

HAND GESTURE SEGMENTATION AND SIGN LANGUAGE RECOGNITION USING DEEP LEARNING

BY

M FAISAL NURNOBY

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

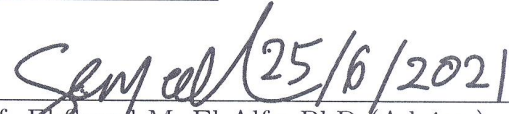
JUNE 2021

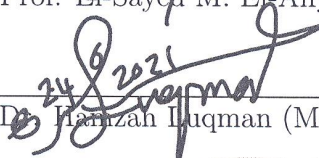
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

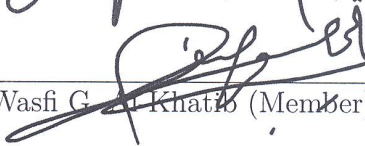
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **M FAISAL NURNOBY** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee


Prof. El-Sayed M. El-Alfy, PhD (Adviser)


Dr. Hamzah Luqman (Member)


Dr. Wasfi G. Khatib (Member)

For 

Dr. Hamoud Aljamaan
Department Chairman


Prof. Suliman Saleh Al-Homidan
Dean of Graduate Studies

Date



©M Faisal Nurnoby
2021

This thesis is dedicated to my elder brother, Mohammed Delwar Hossain, whose affection, encouragement and contribution make me strong despite many obstacles in my life. Also to my parents, wife, siblings and other family members for their endless love and care.

ACKNOWLEDGMENTS

All praise and prayers be to Almighty Allah, the most Kind and the most Merciful for giving me the opportunity to come thus far and complete this work.

I would like to express my deepest appreciation to my esteemed thesis advisor, Prof. El-Sayed M. El-Alfy, PhD, who has the substance and attitude of a genius. I owe a deep sense of gratitude for his meticulous scrutiny, patience, encouragement and excellence throughout the course of this thesis. Without his continuous support and guidance this work would not have been possible.

My sincere thanks goes to my other committee members, Dr. Hamzah Luqman and Dr. Wasfi G. Al-Khatib for their cooperation, advice and valuable support towards the improvement and completion of this work.

I would also like to show my gratitude to the department chair, Dr. Hamoud Aljamaan, for his sincerity and dynamism. I acknowledge the opportunity and support I receive from King Fahd University of Petroleum and Minerals (KFUPM).

Finally, my heartfelt thanks to my beloved mother, wife and other family members for the prayers, encouragement and support.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	v
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xiv
ABSTRACT (ENGLISH)	xvii
ABSTRACT (ARABIC)	xix
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Research Motivations	3
1.3 Thesis Objectives	4
1.4 Thesis Contributions	5
1.5 Thesis Organization	7
CHAPTER 2 BACKGROUND STUDY	8
2.1 Taxonomy of Gestures	8
2.2 Types of Recognition Approaches	10
2.3 Main Steps in Gesture Recognition	11
2.3.1 Visual Data Acquisition	12
2.3.2 Data Pre-processing	12
2.3.3 Image Segmentation	13

2.3.4	Feature Extraction	15
2.4	Hand Gesture Classification	18
2.4.1	Template Matching	18
2.4.2	Support Vector Machine (SVM)	19
2.4.3	Hidden Markov Model (HMM)	20
2.4.4	Deep Learning	21
2.5	Convolutional Neural Network (CNN)	21
2.5.1	Perceptron	23
2.5.2	Convolution Layer	24
2.5.3	Pooling Layer	27
2.5.4	Fully-Connected Layer	28
2.5.5	Training Convolutional Neural Network	28
2.5.6	Error and Loss Calculation	29
2.5.7	Optimization Algorithms	30
2.5.8	Overfitting	32
2.6	Popular CNN Architectures	33
2.7	Transfer Learning	38
2.8	Existing Segmentation Methods	40
CHAPTER 3 RELATED WORKS		44
3.1	Semantic Segmentation	45
3.2	Hand Segmentation	45
3.3	Static Sign Gesture Segmentation and Recognition	50
3.4	Static Sign Gesture Recognition	56
3.5	Dynamic Sign Gesture Recognition	57
CHAPTER 4 PROPOSED METHODOLOGY		61
4.1	Proposed System Framework	61
4.2	Semantic Gesture Segmentation Model	63
4.2.1	Semantic Segmentation	63
4.2.2	Encoder-Decoder Architecture	64

4.2.3	Skip Connection/Short Circuit	66
4.2.4	Transfer Learning-based Segmentation	66
4.2.5	Custom-based Semantic Segmentation	68
4.3	Manual Sign Recognition Models	69
4.3.1	TRNetV16	70
4.3.2	WB3CN	70
4.3.3	B4CN	72
4.3.4	MC-SLR	72
4.4	Integrated Model	74
4.5	Creation of Ground-truth Dataset	75
4.6	Other Tools and Techniques	77
CHAPTER 5	EVALUATIONS AND DISCUSSIONS	81
5.1	Introduction	81
5.2	Datasets	82
5.2.1	ArASL-PMU Dataset	82
5.2.2	Massey-G12 Dataset	83
5.2.3	GRASSK-ASL dataset	84
5.2.4	NUS-HII Dataset	84
5.3	Experimental Setup and Performance Measures	85
5.4	Recognition Models on Imagesets	86
5.5	Combining Segmentation with Recognition	94
5.6	Cross-Language Recognition	102
CHAPTER 6	CONCLUSIONS AND FUTURE WORKS	106
6.1	Limitation and Future Work	108
APPENDIX A	SUPPLEMENTARY MATERIAL	109
REFERENCES		144
VITAE		161

