} + \beta \tag{19}$$

Where γ and β are the learnable parameters of the batch normalization which allow hidden units to select different values of mean and variance.

3.3.3.3.5 Dropout

Deep neural networks are powerful machine learning systems. However, a large number of parameters make networks slow and when trained with the small data can overfit the training data (Bjorck, Gomes, Selman, & Weinberger, 2018). The overfitting is the main problem of the deep neural network and as a result, the model performs poorly on the test dataset (new data), and the error of generalization increases. Dropout is the technique to solve overfitting during the training of the network.

Dropout allows networks to train faster, reduces overfitting, and improve the generalization of the deep networks. It randomly drops out nodes in every layer during the training as shown in Figure 29. According to Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014), nodes are temporarily removed from the network including all the incoming and outgoing connections. As a result, parameters are reduced and train the network faster and provides a regularization effect by not learning all the data points in a single epoch to address the overfitting. The different dropout ranges are used but a good range is between 0.5% to 0.8%. Usually, the dropout rate of the input layer is bigger.

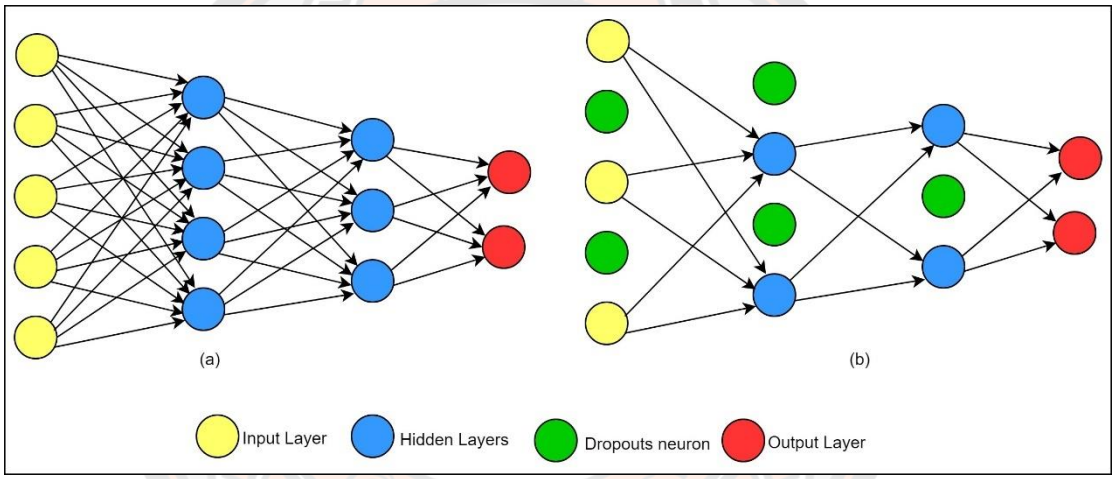


Figure 29 Dropout neural network: (a) Neural network with 5 inputs, 2 hidden layers, and 2 outputs, (b) Implementation of dropouts and displaying reduced parameters of the neural network

3.3.3.3.6 Fully Connected

The fully connected (FC) layer is attached to the end of the network and detects high-level features of the data to predict one of the classes. The lower features are learned in the convolution layers with activation functions and pooling layers in previous layers. The feature maps have all the prominent features of the data that are fed into the network. However, the feature map matrix is flattened into a single column vector before feeding it to FC as shown in Figure 30. The model is trained for a series of iteration and able to learn from the training. Finally, the classes are predicted using the softmax activation function.

The state-of-the-art image classification algorithms are mostly designed based on CNN with slightly different architecture by stacking CNN components differently. The popular CNN based architectures such as AlexNet, VGGNet, GoogLeNet, and ResNet are discussed in the following sections.

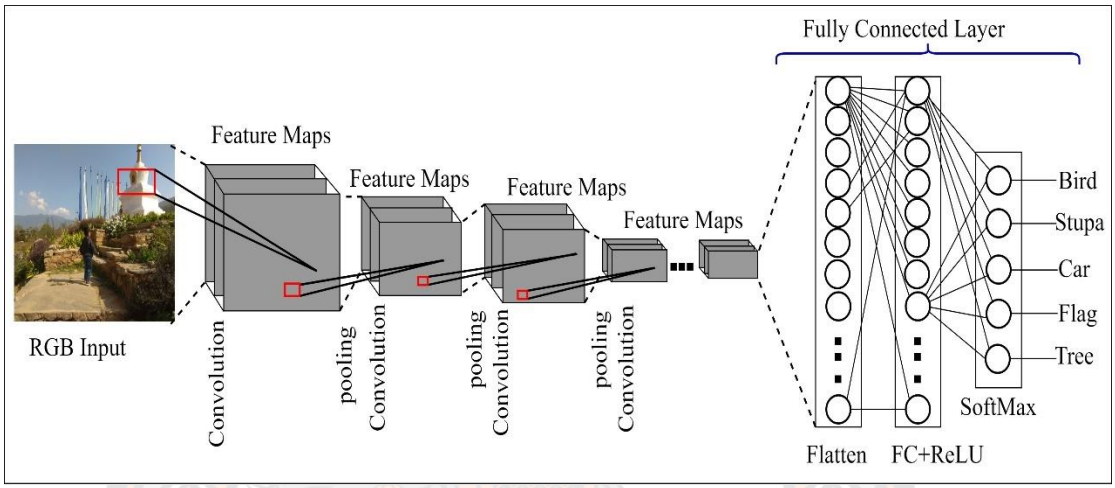


Figure 30 Flatten feature maps into a vector and feed into FC

3.3.3.4 Visual Geometry Group

The VGG is a deep CNN based state-of-the-art network for visual recognition developed by Oxford's team called Visual Geometry Group (VGG). It achieves good performances on the ImageNet dataset and other images beyond ImageNet. VGG was the first runner up of the ILSVRC2014 image classification competition (Srivastava et al., 2014). It is one of the mostly used architectures for image classification and recognition tasks.

Simonyan and Zisserman (2014) presented six different VGG networks by stacking layers to evaluate the performances as shown in Table 7. The RGB images 224 x 224 are feed to the VGG network consisting of convolution layers with 3x3 filters of a stride 1 and same padding followed by max-pooling layers with filter 2x2 of stride 2. The filter size is 3x3 but the number of filters in convolution layers increase by double of the previous layer such as 64, 128, 256, and 512. The arrangement of these layers is the same throughout the architecture. However, the number of layers is different in different VGG network.

Table 7 Configuration of different models of VGG

VGG11-A	VGG11-B	VGG13	VGG16-A	VGG16-B	VGG19
Conv3_64	Conv3_64	Conv3_64	Conv3_64	Conv3_64	Conv3_64
	LRN	Conv3_64	Conv3_64	Conv3_64	Conv3_64
maxpool					
Conv3_128	Conv3_128	Conv3_128	Conv3_128	Conv3_128	Conv3_128
		Conv3_128	Conv3_128	Conv3_128	Conv3_128
maxpool					
Conv3_256	Conv3_256	Conv3_256	Conv3_256	Conv3_256	Conv3_256
Conv3_256	Conv3_256	Conv3_256	Conv3_256	Conv3_256	Conv3_256
			Conv1_256	Conv3_256	Conv3_256
					Conv3_256
maxpool					
Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512
Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512
			Conv1_512	Conv3_512	Conv3_512
					Conv3_512
maxpool					
Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512
Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512	Conv3_512
			Conv1_512	Conv3_512	Conv3_512
					Conv3_512
maxpool					
FC4096					
FC4096					
FC1000					
softmax					

3.3.3.5 Residual Network (ResNet)

In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification competition, AlexNet (Simonyan & Zisserman, 2014) CNN-based architecture obtained state-of-the-art performance and won the contest in 2012. In the subsequent years, the winner architectures (Krizhevsky et al., 2012; Szegedy et al., 2015; Zeiler & Fergus, 2014) used more and deeper layers to increase accuracy and reduce the error. However, increased layers posed either vanishing gradient or exploding gradient problem (Simonyan & Zisserman, 2014). The gradient either becomes 0 or too large making train and test error rise. Besides, deep neural networks require a huge amount of data to extract features and high-performance GPUs to train complex data models. To address these issues, after 2012, a variant of CNN models

(Jiwon Kim, Kwon Lee, & Mu Lee, 2016; Simonyan & Zisserman, 2014; Szegedy et al., 2015; Zeiler & Fergus, 2014) has been developed by the researchers and took part in the ILSVRC competition. In 2015, deep ResNet (He et al., 2016) won the ILSVRC contest. It has been a ground-breaking and influential algorithm in the deep learning community. The ResNet allows hundreds or thousands of layers to train the model until better performance is achieved.

ResNet was proposed by researchers at Microsoft (He et al., 2016). It trains a very deep neural network without causing vanishing/exploding gradient problems with the concept called Residual Network. The architecture of the ResNet is similar to CNN or VGG consisting of the number of convolutional layers, activation functions, batch normalization, dropouts, and dense layers. However, in addition to these layers, ResNet consists of a shortcut connection (skip connection). The shortcut connection (identity) skips training by a few layers and adds directly to the output as shown in Figure 31. It allows a network to learn from the residual $(H(x) - x)$ mapping rather than layers learn from the underlying mapping $F(x)$. The network fits $F(x) = H(x) - x$ that gives $H(x) = F(x) + x$.

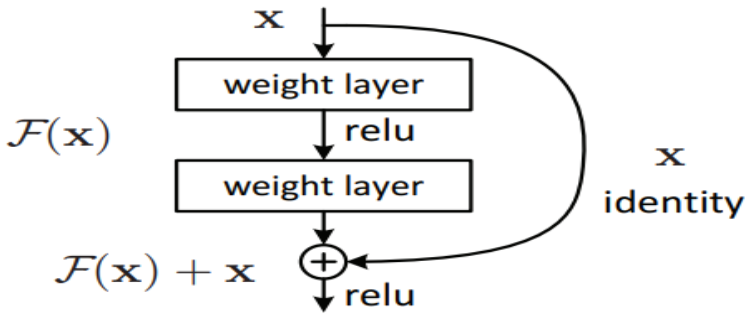


Figure 31 Building blocks of Residual Learning

Source: He et al. (2016)

There are two types of residual block: identity block and convolutional block as shown in Figure 30. The identity block passes input x directly to the output by skipping training layers as shown in Figure 32 (a). However, the size of the output image $F(x)$ must be the same as the input image x in ResNet. On the contrary, the

ResNet with convolutional block deals with different image sizes. The input image size 128x128 with the filter size of 32 gets reduced to 64 x 64 as shown in Figure 32 (b).

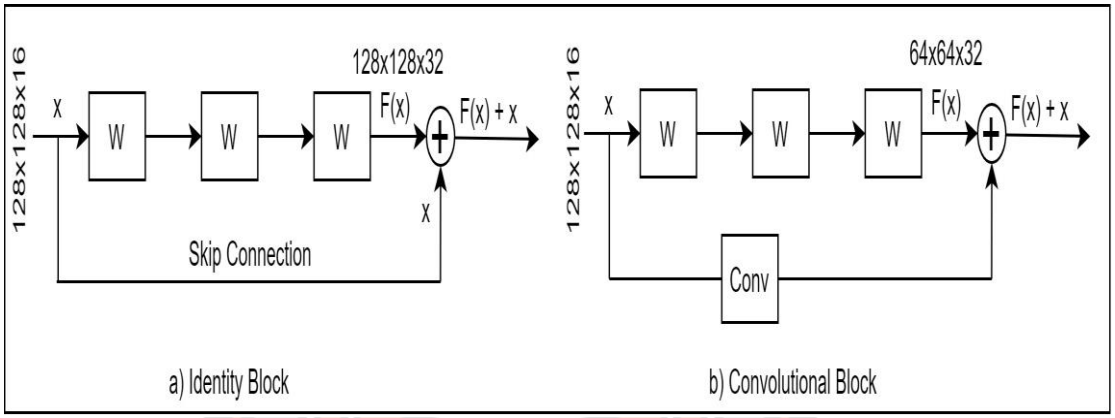


Figure 32 Residual blocks: a) Identify block, b) Convolution block

Using a 1x1 convolutional block, the input image is resized to match the output image. The shape of the next layer is calculated by Equation. 20. However, padding is not used. Therefore, the equation changed to Equation. 21 as shown below:

$$\frac{n + 2p - f}{s} + 1 \quad (20)$$

$$\frac{n - f}{s} + 1 \quad (21)$$

Therefore, the shape of the feature map is $\frac{128-1}{2} + 1 = 64$

There are different variants of ResNet such as ResNeXt (He et al., 2016), DenseNet (S. Xie, Girshick, Dollár, Tu, & He, 2017), Deep Network with stochastic depth (G. Huang, Liu, Van Der Maaten, & Weinberger, 2017), and ResNet as an Ensemble of Smaller Network (G. Huang, Sun, Liu, Sedra, & Weinberger, 2016).