LIST OF TABLES

2.1	Interpreted gestures	10
2.2	Hand gesture classification models: pros & cons	21
3.1	Summary of segmentation methods used in the literature	48
3.2	Summary of static sign gesture segmentation and recognition in the literature	55
3.3	Summary of static sign gesture recognition in the literature	57
3.4	Summary of dynamic sign gesture segmentation and recognition in the literature	59
5.1	Summary of datasets	82
5.2	The percentage performance of SLR for different CNN models on dif- ferent datasets (approx. two decimal places)	93
5.3	The percentage performance of segmentation model while tuning train- ing parameters (approx. two decimal places)	95
5.4	The percentage performance of sensitivity analysis of S3RNet for aug- mentation parameters (approx. two decimal places)	98
5.5	The percentage performance of segmentation and recognition for dif- ferent CNN models (approx. two decimal places)	102
5.6	The percentage performance for different CNN models on cross-sign datasets for two phases (S/D) (approx. two decimal places)	104

LIST OF FIGURES

2.1	Taxonomy of gestures	9
2.2	Main steps in traditional vision-based gesture recognition	12
2.3	Difference of Gaussian [1]	16
2.4	SIFT terminologies	17
2.5	SVM: Feature hyperplane [2]	19
2.6	Basic CNN building blocks	22
2.7	A single layer perceptron	23
2.8	Convolution operation with a 3×3 kernel [3]	24
2.9	Activation functions	26
2.10	Max-pooling and average-pooling example	27
2.11	Cost curve	30
2.12	AlexNet (2012) architecture	33
2.13	VGG16 (2014) architecture	34
2.14	VGG19 architecture	34
2.15	Inception block architecture	35
2.16	Inception stem and 1st block architecture	35
2.17	Inception 2nd & 3rd block architecture	36
2.18	Simplified inception module	37
2.19	ResNet-50 architecture	38
2.20	Pre-trained model tuning strategy	40
2.21	FCN architecture [4]	41
2.22	U-Net architecture [5]	42
2.23	FC-DenseNets architecture [6]	43

3.1	Literature review areas of preference	44
4.1	Proposed system framework	62
4.2	Labels in a semantic segmentation	64
4.3	Encoder-decoder structure of the proposed S3RNet model	65
4.4	Atrous convolution at different rates [7]	67
4.5	Semantic-segmentation using DeepLabV3+ [7]	67
4.6	Custom-based semantic segmentation component	69
4.7	TRNetV16 architecture	70
4.8	Two custom-based recognition model architectures	71
4.9	MC-SLR system framework	73
4.10	S3RNet architecture	74
4.11	Alternate model: DLSR	75
4.12	Sample images of three annotation methods	76
4.13	Labeling with <i>MakeSense</i> tool	77
5.1	Sample sign gestures of ArASL-PMU dataset	83
5.2	Sample sign gestures of Massey-G12 dataset	83
5.3	Sample sign gestures of GRASSK-ASL dataset	84
5.4	Sample sign gestures of NUS-HII dataset	85
5.5	Learning curves for different models on the Massey-G12 dataset	89
5.6	Learning curves for different models on the ArASL-PMU dataset	90
5.7	Learning curves for different models on the GRASSK-ASL dataset	91
5.8	Learning curves for different models on the NUS-HII dataset	92
5.9	A sample of NUS-HII images cropped with ground truth masks	92
5.10	Segmentation model(S3RNet) tuning trials' results	95
5.11	Sensitivity analysis for shear range $[0 - 45.0]^\circ$	99
5.12	Sensitivity analysis for width shift $[0.5]$	100
5.13	Sensitivity analysis for rotation range $[0 - 25.0]^\circ$	101
5.14	ArASL-PMU: 2P model results	104
5.15	Training accuracy for single phase models	105

5.16 GRASSK-ASL: 2P results	105
A.1 Sensitivity analysis for brightness range $[0.2, 1.8]$	110
A.2 Performance measures: rotation range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII	111
A.3 Confusion matrix: rotation range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII	111
A.4 Performance measures: rotation range $[0 - 25.0]^\circ$ with S3RNet on NUS-HII	112
A.5 Confusion matrix: rotation range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII	112
A.6 Performance measures: rotation range $[0 - 20.0]^\circ$ with S3RNet on NUS-HII	113
A.7 Confusion matrix: rotation range $[0 - 20.0]^\circ$ with S3RNet on NUS-HII	113
A.8 Performance measures: width shift $[0.5]$ with S3RNet on NUS-HII . .	114
A.9 Confusion matrix: width shift $[0.5]$ with S3RNet on NUS-HII	114
A.10 Performance measures: width shift $[0.3]$ with S3RNet on NUS-HII . .	115
A.11 Confusion matrix: width shift $[0.3]$ with S3RNet on NUS-HII	115
A.12 Performance measures: shear range $[0 - 20.0]^\circ$ with S3RNet on NUS-HII	116
A.13 Confusion matrix: shear range $[0 - 20.0]^\circ$ with S3RNet on NUS-HII .	116
A.14 Performance measures: sear range $[0 - 45.0]^\circ$ with S3RNet on NUS-HII	117
A.15 Confusion matrix: sear range $[0 - 45.0]^\circ$ with S3RNet on NUS-HII . .	117
A.16 Performance measures: shear range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII	118
A.17 Confusion matrix: sear range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII . .	118
A.18 Confusion matrix: UNet on NUS-HII	119
A.19 Performance: UNet on NUS-HII	119
A.20 Performance measures: FCN16 on NUS-HII	120
A.21 Confusion matrix: FCN16 on NUS-HII	120
A.22 Performance measures: FCN32 on NUS-HII	121
A.23 Confusion matrix: FCN32 on NUS-HII	121
A.24 Performance measures: MobileNet on NUS-HII	122
A.25 Confusion matrix: MobileNet on NUS-HII	122
A.26 Performance measures: DLSR on augmented NUS-HII	123

A.27 Confusion matrix: DLSR on augmented NUS-HII	123
A.28 Performance measures: S3RNet on augmented NUS-HII dataset	124
A.29 Confusion matrix: S3RNet on augmented NUS-HII	124
A.30 Performance measures: S3RNet on NUS-HII	125
A.31 Confusion matrix: S3RNet on NUS-HII	125
A.32 Performance measures: AlexNet on ArASL-PMU	126
A.33 Performance measures: AlexNet on GRASSK-ASL	127
A.34 Performance measures: LeNet on ArASL-PMU	128
A.35 Confusion matrix: LeNet on GRASSK-ASL	129
A.36 Performance measures: TRNetV16 on Massey-G12	130
A.37 Performance measures: B4CN on Massey-G12	131
A.38 Performance measures: WB3CN on Massey-G12	132
A.39 Performance measures: WB3CN on ArASL-PMU	133
A.40 Performance measures: B4CN on ArASL-PMU	134
A.41 Performance measures: TRNetV16 on ArASL-PMU	135
A.42 Performance measures: WB3CN on GRASSK-ASL	136
A.43 Performance measures: B4CN on GRASSK-ASL	137
A.44 Performance measures: TRNetV16 on GRASSK-ASL	138
A.45 Performance measures: B4CN on NUS-HII	139
A.46 Performance measures: B4CN on NUS-HII	139
A.47 Performance measures: TRNetV16 on NUS-HII	140
A.48 Confusion matrix: TRNetV16 on NUS-HII	140
A.49 Confusion matrix: B4CN on NUS-HII	141
A.50 Performance measures: TRNetV16 on NUS-HII (masked)	141
A.51 Performance measures: B4CN on NUS-HII (masked)	142
A.52 Performance measures: WB3CN on NUS-HII (masked)	142
A.53 Confusion matrix: B4CN on NUS-HII	143

LIST OF ABBREVIATIONS

AdaM	Adaptive Momentum
AI	Artificial Intelligence
ArASL-PMU	Arabic Sign Language Alphabet Dataset by Prince Mohammad Bin Fahd University
ArSL	Arabic Sign Language
ASL	American Sign Language
ASPP	Atrous Spatial Pyramid Pooling
B4CN	Batch Normalization 4 Convolution Network
BOFs	Bag of Features
BSL	Bangladeshi Sign Language
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
CSL	Chinese Sign Language
CV	Computer Vision
DB	Dense Connection Block
DeepLabV3+	DeepLab Version 3-Plus
DLSR	Deep Lab Segmentation and Recognition

DoG	Difference of Gaussian
DSL	Dutch Sign Language
DTW	Dynamic Time Warping
EMG	Electromyography
eMOS	Estimated Mean Opinion Score
ESL	English Sign Language
FCN	Fully Convolutional Networks
GRASSK-ASL	Grassknotted Dataset
HCI	Human Computer Interaction
HMM	Hidden Markov Model
HOG	Histogram of Gradient
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IMU	Inertial Measurement Unit
ISL	Indian Sign Language
JPN	Japanese Sign Language
JSON	JavaScript Object Notation
kNN	K-Nearest Neighbour
KSL	Korean Sign Language
LMC	Leap Motion Controller
LSTM	Long Short Term Memory
Massey-G12	Gesture 2012 Dataset by Massey University

MC-SLR	Multi-culture Sign Language Recognition
ML	Machine Learning
NUS-HII	Gesture Hand II Dataset by National University of Singapore
NZSL	New Zealand Sign Language
PDF	Probability Distribution Function
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
ROI	Region of Interest
S3RNet	Sign Gesture Semantic Segmentation and Recognition Network
SGD	Stochastic Gradient Descent
SIFT	Scale Invariant Feature Transform
SLRS	Sign Language Recognition System
SPP	Spatial Pyramid Pooling
SVM	Support Vector Machine
TRNetV16	Transfer Learning Network VGG16
TSL	Tamil Sign Language
VGG	Visual Geometry Group
VOC	Visual Object Classes
WB3CN	Without Batch Normalization 3 Convolution Network
YOLO	You Only Look Once

THESIS ABSTRACT

NAME: M Faisal Nurnoby

TITLE OF STUDY: Hand Gesture Segmentation and Sign Language Recognition Using Deep Learning

MAJOR FIELD: Computer Science

DATE OF DEGREE: JUNE, 2021

Communication through hand gestures is an active research in Human Computer Interaction (HCI). Deaf and speech-impaired community may benefit from the advancement of human gesture recognition, as it is an intermediary communication technique. Segmentation for gestures with complex background is a crucial task for better recognition. Semantic segmentation is gaining momentum in a range of fields. However, sign gestures have not witnessed a broad application of semantic segmentation. In this thesis, we propose a CNN-based integrated mode for the segmentation and recognition of hand sign gestures with conditions such as complex background, blur, and variation of illuminations. Our model, dubbed as S3RNet, is composed of two components: a custom CNN-based segmenter with encoder-decoder architecture and a custom CNN-based recognition module. The segmentation component is designed with depth-wise separa-

ble convolutions networks with fine-tuned layers, and appropriate network parameters with an aim to have optimum performance even for a small dataset. In a second variation of our model, dubbed as *DLSR*, we replaced the segmentation component with a transfer learning-based *ASPP* module. The recognition module for both models remain the same. Besides, in designing a more suitable gesture recognition module, we proposed three architectures: two custom-based and one transfer learning-based. The proposed models were tested with three sign language datasets and one gesture dataset. A ground-truth gesture dataset has been created for evaluating the models. The results demonstrate that *S3RNet* achieved an *mIoU* score of 95.17% for segmentation and a recognition accuracy of 99.72% on the test dataset. Several sensitivity analysis experiments were also performed to observe the robustness of our models. We compared our results with other established models. Besides, as a separate task, this research also explored and presented a new deep classification approach for cross sign language recognition.