3.3.3.6 LeNet

The LeNet was proposed to recognize handwritten digits in images in AT&T Bell Labs. The LeNet was trained successfully using CNNs via backpropagation (Veit, Wilber, & Belongie, 2016). The results obtained were outstanding similar to

SVM. The architecture of LeNet comprised of convolutional layers, pooling, and dense layers. The architecture of LeNet-5 is shown in Figure 33.

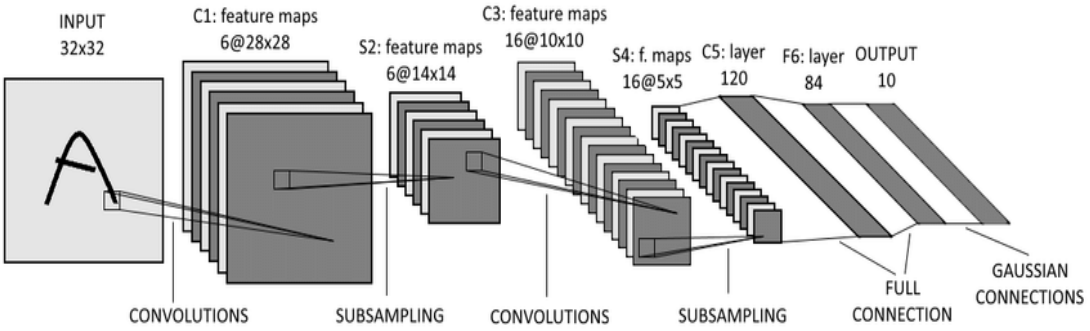


Figure 33 The architecture of LeNet-5

Source: Yann LeCun, Bottou, Bengio, and Haffner (1998)

The LeNet-5 was trained on the MNIST database. The input size of the images was 32x32. The first convolutional layer used six filters to extract features followed by subsampling with six filters. The second convolutional layer used 16 filters followed by another subsampling with 16 filters. Then the all the trainable parameters were passed to three fully connected layers for the classification.

3.3.3.7 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning model that is used in the classification (Yann LeCun et al., 1998) and regression (Burges, 1998) problems. SVM performs classification for both linearly separable or non-linearly separable data as shown in Figure 34 by finding the best line or hyperplane.

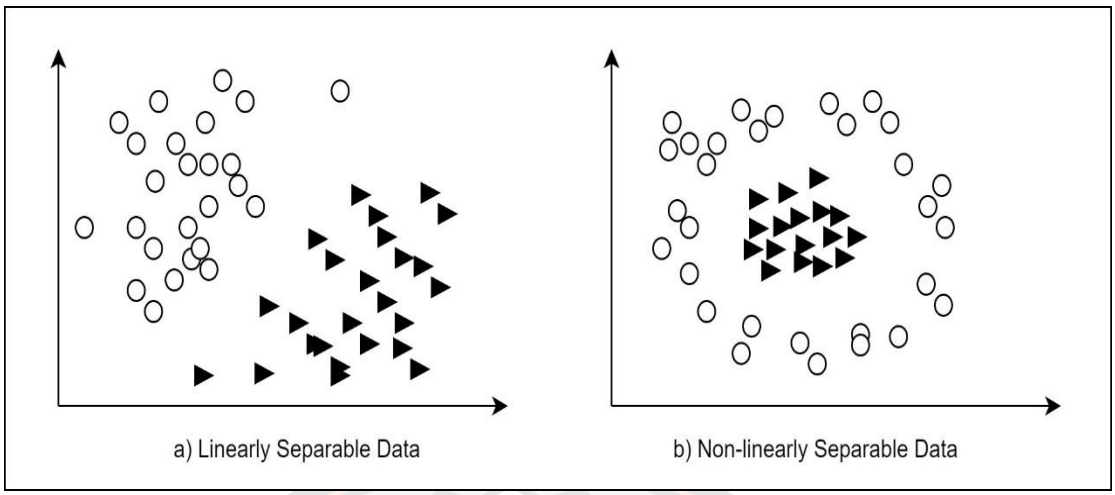


Figure 34 Linearity and non-linearity data

The linear SVM uses hyperplane, marginal distance, and support vector to linearly separate data as shown in Figure 35. The data points are classified by decision boundaries called Hyperplanes. The hyperplane is either a line (two features) or a two-dimensional plane (higher features). Different classes fall on either side of this optimal hyperplane. The hyperplane with the maximum marginal distance classifies the data point better. Therefore, support vectors help to maximize the margin between classes. Support Vectors are the data points (blue and green) as shown in Figure 35(b) that affect the position and orientation of the hyperplane.

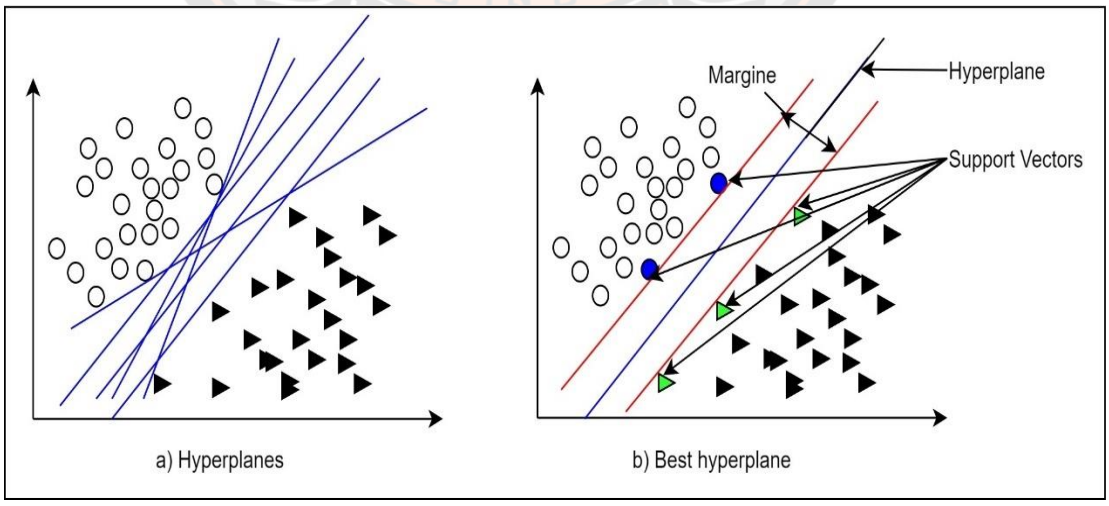


Figure 35 Selection of the optimal hyperplane

The non-linear SVM is used to classify non-linearly separable data. A line cannot separate non-linear data. The kernel functions (Smola & Schölkopf, 2004) such as polynomial kernel, radial basis function (RBF) (Patle & Chouhan, 2013), and sigmoid kernel are used to map input vectors into higher-dimensional space (S. Han, Qubo, & Meng, 2012) to transform non-linearity data to linearity as shown in Figure 36.

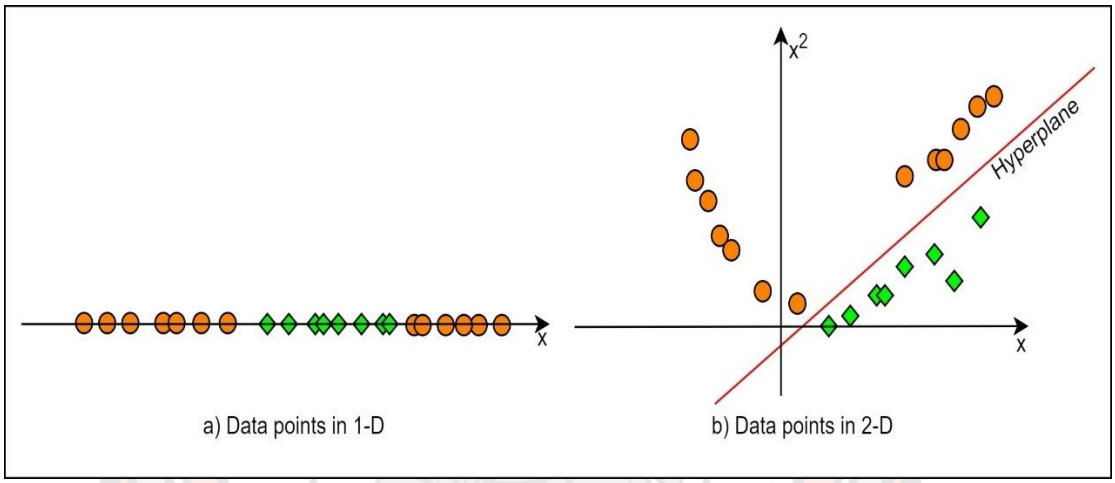


Figure 36 Transformation of non-linearity 1-D data to 2-D

SVM is used with the smaller dataset because it takes a longer time to process. It can be implemented in various fields such as bioinformatics, face detection, categorization of text and hypertext, handwriting recognition, and so on. Funaya and Ikeda (2012) recognized handwritten digits with a novel method based on SVM and Bat algorithm. The proposed method obtained an accuracy of 95.60%.

3.3.3.8 K-Nearest Neighbours

K-Nearest Neighbour (KNN) is a supervised machine learning algorithm, similar to SVM, used for classification and regression problems. In addition to classification and regression, KNN is also used for imputation of missing values (Tuba, Tuba, & Simian, 2016). KNN assumes that similar classes exist in proximity (Zhang, Li, Zong, Zhu, & Cheng, 2017). Therefore, KNN finds the similarity between a new data point and the classes. The new data gets a class assignment with the shortest

distance. The similarity between classes is calculated by Euclidean distances. The algorithm for KNN is given below:

1. START
2. LOAD the data
3. SELECT number of k
4. COMPUTE Euclidean distance of k number of neighbours
5. TAKE k nearest neighbours with which distance is computed
6. COUNT the number of data points in each class
7. ASSIGN a class to a new data point after voting
8. STOP

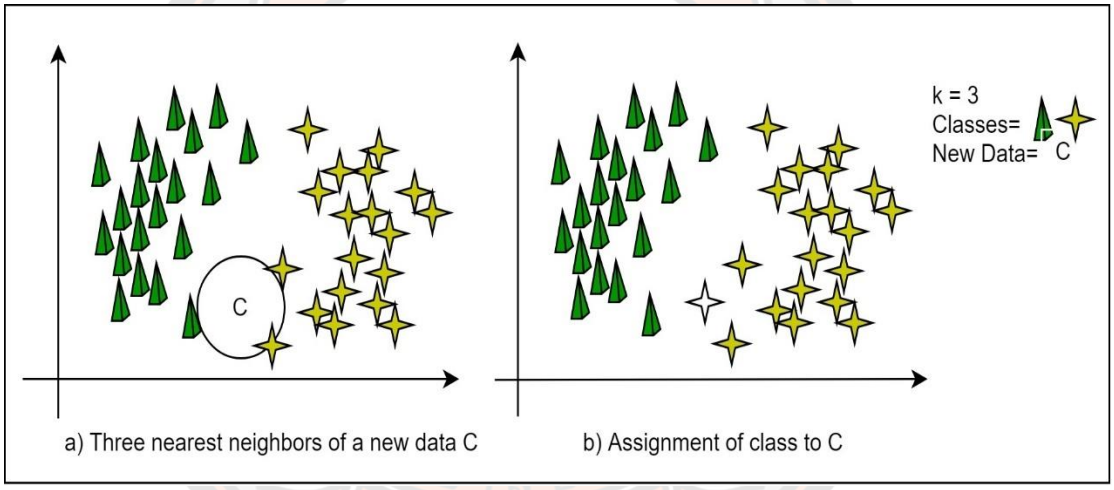


Figure 37 Class assignment to the new data point

Figure 37 explains the identification of a class for the new data point C. The nearest neighbour data points are selected based on the k value. There are two data points (star) and one data point (pyramid) as shown in Figure 37(a) that are the nearest to point C. The class for the new data C is assigned based on voting (Kotsiantis, Pierrakeas, & Pintelas, 2003) as shown in Figure 37(b). The new data C is assigned to star class because there are two votes (two data) when k is 3.

KNN is simple to implement. The model building is not required. KNN performs well with fine-tuning of several parameters. However, KNN is slower with a higher number of classes. Babu, Venkateswarlu, and Chintha (2014) conducted handwritten digits recognition using KNN and obtained an accuracy of 96.64%.

3.3.3.9 Logistic Regression

Linear Regression is used for assessing the relationship between a dependent (Y) and independent (X) variables that predicts future dependent variable (Babu et al., 2014). However, Logistic Regression (LR) is used for the multi-class classification problem. In LR, the value of Y is either categorical or binary. The sigmoid function is used to calculate probability values that are mapped to the number of classes. The different types of LR are Binary LR, Ordinal LR, Nominal LR, and Poisson LR.

Binary LR has two categorical values and 2 levels of characteristics such as On/Off, Male/Female, Yes/No, True/False. However, Ordinal LR has three or more categorical values and characteristics levels are ordered such as good, better, best or agree, neutral, disagree. Nominal LR also has three or more categorical values but characteristics levels are not ordered such as country (Bhutan, Thailand, Australia), employee (Tandin, Gawa, Wangyal), color (Red, Orange, Blue). Similarly, Poisson LR has three or more categorical values but characteristics are the number of events that occurred. For instance, the total number of goals scored in each match in the football tournament (2, 3,1,2,5,3,5).

The researcher (Palvanov & Im Cho, 2018; Saleem & Chishti, 2020) have used LR for the recognition of the handwritten digit on the dataset called MNIST (Modified National Institute of Standards and Technology) that was curated by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. However, CNN based neural network outperformed the LR algorithm.

CHAPTER IV
RESULT AND DISCUSSION

4.1 Introduction

In this chapter, the discussion of the detailed results of the proposed study is presented. The main purpose of the study was to develop a machine learning model for the Bhutanese Sign Language hand-shaped alphabets and digits recognition. Besides, the study also focused on curation and preparation of the first-ever BSL hand-shaped alphanumeric image dataset. The dataset is evaluated with different sign language models discussed in Chapter II and Chapter III. An appropriate model is selected and modified to suit the BSL dataset that consisted of 60,000 images from 40 classes.

In this study, three different BSL datasets were created. The first and second datasets consisted of digits with 10 classes and alphabets with 30 classes respectively. The third dataset consisted of both alphabets and digits with 40 classes. Different Sign Language models are trained and evaluated on these datasets using Google's cloud service called Google Colab. The Google provides 12 hours free GPU and TPU resources with 12 GB RAM. The size of the RAM can be optionally increased to 25 GB. In the following sections, the result discussion on digit dataset with different models is presented first followed by an alphabet dataset. Finally, alphanumeric dataset results are discussed.

4.2 BSL Digits Detection and Recognition

The BSL digit dataset consisted of 20,000 images. The CNN based algorithm was used to extract features from images. The algorithm was designed similar to VGGNet that had three blocks consisting of two convolutional layers in each block as shown in Figure 38. The number of filters used was 32, 64, and 128 in three blocks respectively. The batch normalization was used after two consecutive convolutional layers followed by max-pooing, dropout layers, and two dense layers as shown in Table 8. The batch normalization and dropout layers are used to speed up training convergence and mitigation of overfitting respectively. The input images were resized

to 64 x 64 x 3 pixels and fed into the model with a batch size of 32. Using the ReLU and softmax activation functions, the model was trained with max-pool of stride two. The training accuracy of the model was 99.94%.

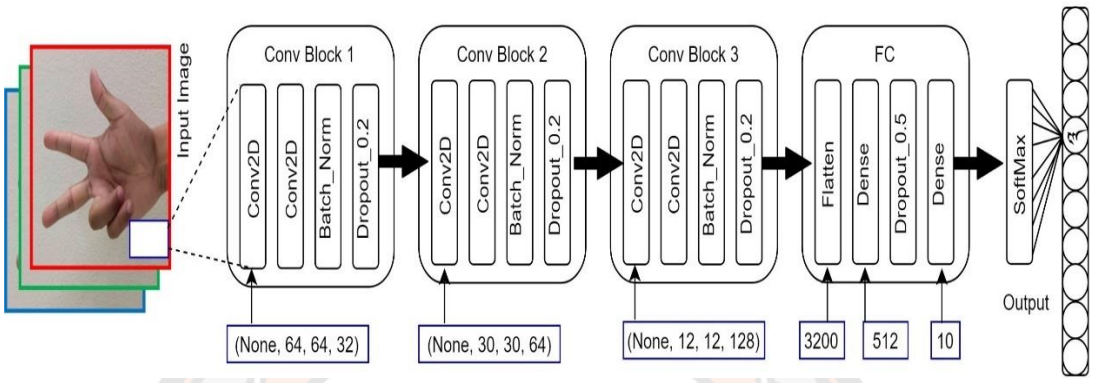


Figure 38 Architecture of VGG-8 network

Source: Saleem and Chishti (2020)

Table 8 VGG-8 network configuration

Type of Layer	Number of Filters	Filter size/stride	Output
Convolution	32	5x5/1	64x64
Convolution	32	5x5/1	64x64
Batch Normalization	-	-	-
Max Pooling	-	2x2/2	32x32
Dropout_20	-	-	-
Convolution	64	3x3/1	30x30
Convolution	64	3x3/1	28x28
Batch Normalization	-	-	-
Max Pooling	-	2x2/2	14x14
Dropout_20	-	-	-
Convolution	128	3x3/1	12x12
Convolution	128	3x3/1	10x10
Batch Normalization	-	-	-
Max Pooling	-	2x2/2	5x5
Dropout_20	-	-	-
	Flatten 3200		
	Dense_512		
	Dropout_50		
	Dense 10		

Different sign language models were evaluated as shown in Table 9. It was observed that the VGGNet with six convolutional layers achieved the highest testing accuracy of 97.62%. It was also observed that the precision, recall, and f1-score for the VGG-8 is the highest. However, the minimum and maximum training time were observed with logistic regression (175 microseconds) and SVM (825 seconds) respectively.

Table 9 Evaluation of different models

Model	Training Time (s)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG-8	294	97.62	98	98	98
LeNet-5	457	91.07	91	91	91
Logistic Regression	0.000175	67.38	67	67	67
SVM	825	70.25	71	70	70
KNN	795	78.95	80	79	79

Source: Wangchuk et al. (2020b)

The accuracy and loss are shown in Figure 39. The model learned in the first 40 epochs but after 50 epochs, learning remained stationary. However, the loss plunged in the first a few epochs but did not decay after 45 epochs. It has been observed that the model stopped learning after 70 epochs.

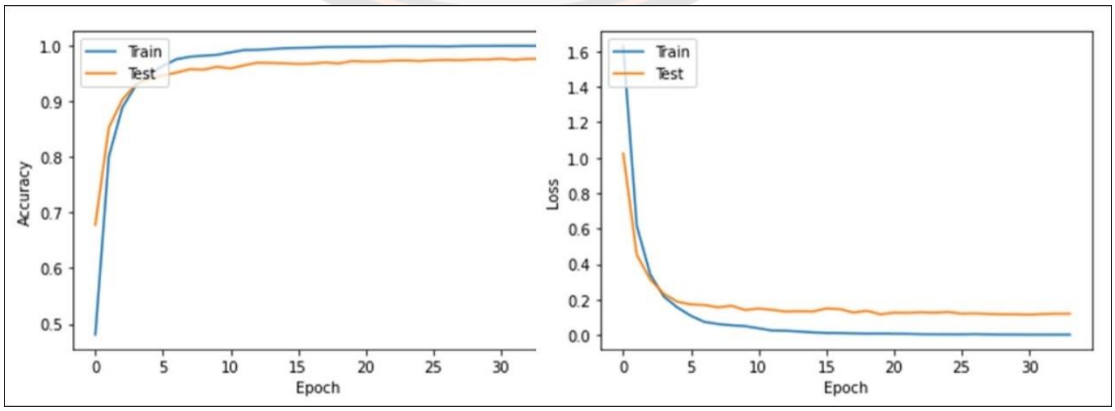


Figure 39 Accuracy and loss of train and test

Source: Wangchuk et al. (2020b)

The confusion matrix evaluates the performance of the classification models. The confusion matrix is shown in Table 10 that shows the correct predictions and rejections of the digits. The digits 0 and 6 have the lowest (two times) and highest (19 times) false positive respectively. However, the highest false negative was observed with digit 7 with 20 times misclassification. It was observed that the lowest misclassification classes were 0, 1, and 3 having misclassified six times each.

Table 10 Digits confusion matrix

		PREDICTED									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
ACTUAL	**0**	394	0	1	1	0	0	0	1	1	2
	1	1	394	1	1	0	0	1	0	0	2
	2	0	2	393	1	0	0	2	0	1	1
	3	0	1	2	394	1	2	0	0	0	0
	4	0	1	1	0	387	1	6	2	2	0
	5	0	0	0	3	2	391	1	0	1	2
	6	0	2	1	0	0	0	391	6	0	0
	7	0	0	3	1	0	2	8	380	2	4
	8	0	0	4	1	0	2	0	0	392	1
	9	1	0	1	3	1	1	1	2	1	389

Table 11 Precision, Recall, and F1-Score for each digit

Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Weight Avg**
Precision	0.99	0.98	0.97	0.97	0.99	0.98	0.95	0.97	0.98	0.97	**0.98**
Recall	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.95	0.98	0.97	**0.98**
F1-Score	0.99	0.98	0.97	0.98	0.98	0.98	0.97	0.96	0.98	0.97	**0.98**

The percentage of precision, recall, and F1-score for each class are shown in Table 11. The lowest precision and recall for 6 and 7 was 95%. The minimum percentage of the F1-score was 96% for class 7. Overall, the weighted average raised to 98%. The precision, recall, and F1-score are defined as,

$$precision = \frac{TP}{TP + FP} \tag{22}$$

$$recall = \frac{TP}{TP + FN} \tag{23}$$

$$F1\_score = \frac{2 * precision * recall}{precision + recall} \tag{24}$$

Where TP stands for true positive and FP for false positive. FN represents a false negative.

The VGG-8 model was trained and saved for deployment in the local system. The size of the model is 22.949 MB. Using TensorFlow as backend and OpenCV, the model was deployed in the local system (laptop) for real-time detection and prediction of BSL digits as shown in Figures 53 and 54. The resolution of the images taken for prediction was 64x64x3 pixels. It was observed that the prediction was better indoor compared to outdoor. However, the room must be bright. The model was tested with different backgrounds and found that varying backgrounds do not hinder the prediction. The model was robust.

4.3 BSL Alphabets Detection and Recognition

There are 30,000 RGB images of size 200x200 in the BSL alphabet dataset. The dataset is further partitioned into train and test sets that consisted of 80% and 20% images respectively. The images are further resized to 64x64x3 pixels at the time of training the model. The images are resized to reduce the loading and training time. In the previous section, the VGG-8 network outperformed various sign language models with the BSL digit dataset. Therefore, VGG-8 is used with the alphabet dataset as well. The convolutional blocks extract features and fully connected blocks classify images into one of the classes as shown in Figure 40. The CNN and VGG-8 were compared and evaluated with different parameters fine-tuning as shown in Table 12. Different models with varying filter sizes and dense layers were trained by either inclusion or exclusion of batch normalization and augmentation. However, VGG-8 with batch normalization and augmentation obtained the best train and test accuracy of 98.27% and 99.72% respectively.