ملخص الرسالة

الاسم: م. فيصل نور نبي

عنوان الدراسة: التقسيم الدلالي لإيماءات اليد والتعرف على لغة الإشارة باستخدام التعلم العميق

التخصص: علوم الحاسب الآلي

تاريخ الدرجة العلمية: يونيو 2021

يعد التواصل من خلال إيماءات اليد أحد الاتجاهات البحثية النشطة في مجال التفاعل البشري مع الحاسب، وقد كان له أثر بالغ في تطوير تطبيقات لتسهيل التواصل مع الصم من خلال التعرف على لغة الإشارة. ورغم التقدم الملحوظ في هذا المجال خصوصاً مع التطور التقني الحديث، ما زالت هناك تحديات لمعالجة الصور ذات الخلفيات المعقدة بشكل أفضل. وتهدف هذه الدراسة إلى مناقشة الجوانب المختلفة للمشكلة وتطوير ومقارنة مجموعة من الحلول المبنية على التعلم العميق لدمج التقسيم الدلالي للإيماءات مع أساليب تصنيفها، مما يساهم في رفع كفاءة التعرف عليها عند وجود خلفية معقدة أو تشويش في الصورة. وقد بينت نتائج التجارب المختلفة فعالية النموذج الذي أطلقنا عليه اسم S3RNet في التقسيم الدلالي بلغت 95.17% مع دقة تصنيف للإيماءات بلغت 99.72%. كذلك قمنا بإجراء العديد من تجارب تحليل الحساسية لمراقبة متانة النموذج المقترح ومقارنته مع نماذج أخرى.

CHAPTER 1

INTRODUCTION

1.1 Introduction

As computers get more powerful and prevalent, Human Computer Interaction (HCI) is also gaining a significant role to play in enhancing user's experience. Therefore, all the communication barriers between human and computer should be as minimal as possible. The main goal, in this regard, is to make the interaction as natural as possible for communication between humans. For this reason, recognition of human gestures has been an important research topic. It is regarded as an intuitive method for a more natural and effective communication with machines. Human sign language is a defined form of hand gestures and other body postures that include visual movements and signs. It is a conventional way to communicate with others in human civilization, apart from voice and speech. It also acts as an intermediary communication technique for regular interaction among the deaf and speech-impaired people. Most of the language signs deal with the top half part of the body, starting from waist to head. One can group human hand gestures into several categories such as

conversational, controlling, communicative, and manipulative [8], [9]. In that context, sign language is a highly structured communicative type of gestures. The recognition of sign language and gestures involves a complete process of detecting and recognizing the signs, and then translating them into relevant words that are semantically significant. Normal hearing people have difficulties in understanding the sign language. Researchers have been showing great interests in solving this communication problem for a long time. The recognition process is not free from challenges. The success of the recognition accuracy depends largely on the effectiveness of the image segmentation method, especially for a vision based gesture recognition system. From the literature, we have observed that the most frequent recognition researches in sign languages are on American Sign Language (ASL). There are many other researches on sign languages including Chinese Sign Language (CSL), Indian Sign Language (ISL), Arabic Sign Language (ArSL), Tamil Sign Language (TSL), Korean Sign Language (KSL), Dutch Sign Language (DSL), Persian Sign Language (PSL), Malaysian Sign Language (MSL), English Sign Language (ESL), New Zealand Sign Language (NZSL), Japanese Sign Language (JPN), and Bangladeshi Sign Language (BSL). This thesis aims at exploring new methodologies for segmentation and recognition of hand gestures in the context of sign languages. In the following two sections, we define our research motivations and thesis objectives.

1.2 Research Motivations

Recognition of human gestures involves a complex set of activities such as modeling of motion, analysis of motion, pattern matching/recognition, and several machine learning techniques [10]. Moreover, it involves functions with both manual and non-manual arguments. The composition of the surrounding environment, such as hand movement, speed and background light, influences the gesture prediction performance. In a 2D space, viewpoints play a crucial role and any change affects the appearance of a gesture. Sometimes in sign language recognition, the signer wears colored gloves or wrist bands in order to help the image segmentation process and to decrease the complexity of sign image segmentation steps. In a general situation, wearing extra equipment may not always be a feasible scenario; instead a good segmentation method can take care of the region of interest in a more efficient way. In gesture recognition, there exists a number of problems, such as temporal variance, epenthesis of movement, spatial complexity, and repeatability [11]. In the literature, we found many novel approaches for tackling the segmentation issue from diverse perspectives (e.g. [12], [13]). However, we found no single state-of-the-art model that suitably fits for sign gesture recognition under special conditions such as complex background, poor illumination and blur. Many researchers in Sign Language Recognition System (SLRS) also incorporated CNNs in their models, but they were mainly for gesture classification purpose. Even for those, who applied semantic segmentation, it was for ego-hands only. A robust CNN-based integrated method for sign arm segmentation and recognition begs more traction. Most of the solutions we have come across involve

skin-color partitioning in the YCbCr color spectrum. This method is not suitable for sign images having complex backgrounds. We have also found that skin-color segmentation method seldom brings in perfect hand regions, whereas in other domains a good number of models show almost near-ground-truth semantic segmentation. As the quality of segmentation has a profound impact on the efficiency of gesture recognition systems, we aspire to contribute a good effort on designing a more effective semantic segmentation method that accurately localizes all regions of gesture hands from complex backgrounds in colored images and generates masks to crop the images for gesture recognition.

1.3 Thesis Objectives

The main objectives of this thesis are:

- Surveying and analyzing the state-of-the-art techniques and datasets of both gesture segmentation and recognition.
- Collecting and building a sign language dataset under the specific requirements of complex background and occlusion issues.
- Exploring and developing a segmentation framework for hand sign gestures.
- Evaluating and comparing the proposed techniques.