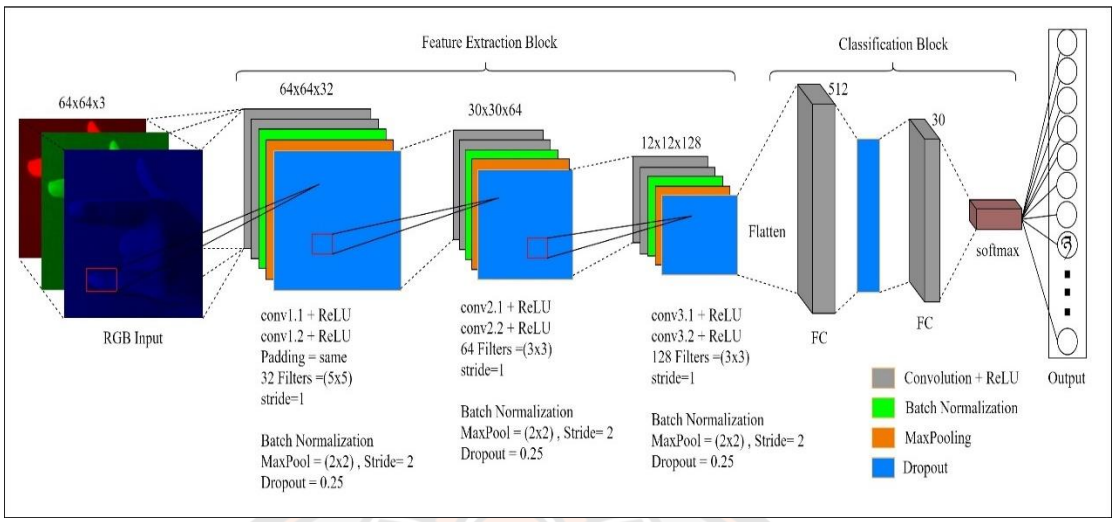


Figure 40 Architecture of CNN model

Source: Wangchuk et al. (2020b)

Table 12 Accuracy analysis with different parameters

Layers	Number of Filters	BatchNorm	Dropout	Augmentation	FC	Train Acc (%)	Test Acc (%)
1	32	no	yes	no	2	16.31	38.35
	32	yes	yes	yes	2	62.08	86.22
2	32, 64	no	yes	no	2	94.93	95.95
	32, 64	yes	yes	yes	3	38.14	45.20
4	32, 64, 128, 256	no	yes	no	3	96.55	98.58
	32, 64, 128, 256	yes	yes	yes	3	94.72	99.55
6	32, 32, 64, 64, 128, 128	yes	yes	yes	2	98.27	99.72

Source: Wangchuk et al. (2020a)

The network with one convolutional layer without batch normalization and augmentation gave the worst train and test accuracy of 16.31% and 38.35% respectively. However, accuracy increased to 62.08% and 86.22% with the addition of batch normalization and augmentation in the network. The huge difference between train and test accuracy clearly showed the underfitting during training the model. The network with two convolutional layers performed better than one convolutional layer. Furthermore, the difference between train (94.93%) and test (95.95%) accuracy is reduced as shown in Table 12. It was observed that the network learns better with the addition of convolutional layers with the different filter sizes, batch normalization, and

augmentation. However, with the addition of more layers, learnable parameters increase and take more time to train the network.

It is clear from Table 12 that the testing accuracies are greater than training accuracies. Generally, it is expected that the training accuracy should be slightly higher than the testing accuracy. It was found that the implementation of different dropout ratios in the networks resulted in lower training accuracy compared with the testing accuracy. The desired neurons are not activated with dropout implementation. For example, the dropout ratios used are 0.3 and 0.5 in different network layers and dense layers, 30% and 50% of the neurons would be dropped in respective layers and would not participate in the training of the network.

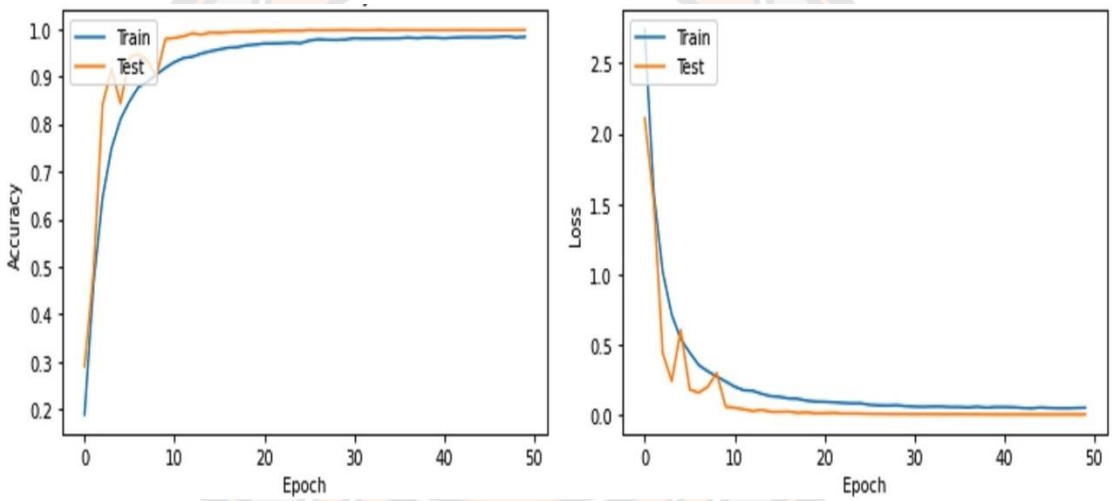


Figure 41 Analysis of accuracy and loss with varying epoch

Figure 41 illustrates the accuracy and loss of training and testing between 50 epochs. It was observed that the accuracy of both train and test increased sharply in the first 10 epochs. However, learning stopped from 25 epochs and maintained constant accuracy. The loss plummeted in the beginning but stopped declining after having reached 20 epochs. Overall, the accuracy increased and loss decreased without showing overfitting nor underfitting.

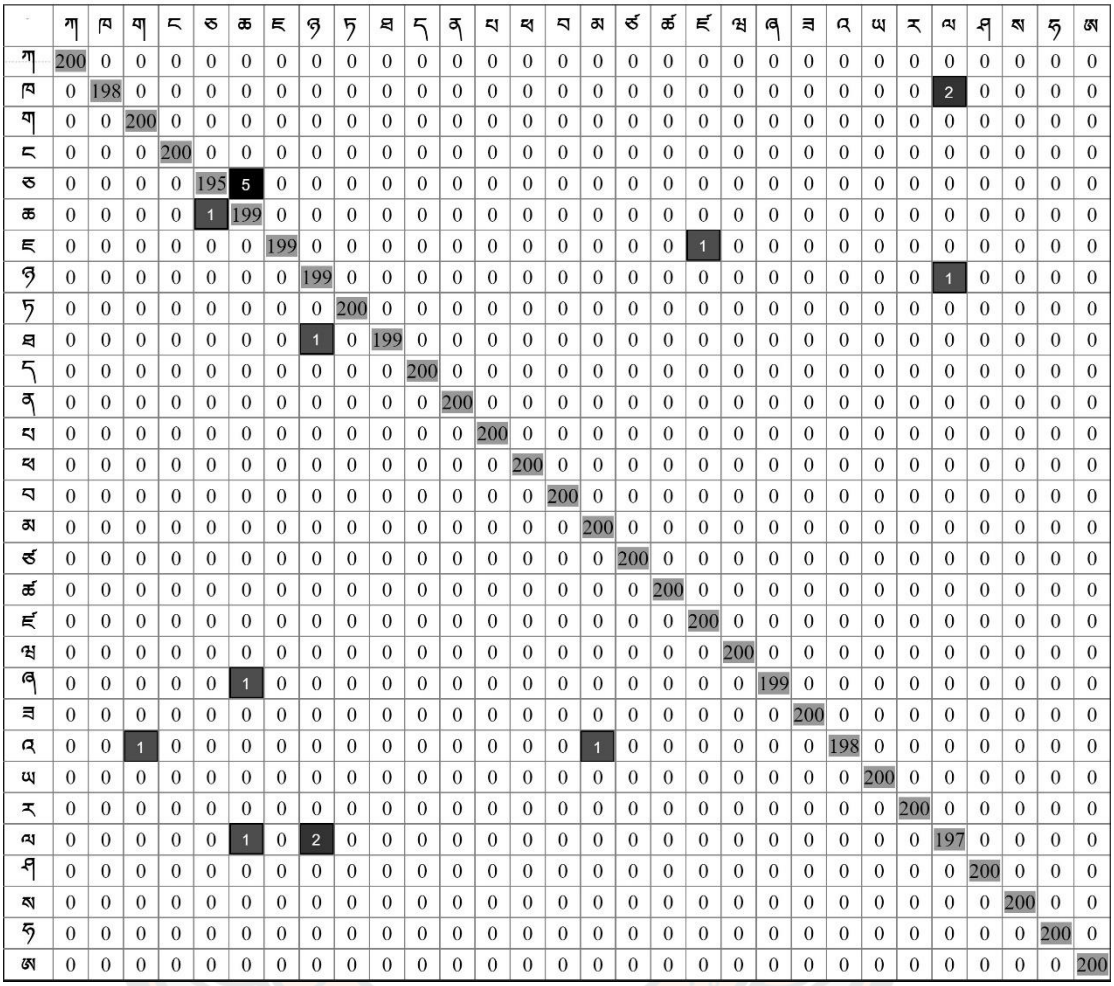


Figure 42 Alphabets' confusion matrix

Source: Wangchuk et al. (2020a)

The confusion matrix is shown in Figure 42. The highest false positive was with the class ཆ (chha) and the highest false negative was observed with the class ཅ (cha). However, the alphabet "cha" is 5 times misclassified as "chha" which was the highest misclassified class. The alphabets ཁ (kha) and ལ (la) are the second-highest misclassified classes. Overall, both false positives and false negatives were 17.

The precision, recall, and F1-score are shown in Table 13. It was analyzed that the lowest precision and recall for class "chha" and "cha" respectively was 97%. However, the weighted average for the precision, recall, and F1-score was 100%.

Table 13 Tabulation of precision, recall, and F1-score of alphabets

Class	Precision	Recall	F1-Scores	Class	Precision	Recall	F1-Scores
ka	1.00	1.00	1.00	**ma**	1.00	1.00	1.00
kha	1.00	0.99	0.99	**tsa**	1.00	1.00	1.00
ga	1.00	1.00	1.00	**tsha**	1.00	1.00	1.00
nga	1.00	1.00	1.00	**dza**	1.00	1.00	1.00
cha	0.99	0.97	0.98	**wa**	1.00	1.00	1.00
chha	0.97	0.99	0.98	**zha**	1.00	0.99	1.00
ja	1.00	0.99	1.00	**za**	1.00	1.00	1.00
nya	0.99	0.99	0.99	**`a**	1.00	0.99	0.99
ta	1.00	1.00	1.00	**ya**	1.00	1.00	1.00
tha	1.00	0.99	1.00	**ra**	1.00	1.00	1.00
da	1.00	1.00	1.00	**la**	0.98	0.98	0.98
na	1.00	1.00	1.00	**sha**	1.00	1.00	1.00
pa	1.00	1.00	1.00	**sa**	1.00	1.00	1.00
pha	1.00	1.00	1.00	**ha**	1.00	1.00	1.00
ba	1.00	1.00	1.00	**a**	1.00	1.00	1.00
Weighted average					**1.00**	**1.00**	**1.00**

The trained model of size 23.027 MB was saved and deployed using laptop with OpenCV and TensorFlow. Similar to BSL digits dataset, image of size 64x64x3 pixels were used to detect and recognize BSL alphabets in real-time using webcam as shown in Figures 51 and 52. The detection results were better indoor rather than outdoor. However, indoor should be bright.

4.4 Alphanumeric Detection and Recognition

There are 60,000 images in an alphanumeric BSL dataset. The increased number of images takes a longer time to train the networks. Furthermore, deeper layers increase the complexity of the network. Therefore, pickle is used for the serialization of images that converts them into byte streams. The serialized objects are faster to transfer or read images during the training. During serialization, images are further resized to 128x128x3 pixels that are given as input to the model. The modification of ResNet50 with 43 convolutional (Conv) layers obtained the best train and test accuracy of 100% and 98.38% respectively.