1.4 Thesis Contributions

The main the contribution of the thesis can be summarized as follows:

1. *Proposal of a more effective sign language recognition model:* This thesis has explored several best practiced CNN based recognition models. For sign gesture recognition, a range of models have been applied in the literature, from as simple as LeNet to as deep as AlexNet and VGG16. There are some other deeper models available too, but unlike hand gesture, they were designed for critical objects recognition. However, this research proposes three custom models, suitable for the recognition of hand sign gestures. The outcome and analysis of each model is provided in the thesis for deeper understanding.
2. *Proposal and evaluation of an integrated approach for hand gesture segmentation and recognition:* In this work, we have contributed two models. The first model is a completely new model dubbed as S3RNet. This model uses a custom network for semantic segmentation and recognition of hand gestures from complex background. The second model dubbed as DLSR uses transfer learning of a Atrous Spatial Pyramid Pooling (ASPP) for segmentation integrated with a customized model for recognition. We fine-tuned encoder-decoder architecture in terms of appropriate number of layers, stride and other network parameters in order to improve our segmentation. We are inspired by the depth-wise and point-wise separable convolution in our encoder-decoder architecture, that contribute to our overall performance to attain a new height on our custom-developed segmentation dataset. The proposed model will contribute to the

advancement of HCI as well as to digital assistant to deaf and speech-impaired people.

3. *Creation of ground-truth for hand gesture semantic segmentation:* A new annotation dataset has been developed in COCO format from NUS-HII images of about 2,000 sign gestures. The hand-crafted ground truth annotated images have been used to train S3RNet model. This dataset will be made available online for public use. The dataset has been augmented to increase the data size for both sign images and their corresponding segmentation masks.
4. *Exploration of a new direction research for a cross-sign language recognition system:* This research offers a new deep classification approach where instead of working on a single nationality sign dataset, it has combined and normalized two different nationality datasets into one; then we have developed a two-phased identification and recognition of each sign to its corresponding ethnic language. The proposed approach has been implemented with different variants of CNN architectures. This model can be regarded as the standard procedure for any sets of multi-sign language datasets. The results showed compelling recognition ability without major intra-language recognition fault for multi-ethnic sign gestures.
5. *Comprehensive literature review:* This thesis provides a comprehensive literature review on sign language recognition, contemporary segmentation models for sign recognition, semantic segmentation models, and standard CNN-based architectures. First, we have explored different machine learning-based classi-

fiers in the literature and then analyzed the tools and techniques they used for both segmentation and recognition. We have also made a comparative analysis of their performances and processes.

1.5 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 and 3 provide detailed background and review related works. We have provided background on gesture taxonomy, types of recognition approaches, gesture classification techniques, Convolutional Neural Networks (CNN), popular CNN architectures and transfer learning. The literature review section covers recognition of both dynamic and static signs, and semantic segmentation. Chapter 4 presents our proposed methodology for sign recognition, sign hand segmentation, and combined approach. Chapter 5 provides the details of experiments, results, and discussions. Finally, Chapter 6 concludes the thesis and gives a brief description of future research directions.

CHAPTER 2

BACKGROUND STUDY

2.1 Taxonomy of Gestures

In the study of psycho-linguistics, a great deal of efforts have been made for describing and grouping gestures. A detailed review on this topic is beyond the scope of this thesis. Besides, in reviewing the literature, we encounter one fact: there is no fixed categorization of gestures so far. For instance, there are at least four cogent taxonomies in categorizing them. There are even examples of using the same gesture term to mean two separate things [14]. Therefore, we tried to integrate those taxonomies into one big picture so that one may see the complete view in different perspectives. Figure 2.1 shows a taxonomy of gestures in a tree form. The explanation of gestures can be partitioned along three directions; the first one explains the function of a gesture (manipulative or communicative, etc.), the second one describes the nature of a gesture (static or dynamic), and the final one tells the applications of gestures [15].

Although the gesture taxonomy is shown as a tree, its categories consist only of the end leaves. The inner branches simply label the semantic grouping.