The deep network exhibits better accuracy compared with a fewer number of Conv layers. Table 14 displays the evaluation of an alphanumeric dataset with deep learning. The LeNet5 with a batch size of 512 gained 100% training accuracy but the

validation of 83.08% showed that the network is overfitting. However, AlexNet, VGGNet, and ResNet learned without overfitting nor underfitting. The highest train accuracy of 100% was observed with LeNet and ResNet but formal overfitted the network during the training. Similarly, the highest validation accuracy of 98.44% was observed with AlexNet.

Table 14 Evaluation of deep learning models on BSL dataset

Model	Batch Size	Epochs	Time(s)	Train Acc.	Validate Acc.
LeNet5	512	97	1648	100	83.08
AlexNet	128	37	861	99.92	98.44
VGG	256	55	2133	99.90	97.49
ResNet	128	129	16504	100	98.38

Table 15 illustrates the configuration of the ResNet-44. The network was divided into four stages. The first stage contained one Conv layer without Conv-block and identity-block. However, the remaining stages consisted of one Conv-block each followed by two, three, and five identity-blocks respectively as shown in Figure 44. Further, the Conv-block and identity-block comprised of four and three Conv layers respectively as shown in Figure 43. Therefore, the network was designed with 43 convolutional layers and a dense layer (ResNet-44) as shown in Figure 44.

Table 15 Number of convolutional layers in ResNet-44

stage	conv	conv-block	Identity-Block
1	1	0	0
2	0	1	2
3	0	1	3
4	0	1	5
Total Conv	**1**	**12**	**30**

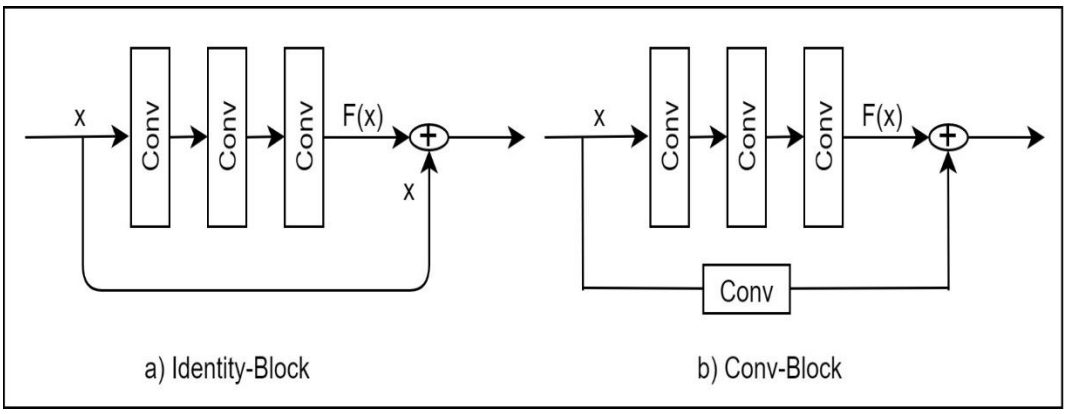


Figure 43 Skip connection with identity-block and Conv-block

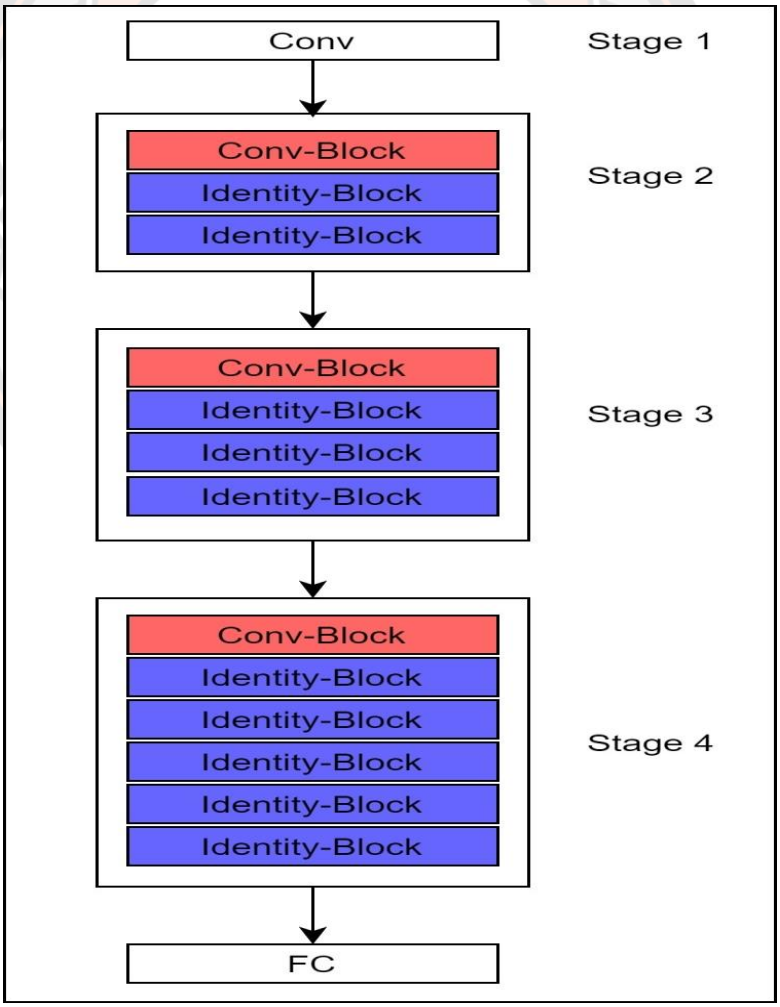


Figure 44 The ResNet-43 architecture

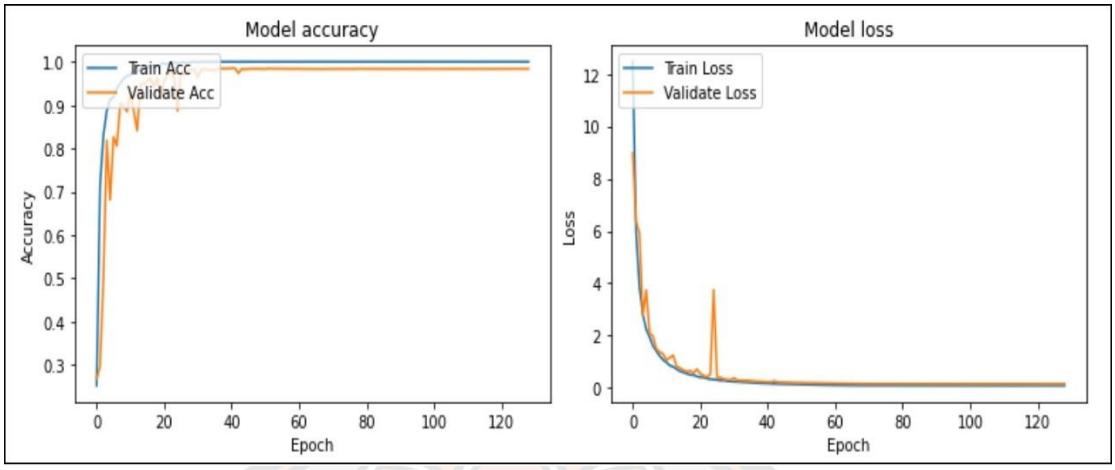


Figure 45 Graph showing model accuracy and loss with varying epochs

Figure 45 shows the accuracy and loss of the model with varying epochs. The accuracy increased sharply in the first 10 epochs and was steady after 20 epochs. Similarly, loss plummeted in the first 5 epochs. However, loss stopped declining after 40 epochs. Overall, the model demonstrated 100% training accuracy with the minimum error. The evaluation of the classification is further illustrated in the confusion matrix in Figures 46 and 47. The highest false positive was 22 with class EIGHT (38). However, both the classes CHHA (5) and SEVEN (37) have the highest false negative of 18 each. The classes CHHA and SEVEN were nine times misclassified as CHA and EIGHT respectively as shown in Figure 46. A total of 18000 images were used as the validation set and of which 296 images were misclassified.