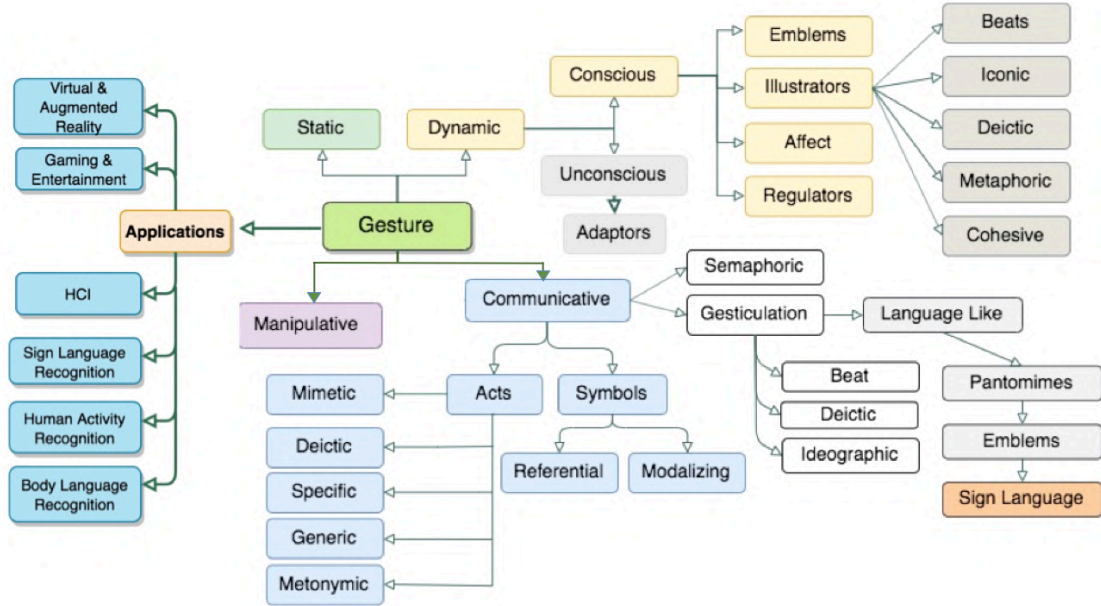


Figure 2.1: Taxonomy of gestures

Another way to classify gestures is based on interpretation, and based on observable features. If we consider temporal relationships, then observable features based gestures can be categorized into either static or dynamic (Figure 2.1). In static category, hand position usually does not change much during gesturing; these types of gestures primarily depend on the shape and angles (flexure) formed with fingers. In contrast, hand position shifts simultaneously with time, in dynamic gestures. There are three distinct movement phases in dynamic category, such as preparation, stroke, and reversal. The core message in a dynamic sign gesture is incorporated in the stroke temporal sequence. Furthermore, dynamic sign gestures depend on the orientations and trajectories of hand(s), in conjunction to the flex angles (of fingers) and shape. However, dynamic gestures can be interpreted as conscious and unconscious. Adaptors are unconscious type of gestures whereas conscious gestures include emblem, illustrators, affect and regulators. These dynamic interpretation

Table 2.1: Interpreted gestures

Gesture	Meaning	Example
Illustrators	spoken words illustration thru gestures	Showing directions by finger pointing
Emblems	Can be a replacement for spoken words	Displaying thumbs-up meaning "all right"
Regulators	Helps the communication/interaction between speaker and audience	Raise hand to administer turn-taking)
Affect	Facial expressions, showing the level of an emotion	staring at something and cower to display the emotion of fear
Adaptors	used for personal convenience, usually a habit	adjusting/taking off glasses in a critical situation

based gestures have certain translated meanings. For instance, Table 2.1 gives a list of interpreted-gestures along with examples for better understanding of these special classes of gestures. One may further make sub-groups of illustrators type gesture into beats, iconic, deictic, metaphoric, and cohesive.

2.2 Types of Recognition Approaches

Detection of gestures is performed by applying a vision-based approach or a sensor/hardware-based system. In a vision-oriented system, a video camera is needed for capturing images of the gestures. In the literature, many types of cameras were used for the acquisition of sign gesture images. We can categorize them as single camera (webcam, video-camera, smartphone), stereo-camera (multi-monocular cameras for depth information), etc. Also in some active techniques, the projection of structured light was used; such tools may incorporate Kinect and Leap Motion Controller (LMC). Besides, many articles in the literature used some invasive equipment, that

may include body markers, gloves with color, LED lights, and wrist bands. On the other hand, many researches chose sensor-based approach, where the motion, spatial position, and speed of the hand was captured by sensors or hardware instruments [16]. One may also use accelerometer and gyroscope for measuring position, acceleration, degree of freedom, etc; these instruments are commonly known as Inertial Measurement Unit (IMU). In order to designate intuitive gestures by hand, the movements are captured by the use of Euler angles resulted from IMU [17]. Electromyography (EMG) is evolving as one of the most crucial techniques for recording muscle activity which may contribute greatly to human-machine interaction for gesture recognition [18]. There are many other devices that were used in research; e.g. Wi-Fi, Radar, flex sensors, ultrasonic, electromagnetics, and haptic tools.

2.3 Main Steps in Gesture Recognition

The method of gesture recognition can be classified into several steps, such as data acquisition, data pre-processing, segmentation, feature extraction, and classification. In case of static gesture, the input is a single image frame whereas for dynamic gesture the input is mostly videos, which are consecutive frames of photos. The main difference between vision-based method and the sensor oriented approach is the data acquisition phase. This thesis is focused on the vision-based sign language and gesture recognition approach. Therefore, a brief detail of each recognition stage is given in the following sub-sections.

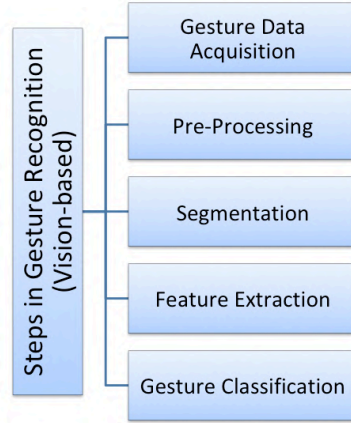


Figure 2.2: Main steps in traditional vision-based gesture recognition

2.3.1 Visual Data Acquisition

An array of image frames is acquired in a vision-oriented sign language recognition system. Those images are captured using devices like webcam, video camera, stereo-camera, and some other advanced tools. Some modern 3D cameras, like Leap Motion Controller (LMC) and Kinect, can gather depth information of the objects inside the frame as well.

2.3.2 Data Pre-processing

Image data pre-processing is a general term for processing images or videos at the lowest abstraction level. The main purpose of this step is to improve the gesture data by suppressing the untoward distortions or noises, and thereby improving some image features which are crucial for further processing. Sometimes some forms of geometric operations of images, like scaling, rotation, translation etc., are also classified as important pre-processing methods [19]. In order to reduce noises in both acquired images or videos, few common filters like median or Gaussian (Eq. 2.1) are vastly

used, where x and y are the distances from origin along horizontal and vertical axes respectively. σ represents the standard deviation.

$$g(x, y) = \frac{1}{\sqrt{2\pi a^2}} e^{-\frac{x^2+y^2}{a^2}} \quad (2.1)$$

In research [20], [21], the authors applied the median filter in the pre-processing stage. However, this is not always the case that noises are removed; sometime it may require extra noise to be added, e.g. [22]. Another pre-processing method, called morphological operation, is generally used for reducing unnecessary details. In [23], after applying median and Gaussian filters, the paper also applied morphological pre-processing. Next, downsizing the captured image into lower resolution has been a common phenomenon in image pre-processing. It substantially reduces the computational complexity in many cases ([24], [25], [23], [26]).

2.3.3 Image Segmentation

Segmentation is the process of splitting an image into multiple fragments and returning the Region of Interest (ROI) out of it. In vision-based sign gesture, a quick and accurate segmentation of hand(s) is processed from a sequence of images. Besides, one may categorize segmentation methods as contextual or non-contextual. Under each category if only one threshold value is applied it is named as global thresholding. Likewise, for local thresholding, the image is partitioned into many sub-regions, and for each of the sub-regions a different threshold values are assigned. Two more prominent segmentation methods, other than thresholding, are commonly used in the vision-based sign gesture recognition models in the literature and are explained below.

However, a tentative comparison among different segmentation techniques is provided in the literature review chapter.

A) *Skin-Color Segmentation:*

Generally, a skin detector is used in this segmentation technique for finding ROI out of a sign language gesture in an image. This dynamic detector depends on the skin-color tone of face and hands [27]. First, the face is detected from the first frame. Then the bins of Probability Distribution Function (PDF) histogram are evaluated and cut at a threshold value (e.g. 0.005). A dynamic threshold value is calculated from pixel values along the two axes of the boundary rectangle. Afterwards, this value is used to the image containing face region, in order to detect skin pixels; the threshold value is updated by gradually increasing the pixel regions around the two axes, until most of the face pixel regions are granted as a skin. This threshold value is imposed on to the entire image for detecting other parts. In addition, YCbCr color space was also used as it reduces the impact of illumination on the image segmentation [28]. The result of this method is a binary image that contains the hands and the face regions with white pixels and the rest in black.

B) *Hand Tracking:*

The aim of tracking in a dynamic gesture is to demarcate hand(s) from the entire background image. Due to fast movement and change in appearance, tracking hand(s) becomes a very difficult task. A method named CAMShift is applied in some researches in order to track the spatial position of the core gesture region.

CAMShift works by identifying the exact position of hand gestures just by calibrating the size of the inquiry window. Next, its Adaboost outputs a strong classifier by computing the corresponding sign of a group of weak classifiers that it consists of. Using Histogram of Gradient (HOG) along with Adaboost, Wang et. al. [29] distinguished movements of hand with formidable result. Another gesture tracking method is known as particle filtering. In order to detect hand with good accuracy, a mix of mean-shift and particle filtering algorithm may also be used to detect hands in real time, and thus it makes gesture recognition more time efficient [30].

2.3.4 Feature Extraction

Feature extraction is the conversion of interest parts of an input image into a group of feature vectors [31]. In the context of gesture recognition, the feature extraction produces relevant feature components from the sign gestures input, and consequently serves for the identification of a particular gesture class. A deep learning based classifier has feature extraction phase integrated within the model; there is no need for explicit effort in this regard. In many other research (e.g. [32]), where no deep learning method applied, hand engineered feature extraction techniques were used. However, we describe Scale Invariant Feature Transform (SIFT) [1] in this section for its high significance in computer vision as a good feature extractor. SIFT is a prominent feature extraction method, invented by D. Lowe in 1999. It is popular over other methods, because of its invariant nature towards rotation, illumination, a defined range of affine transformation, and change in 3D view point. Generally the overall process of SIFT

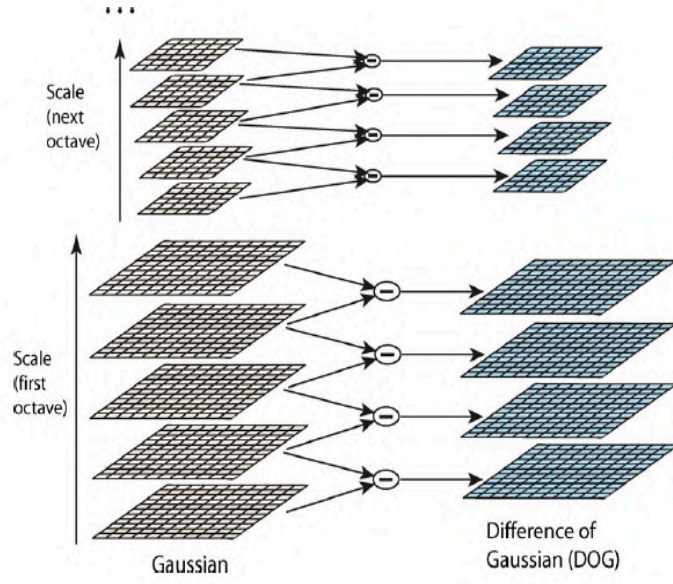
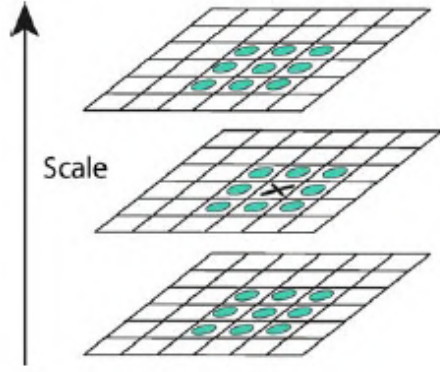


Figure 2.3: Difference of Gaussian [1]