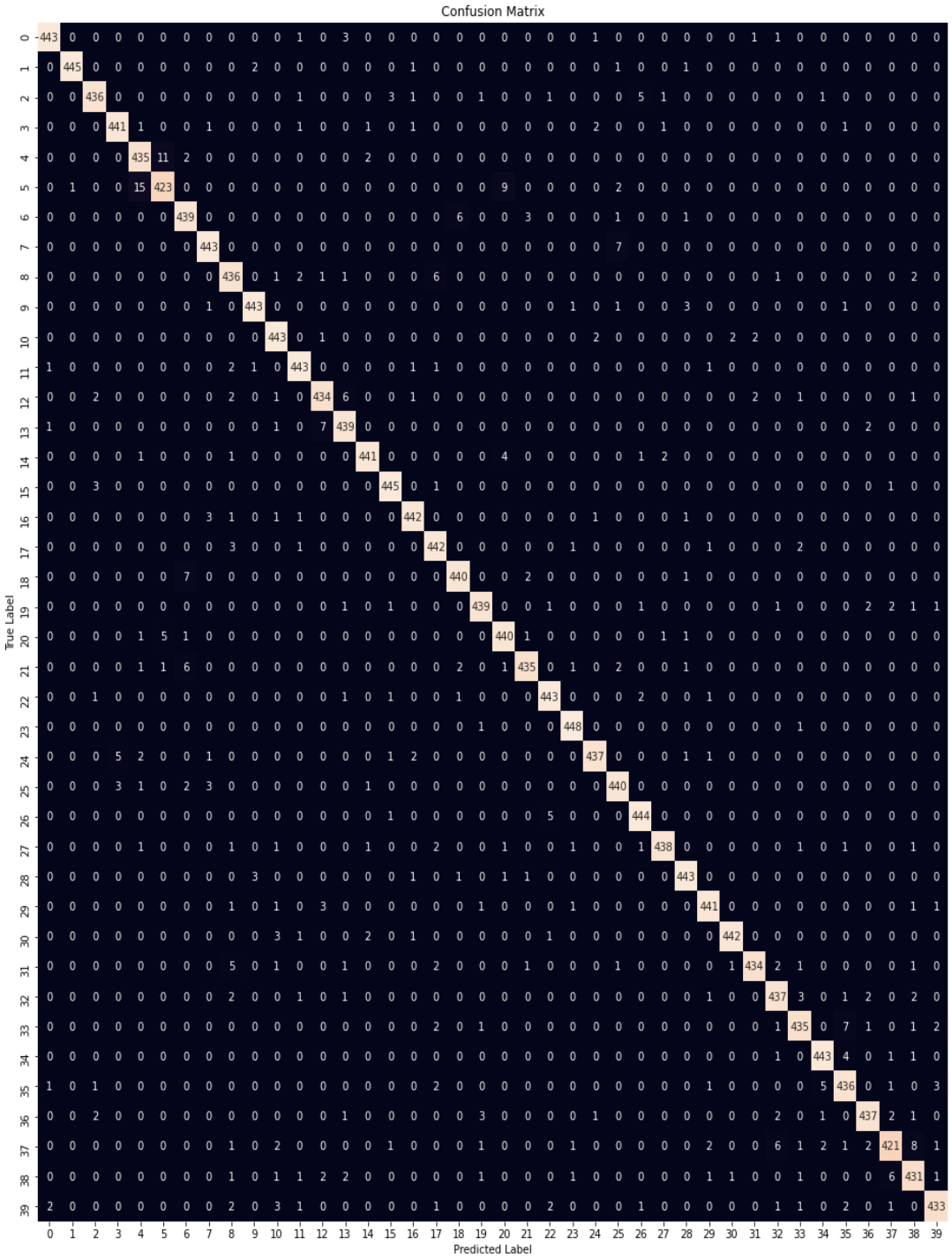


Figure 46 ResNet-43 confusion matrix

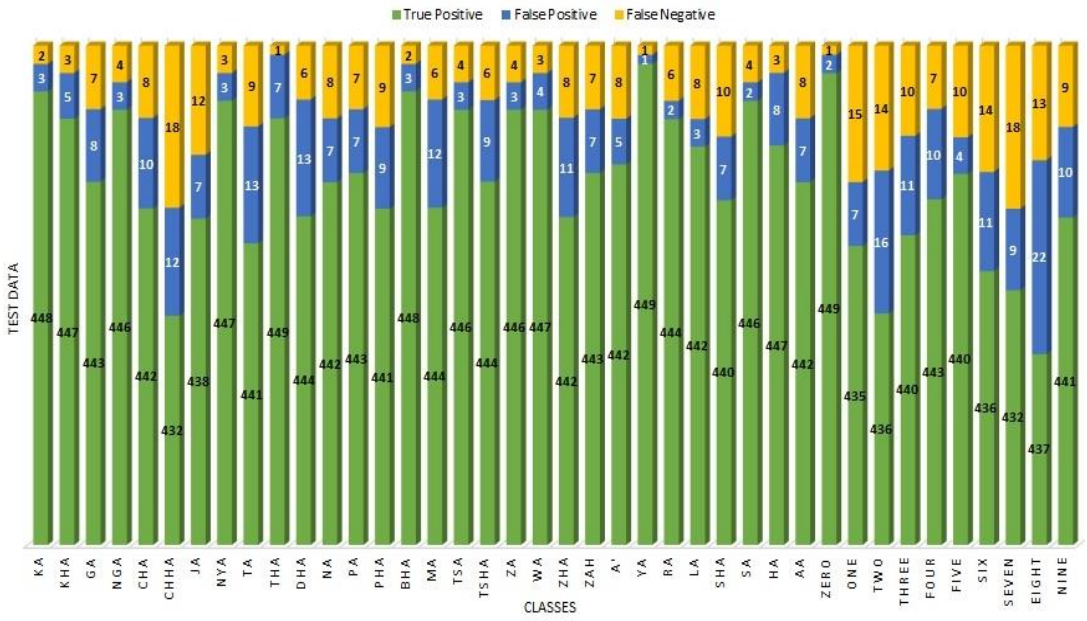


Figure 47 Graph displaying false positive and negative over true positive

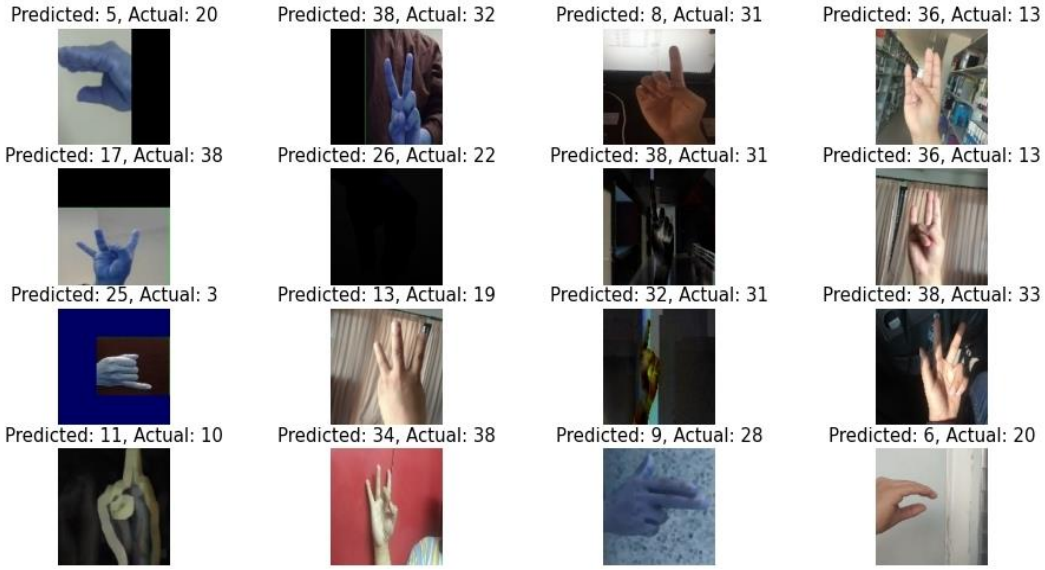


Figure 48 Analysis of predicted class with actual class

Figures 46, 47, and 48 shows the classification result of predicted classes and actual classes. The misclassification was due to the similar shapes, images collected with varying angles, and augmentation. Figure 49 shows similar shapes that were misclassified in Figure 48. For example, classes ZHA (ঝ) and CHHA (ছ) or classes A’

(འ) and SHA (ཤ) are exhibiting similar shape. Similarly, Figure 49 illustrates similar features.

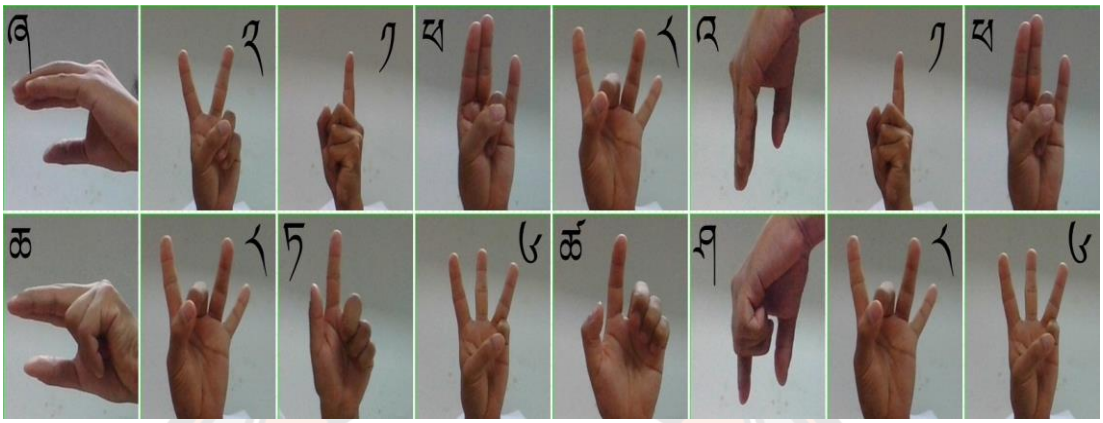


Figure 49 First row displays the actual class and the second row shows predicted class

The second reason for the misclassification was due to the varying angles in the dataset. The images or videos were recorded with different angles and distances from the actors to add variations to the dataset. The angles during image recording play a vital role. For example, in Figure 49, the classes TWO (ཉ) and EIGHT (ཇ) or PHA (ཕ) or SIX (ཆ) become similar due to the rotation of the camera during image collection. Finally, misclassification was because of the width and height shift during augmentation. Figure 50 shows class LA (ལ) changed to class NYA (ཉ) due to width shift by a certain pixel.

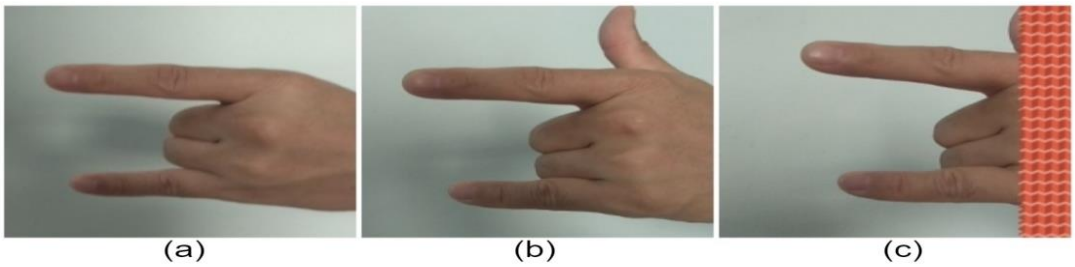


Figure 50 Augmentation changes classes: (a) La, (b) Nya, (c) with shift converted (a) to (b)

Source: Wangchuk et al. (2020a)

The VGGNet and ResNet models were deployed for real-time detection and recognition of BSL alphanumeric using a laptop webcam. The size of the VGG and ResNet models were 108.886 MB and 38.168 MB Respectively. The models take 128x128x3 resolution images for detection and recognition. Similar to BSL Digits and Alphabet datasets, the BSL Alphanumeric models performed better in closed-door compared to outdoor but the room must be bright.

4.5 Real-time Detection and Recognition Using Webcam

The models were tested in real-time using the webcam. The trained models were saved and deployed with the TensorFlow, VS code, python, and OpenCV using the laptop. The webcam of the laptop reads the image in real-time and inferences with the trained model to predict one of the classes. The models were evaluated both indoor and outdoor with varying backgrounds. It was observed that the predictions were better indoor but the room must be bright. It was also observed that varying backgrounds did not affect classification. Figures 51 and 52 show real-time prediction of alphabets Zha (ཟ) and La (ལ) respectively. Figures 53 and 54 show digits 3 (༣) and 9 (༩) respectively. The outputs of the model were both in Dzongkha and English text. The English output validates the Dzongkha output as shown in the figures.

Figure 51 Real-time detection and recognition of Zha

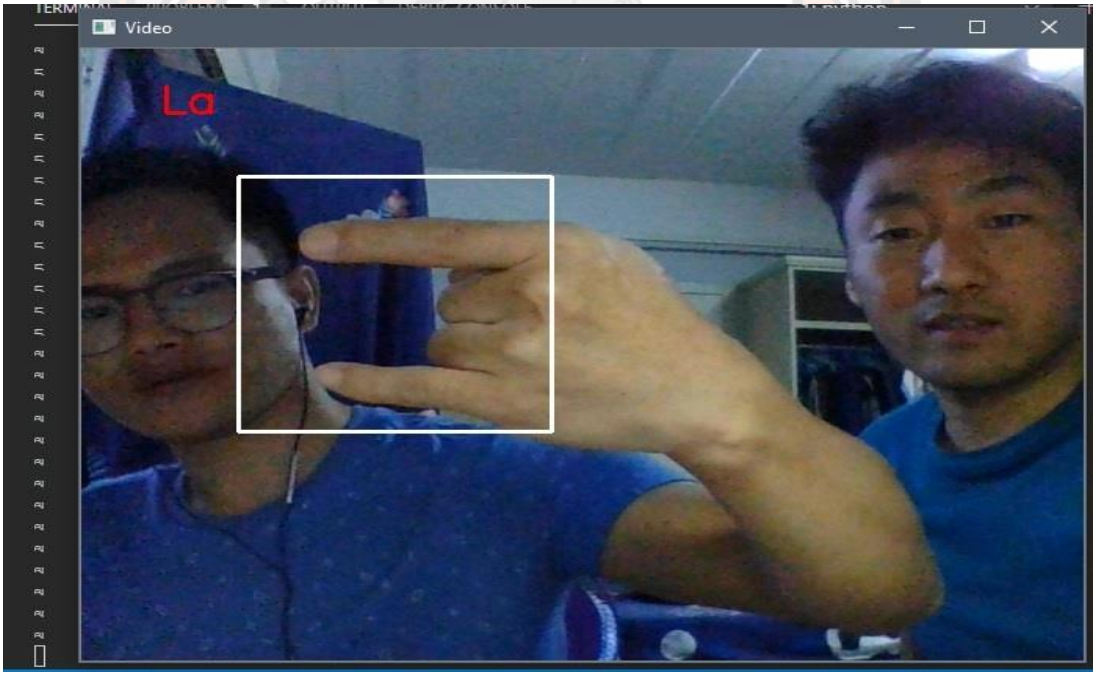


Figure 52 Real-time detection and recognition of La

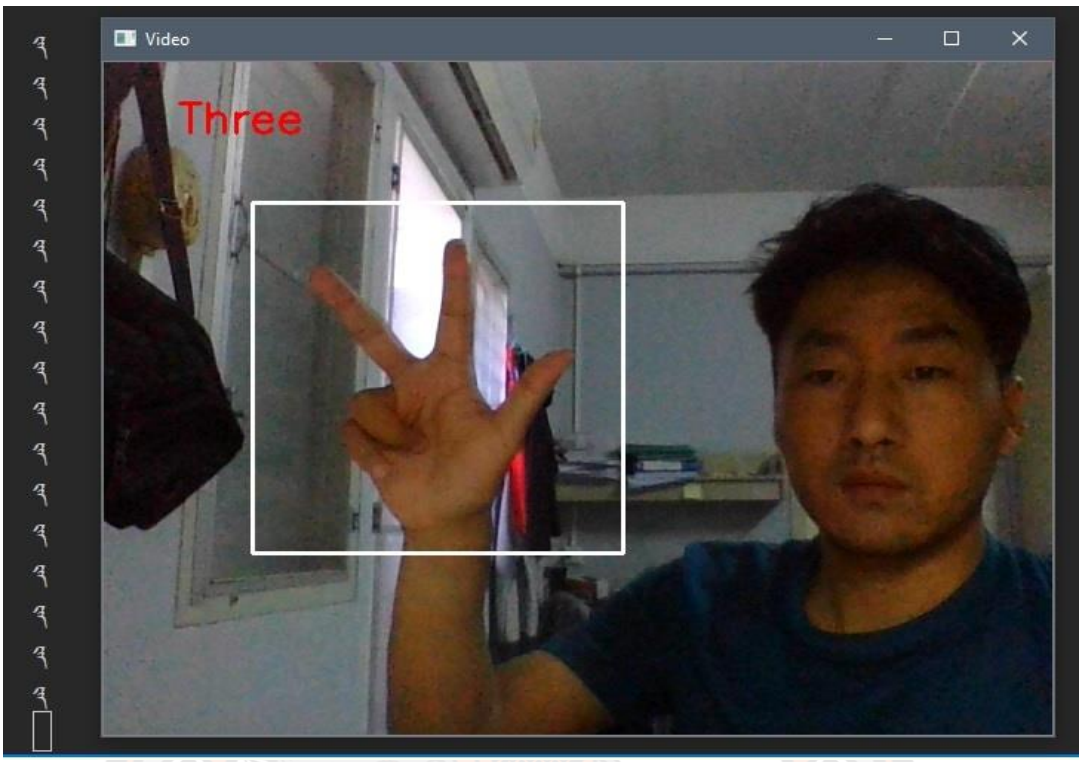


Figure 53 Real-time detection and recognition of Three

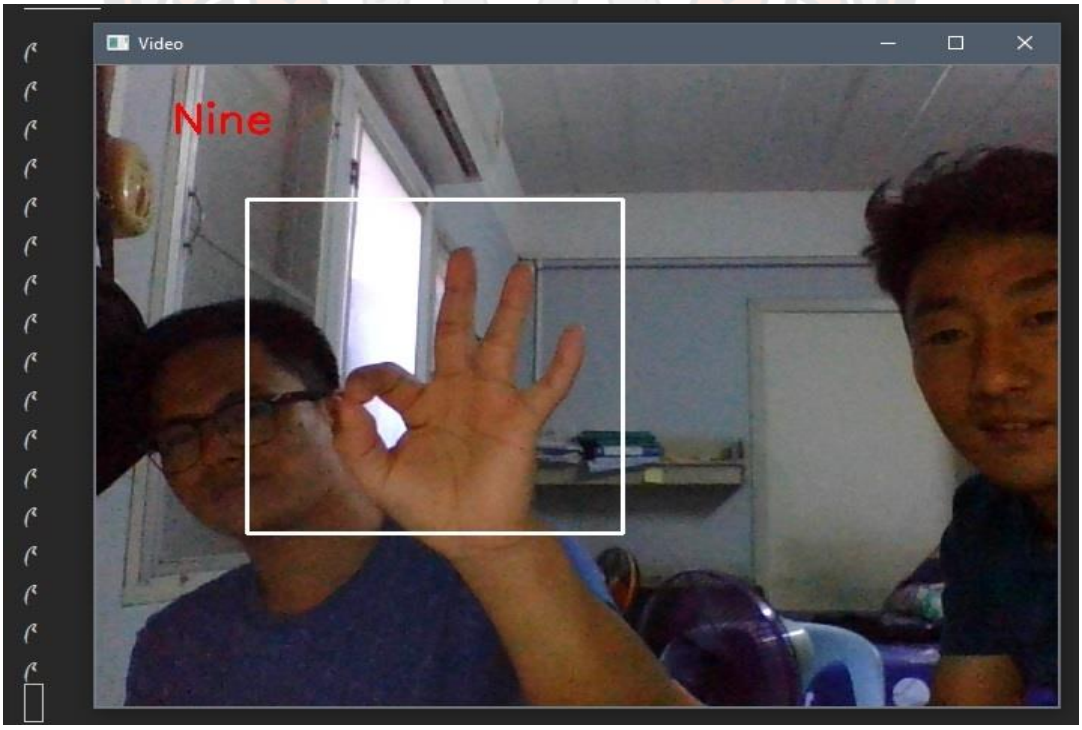


Figure 54 Real-time detection and recognition of Nine

CHAPTER V
CONCLUSION

5.1 Introduction

The background of the study was presented in Chapter I followed by literature reviews in Chapter II. The methodology details were explained in Chapter III and discussion of results in chapter IV followed by conclusion in Chapter V. In the conclusion section, a summary of the thesis is presented first followed by limitations and future works.

5.2 Summary

The study examined different sign languages that have implemented state-of-the-art machine learning algorithms as discussed in Chapter II. It was found that sign languages are different in each country. However, there are some common signs between sign languages such as numbers. Besides, many countries have both static and dynamic gestures in sign languages. Nevertheless, the BSL sign language does not have a static sign except alphabets and digits. The BSL word requires either one hand or two hand gestures. The main purpose of the study was to build a suitable machine learning model to detect and recognize BSL hand-shaped alphabets and digits. Furthermore, first-ever BSL hand-shaped alphabets and digits dataset was curated and created. There are three different datasets created: 20,000 digits dataset, 30,000 alphabets dataset, and 60,000 alphanumeric datasets. The appropriate machine learning models are tested with these BSL datasets.

Based on the literature reviews in Chapter II, BSL datasets were evaluated with different sign language models. The CNN algorithm was used to extract features from the images with parameters fine-tuning to suit BSL datasets. For digit detection and recognition, six convolutional layers with the batch normalization and dropout outperformed LeNet, SVM, KNN, and logistic regression with 99.94% and 97.62% training and testing respectively. Similarly, for alphabets recognition, the different number of convolutional layers and either inclusion or exclusion of batch normalization and dropout were evaluated. The VGG-8 network that consisted of six convolutional

layers obtained the best training and validation accuracy of 98.27% and 99.72% respectively. However, alphanumeric used deeper layers that were built similar to ResNet-50 with 43 convolutional layers obtained better training accuracy of 100% compared to LeNet, AlexNet, and VGG. The deeper layers learn better but take more time to train the model.

In this study, the possibility of Computer Vision applications with the BSL was studied. This research is the first attempt to introduce Computer Vision with the Bhutanese Sign Language. Different sign language algorithms were tested and evaluated with hand-shaped alphabets and digits. It was observed that multiclass with a huge number of images take a longer time to train the model. However, serialization of images (byte streams) using pickle reduced the training time. It was observed that the Bhutanese sign language applications can be developed using machine learning models.

5.3 Limitation of the study

The BSL dataset shows high accuracy in training the models as discussed in Chapter IV. However, there are limitations in data acquisition and model training. The limitations are given below:

1. Smartphones and webcam were also used for recording videos and capturing images respectively.
2. Images captured with varying angles gave false classes impression. For example, PA and PHA have similar shapes. The angles play an important role.
3. Augmentation gives variation to the dataset. However, width and height shift in the BSL dataset gave an incorrect classification of classes. For example, LA and NYA classes. The width shift of NYA by a certain pixel makes LA.
4. The algorithms implemented were having difficulties in correctly detecting and recognizing classes CHHA and SEVEN. The class CHHA was misclassified as CHA and SEVEN as EIGHT and vice-versa is true.
5. Cannot train a model with a huge number of images in the dataset. This increases the training time and requires higher system specifications.

5.4 Future Work

In the future, high-resolution images can be collected without varying angles. This would classify similar shape classes with better accuracy. The number of actors can be increased with the inclusion of children's hand-shaped data. In addition, dynamic gesture recognition using LSTM or 3D CNN can be studied by creating a video-based BSL dataset. Furthermore, the study can be extended to gesture to voice translation with android or desktop-based application development.

REFERENCES

REFERENCES

Ahsan, M. R., Ibrahimy, M. I., & Khalifa, O. O. (2011). *Electromygraphy (EMG) signal based hand gesture recognition using artificial neural network (ANN).* Paper presented at the 2011 4th International Conference on Mechatronics (ICOM).

Akl, A., & Valaee, S. (2010). *Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing.* Paper presented at the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing.

Akmeliawati, R., Dadgostar, F., Demidenko, S., Gamage, N., Kuang, Y. C., Messom, C., . . . SenGupta, G. (2009). *Towards real-time sign language analysis via markerless gesture tracking.* Paper presented at the 2009 IEEE Instrumentation and Measurement Technology Conference.

Alvi, A. K., Azhar, M. Y. B., Usman, M., Mumtaz, S., Rafiq, S., Rehman, R. U., & Ahmed, I. (2004). Pakistan sign language recognition using statistical template matching. *International Journal of Information Technology, 1*(1), 1-12.

Ariesta, M. C., Wiryana, F., & Kusuma, G. P. (2018). A Survey of Hand Gesture Recognition Methods in Sign Language Recognition. *Pertanika Journal of Science & Technology, 26*(4).

Attenberg, J., Melville, P., Provost, F., & Saar-Tsechansky, M. (2011). Selective data acquisition for machine learning. *Cost-sensitive machine learning*, 101.

Azar, S. G., & Seyedarabi, H. (2020). Trajectory-based recognition of dynamic Persian sign language using hidden Markov model. *Computer Speech & Language, 61*, 101053.

Babu, U. R., Venkateswarlu, Y., & Chintha, A. K. (2014). *Handwritten digit recognition using K-nearest neighbour classifier.* Paper presented at the 2014 World Congress on Computing and Communication Technologies.

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., & Baskurt, A. (2011). *Sequential deep learning for human action recognition.* Paper presented at the International workshop on human behavior understanding.

Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). *Understanding batch normalization.* Paper presented at the Advances in neural information processing systems.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery, 2*(2), 121-167.

Camastra, F., & De Felice, D. (2013). LVQ-based hand gesture recognition using a data glove *Neural Nets and Surroundings* (pp. 159-168): Springer.

Chai, X., Liu, Z., Yin, F., Liu, Z., & Chen, X. (2016). *Two streams recurrent neural networks for large-scale continuous gesture recognition.* Paper presented at the 2016 23rd International Conference on Pattern Recognition (ICPR).

Chen, F.-S., Fu, C.-M., & Huang, C.-L. (2003). Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image and vision computing, 21*(8), 745-758.

Cheng, H.-D., Jiang, X. H., Sun, Y., & Wang, J. (2001). Color image segmentation: advances and prospects. *Pattern recognition, 34*(12), 2259-2281.

Cheok, M. J., Omar, Z., & Jaward, M. H. (2019). A review of hand gesture and sign language recognition techniques. *International Journal of Machine Learning and Cybernetics, 10*(1), 131-153.

Cheung, W., & Hamarneh, G. (2009). n-SIFT: n-Dimensional Scale Invariant Feature Transform. *IEEE Transactions on Image Processing, 18*(9), 2012-2021.

Clark, P., & Niblett, T. (1987). *Induction in Noisy Domains.* Paper presented at the EWSL.

Conseil, S., Bourennane, S., & Martin, L. (2007). *Comparison of Fourier descriptors and Hu moments for hand posture recognition.* Paper presented at the 2007 15th European Signal Processing Conference.

Dai, C., Liu, X., & Lai, J. (2020). Human action recognition using two-stream attention based LSTM networks. *Applied Soft Computing, 86*, 105820.

Dardas, N. H., & Georganas, N. D. (2011). Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and measurement, 60*(11), 3592-3607.

Deng, Y., & Manjunath, B. (2001). Unsupervised segmentation of color-texture regions in images and video. *IEEE transactions on pattern analysis and machine intelligence, 23*(8), 800-810.

Dorji, K.-O. (2008). A brief history of Bhutan House in Kalimpong. *Journal of Bhutan Studies, 19*(2), 9-33.

Edel, M., & Köppe, E. (2016). *Binarized-blstm-rnn based human activity recognition.* Paper presented at the 2016 International conference on indoor positioning and indoor navigation (IPIN).

Elmezain, M., Al-Hamadi, A., Appenrodt, J., & Michaelis, B. (2009). A hidden markov model-based isolated and meaningful hand gesture recognition. *International Journal of Electrical, Computer, and Systems Engineering, 3*(3), 156-163.

Funaya, H., & Ikeda, K. (2012). *A statistical analysis of soft-margin support vector machines for non-separable problems.* Paper presented at the The 2012 International Joint Conference on Neural Networks (IJCNN).

Ge, Z., Song, Z., Ding, S. X., & Huang, B. (2017). Data mining and analytics in the process industry: The role of machine learning. *Ieee Access, 5*, 20590-20616.

Gupta, B., Shukla, P., & Mittal, A. (2016). *K-nearest correlated neighbor classification for Indian sign language gesture recognition using feature fusion.* Paper presented at the 2016 International Conference on Computer Communication and Informatics (ICCCI).

Gurjal, P., & Kunnur, K. (2012). Real time hand gesture recognition using SIFT. *International Journal of Electronics and Electrical Engineering, 2*(3), 19-33.

Han, J., Shao, L., Xu, D., & Shotton, J. (2013). Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics, 43*(5), 1318-1334.

Han, S., Qubo, C., & Meng, H. (2012). *Parameter selection in SVM with RBF kernel function.* Paper presented at the World Automation Congress 2012.

Haseeb, M. A. A., & Parasuraman, R. (2017). Wisture: Rnn-based learning of wireless signals for gesture recognition in unmodified smartphones. *arXiv preprint arXiv:1707.08569*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

He, W., Wu, K., Zou, Y., & Ming, Z. (2015). *Wig: Wifi-based gesture recognition system.* Paper presented at the 2015 24th International Conference on Computer Communication and Networks (ICCCN).

Hu, Y., Wong, Y., Wei, W., Du, Y., Kankanhalli, M., & Geng, W. (2018). A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition. *PloS one, 13*(10).

Huang, C.-l., & Jeng, S.-H. (2001). A model-based hand gesture recognition system. *Machine vision and applications, 12*(5), 243-258.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). *Densely connected convolutional networks.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). *Deep networks with stochastic depth.* Paper presented at the European conference on computer vision.

Iannizzotto, G., & Vita, L. (2000). Fast and accurate edge-based segmentation with no contour smoothing in 2-D real images. *IEEE Transactions on Image Processing, 9*(7), 1232-1237.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence, 35*(1), 221-231.

Kar, A., Rai, N., Sikka, K., & Sharma, G. (2017). *Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Kariniotakis, G. (2017). *Renewable energy forecasting: from models to applications*: Woodhead Publishing.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2019). A survey of the recent architectures of deep convolutional neural networks. *arXiv preprint arXiv:1901.06032*.

Kim, J., Kwon Lee, J., & Mu Lee, K. (2016). *Deeply-recursive convolutional network for image super-resolution.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Kim, J., Mastnik, S., & André, E. (2008). *EMG-based hand gesture recognition for realtime biosignal interfacing.* Paper presented at the Proceedings of the 13th international conference on Intelligent user interfaces.

Ko, S.-K., Son, J. G., & Jung, H. (2018). *Sign language recognition with recurrent neural network using human keypoint detection.* Paper presented at the Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems.

Kotsiantis, S. B., Pierrakeas, C., & Pintelas, P. E. (2003). *Preventing student dropout in distance learning using machine learning techniques.* Paper presented at the International conference on knowledge-based and intelligent information and engineering systems.

Koul, M., Patil, P., Nandurkar, V., & Patil, S. (2016). Sign language recognition using leap motion sensor. *International Research Journal of Engineering and Technology (IRJET), 3*(11), 322-325.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* Paper presented at the Advances in neural information processing systems.

Kumar, G., & Bhatia, P. K. (2014). *A detailed review of feature extraction in image processing systems.* Paper presented at the 2014 Fourth international conference on advanced computing & communication technologies.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. Backpropagation Applied to Handwritten Zip Code Recognition, 1989. *Dostupné z: http://yann. lecun. com/exdb/publis/pdf/lecun-89e. pdf*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation, 1*(4), 541-551.

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

Lefebvre, G., Berlemont, S., Mamalet, F., & Garcia, C. (2013). *BLSTM-RNN based 3D gesture classification.* Paper presented at the International conference on artificial neural networks.

Li, Y.-T., & Wachs, J. P. (2014). HEGM: A hierarchical elastic graph matching for hand gesture recognition. *Pattern Recognition, 47*(1), 80-88.

Lindeberg, T. (2012). Scale invariant feature transform.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision, 60*(2), 91-110.

Mahbub, U., Imtiaz, H., Roy, T., Rahman, M. S., & Ahad, M. A. R. (2013). A template matching approach of one-shot-learning gesture recognition. *Pattern Recognition Letters, 34*(15), 1780-1788.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), 115-133.

Mingqiang, Y., Kidiyo, K., & Joseph, R. (2008). A survey of shape feature extraction techniques. *Pattern recognition, 15*(7), 43-90.

Mohandes, M., Aliyu, S., & Deriche, M. (2014). *Arabic sign language recognition using the leap motion controller.* Paper presented at the 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE).

Mufarroha, F. A., & Utaminingrum, F. (2017). Hand gesture recognition using adaptive network based fuzzy inference system and K-nearest neighbor. *International Journal of Technology, 8*(3), 559-567.

Murthy, G., & Jadon, R. (2009). A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management, 2*(2), 405-410.

Murthy, G., & Jadon, R. (2010). *Hand gesture recognition using neural networks.* Paper presented at the 2010 IEEE 2nd International Advance Computing Conference (IACC).

Naidu, M., Kumar, P. R., & Chiranjeevi, K. (2018). Shannon and fuzzy entropy based evolutionary image thresholding for image segmentation. *Alexandria engineering journal, 57*(3), 1643-1655.

Nikam, A. S., & Ambekar, A. G. (2016). *Sign language recognition using image based hand gesture recognition techniques.* Paper presented at the 2016 Online International Conference on Green Engineering and Technologies (IC-GET).

Nimbarte, N. M., & Mushrif, M. M. (2010). *Multi-level thresholding algorithm for color image segmentation.* Paper presented at the 2010 Second International Conference on Computer Engineering and Applications.

Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence, 60*, 97-116.

Palvanov, A., & Im Cho, Y. (2018). Comparisons of deep learning algorithms for MNIST in real-time environment. *International Journal of Fuzzy Logic and Intelligent Systems, 18*(2), 126-134.

Panwar, M. (2012). *Hand gesture recognition based on shape parameters.* Paper presented at the 2012 International Conference on Computing, Communication and Applications.

Patle, A., & Chouhan, D. S. (2013). *SVM kernel functions for classification.* Paper presented at the 2013 International Conference on Advances in Technology and Engineering (ICATE).

Priyal, S. P., & Bora, P. K. (2013). A robust static hand gesture recognition system using geometry based normalizations and Krawtchouk moments. *Pattern Recognition, 46*(8), 2202-2219.

Rahaman, M. A., Jasim, M., Ali, M. H., & Hasanuzzaman, M. (2014). *Real-time computer vision-based Bengali sign language recognition.* Paper presented at the 2014 17th International Conference on Computer and Information Technology (ICCIT).

Raheja, J., Mishra, A., & Chaudhary, A. (2016). Indian sign language recognition using SVM. *Pattern Recognition and Image Analysis, 26*(2), 434-441.

Rastgoo, R., Kiani, K., & Escalera, S. (2020). Hand sign language recognition using multi-view hand skeleton. *Expert Systems with Applications*, 113336.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation, 29*(9), 2352-2449.

Rinzin, Y. C. (2019a, 9/24/2019). Learning sign language is important to listen to deaf people. *Keunsel*. Retrieved from https://theworldnews.net/bt-news/learning-sign-language-is-important-to-listen-to-deaf-people

Rinzin, Y. C. (2019b, 9/27/2019). Research team developing Bhutanese sign language. *Kuensel*. Retrieved from https://annx.asianews.network/content/research-team-developing-bhutanese-sign-language-105080

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature, 323*(6088), 533-536.

Saleem, T., & Chishti, M. (2020). Assessing the Efficacy of Logistic Regression, Multilayer Perceptron, and Convolutional Neural Network for Handwritten Digit Recognition. *International Journal of Computing and Digital Systems, 9*(2), 299-308.

Sathya, R., & Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence, 2*(2), 34-38.

Schmitt, D., & McCoy, N. (2011). Object classification and localization using SURF descriptors. *CS, 229*, 1-5.

Sezgin, M., & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging, 13*(1), 146-166.

Sharma, S. (2017). Activation functions in neural networks. *Towards Data Science, 6*.

Shih, F. Y., & Cheng, S. (2005). Automatic seeded region growing for color image segmentation. *Image and vision computing, 23*(10), 877-886.

Shin, S., & Sung, W. (2016). *Dynamic hand gesture recognition for wearable devices with low complexity recurrent neural networks.* Paper presented at the 2016 IEEE International Symposium on Circuits and Systems (ISCAS).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sinith, M., Kamal, S. G., Nisha, B., Nayana, S., Surendran, K., & Jith, P. (2012). *Sign gesture recongnition using support vector machine.* Paper presented at the 2012 International Conference on Advances in Computing and Communications.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing, 14*(3), 199-222.

Sprenger, M. E., & Gwin, P. J. (2018). Radar-based gesture recognition: Google Patents.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*(1), 1929-1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). *Going deeper with convolutions.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Tripathy, B., & Anuradha, J. (2017). *Internet of Things (IoT): Technologies, Applications, Challenges and Solutions*: CRC Press.

Tuba, E., Tuba, M., & Simian, D. (2016). Handwritten digit recognition by support vector machine optimized by bat algorithm.

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems, 29*, 550-558.

Verma, V. K., Srivastava, S., & Kumar, N. (2015). *A comprehensive review on automation of Indian sign language.* Paper presented at the 2015 International Conference on Advances in Computer Engineering and Applications.

Walcott, S. (2009). Geographical field notes urbanization in Bhutan. *Geographical Review, 99*(1), 81-93.

Wangchuk, K., Riyamongkol, P., & Waranusast, R. (2020). Real-time Bhutanese Sign Language digits recognition system using Convolutional Neural Network. *ICT Express*.

Wangmo, T., & Choden, K. (2010). 25 The Education System in Bhutan from 747 AD to the First Decade of the Twenty-First Century. *Handbook of Asian education: A cultural perspective*, 442.

Xie, R., & Cao, J. (2016). Accelerometer-based hand gesture recognition by neural network and similarity matching. *IEEE Sensors Journal, 16*(11), 4537-4545.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). *Aggregated residual transformations for deep neural networks.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Xu, D. (2006). *A neural network approach for hand gesture recognition in virtual reality driving training system of SPG.* Paper presented at the 18th International Conference on Pattern Recognition (ICPR'06).

Yan, S., Xia, Y., Smith, J. S., Lu, W., & Zhang, B. (2017). Multiscale convolutional neural networks for hand detection. *Applied Computational Intelligence and Soft Computing, 2017*.

Yang, H.-D. (2015). Sign language recognition with the kinect sensor based on conditional random fields. *Sensors, 15*(1), 135-147.

Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). *Mixed pooling for convolutional neural networks.* Paper presented at the International conference on rough sets and knowledge technology.

Yun, L., Lifeng, Z., & Shujun, Z. (2012). A hand gesture recognition method based on multi-feature fusion and template matching. *Procedia Engineering, 29*, 1678-1684.

Zeiler, M. D., & Fergus, R. (2014). *Visualizing and understanding convolutional networks.* Paper presented at the European conference on computer vision.

Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology (TIST), 8*(3), 1-19.

Zhu, C., & Sheng, W. (2011). Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 41(3), 569-573.