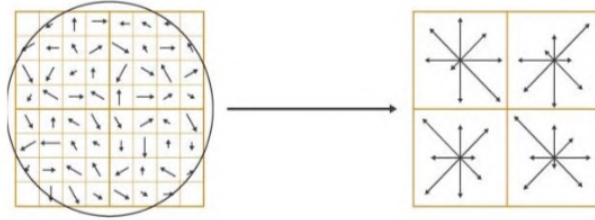
may be divided into four steps, such as keypoint identification, localization of those keypoints, orientation assignment, and finally, the generation of feature descriptors. Under the SIFT method, an image is described by its interest points, and at those points, a way of multi-scale detection is required. At each pyramid scale, the image needs to be re-scaled and then smoothed with Gaussian filters, as shown in Figure 2.3. We may formulate an equation, where the scale space can be described as $L(x, y, \sigma)$ (Eq. 2.3) and this is generated by performing convolution with a multi-scale Gaussian, $G(x, y, \sigma)$, along with the input image, $I(x, y)$. This scenario can be explained in the following equation:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.2)$$

Here, the extracted key-points are the local extrema, obtained from calculating the Difference of Gaussian (DoG), $D(x, y, \sigma)$, and this can be formulated by measuring the variation in scales of two consecutive pyramids, separated by a multiplicative constant



(a) Local extrema [1]



(b) Key point orientation [33]

Figure 2.4: SIFT terminologies

(k) (Eq. 2.3, 2.4) [33]. Then, the overall equation becomes:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.3)$$

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) \quad (2.4)$$

Local extrema calculation is pretty straight forward; each point $D(x, y, \sigma)$ is compared with 26 neighbor points—same scale 8 points, up and down 9 points each (Fig. 2.4a). As mentioned earlier, the third step in SIFT is orientation assignment. Based on the local properties of the image, an orientation is defined for each key point (Fig. 2.4b). Afterwards, the final step is the creation of local descriptors. These descriptors can be generated through applying nearest neighbor and distance calculation.

2.4 Hand Gesture Classification

Gesture classification can be defined as a process of grouping points/tracks of a model specification space into a particular subgroup of the space. To explain this definition, image-based static hand gesture is related to the model specification space point, and the video-based dynamic gesture is related to a track of model specification space. A list of classification methods of hand gesture includes Machine Learning (ML) based approaches, template matching, Hidden Markov Model (HMM), geometric feature based techniques, and Dynamic Time Warping (DTW) [34]. Several classification methods are described briefly in the following subsections. At the end this section, in Table 2.2, a comparison is made among several classification models.

2.4.1 Template Matching

Template matching is one of the oldest and easiest methods for pattern recognition, mainly applied in static sign gesture recognition. The process is to compare the input hand gesture image with the template (edge, point, shape, etc.) and therefore, classify them according to matching score. The methods for evaluating similarity score are Euclidean distance, Contour edge matching, Hausdorff distance, matching of elastic map, etc. The benefit of using a template matching technique is that, it is easy and fast, and it is not influenced by other factors like illumination, posture, background, etc. However, the classification accuracy is not very promising under this simple method.

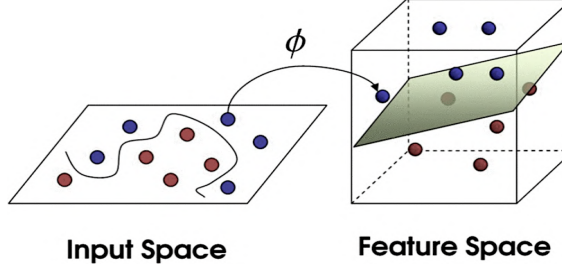


Figure 2.5: SVM: Feature hyperplane [2]

2.4.2 Support Vector Machine (SVM)

SVM is a supervised pattern recognition ML algorithm. The main goal of SVM is to find an optimal hyperplane that separates two classes. This popular classification algorithm measures the distance of the input samples from this separation boundary (a.k.a. decision boundary). This hyperplane boundary is determined at the time of training where the algorithm chooses a portion of input data, the support vectors, in order to construct the SVM model [2]. For deeply non-linear decision boundary, this model can map a higher dimensional feature space from the predictor, and make the possible data separation. Given a dataset, $\{x_i, y_i\}$, where the number of training samples is N , $i = 1, 2, 3, \dots, N$; x_i is a vector representing features, and y_i is the target label with a value either $y = +1$ (*class1*) or $y = -1$ (*class2*). Geometrically speaking, the support vector machine model searches for an ideal hyperplane with optimal margin to divide *class1* and *class2*. The scenario can be expressed as follows [35]:

$$(y_1, x_1), \dots, (y_n, x_n), \quad y_i \in \{-1, 1\} \quad (2.5)$$

$$\begin{aligned} w \cdot x_i + b &\geq 1 & \text{if } y_i &= 1 \\ w \cdot x_i + b &\leq -1 & \text{if } y_i &= -1 \end{aligned} \quad (2.6)$$

where W is a slope vector of the decision boundary/hyperplane, and b is a scalar. The distance between the two classes can be expressed as follows:

$$\rho(w, b) = \min_{\{x:y=1\}} \frac{x \cdot w}{|w|} - \max_{\{x:y=-1\}} \frac{x \cdot w}{|w|} \quad (2.7)$$

Therefore, for an optimal hyperplane the arguments (w_0, b_0) that maximizes the data separation margin is calculated from Eq. 2.6, and 2.7 as the Eq. 2.8:

$$\rho(w_0, b_0) = \frac{2}{|w_0|} = \frac{2}{\sqrt{w_0 \cdot w_0}} \quad (2.8)$$

In the literature, many gesture recognition systems use SVM as a classifier and achieve higher accuracy over other models, e.g. SVM is used in [32] for classifying ArSL dataset and achieved a higher accuracy over kNN model.

2.4.3 Hidden Markov Model (HMM)

HMM is a probabilistic method of analyzing sequential data. This model constructs the process of producing random sequences of unseen state from a hidden Markov series, and subsequently generates an inspection random series from individual state. It is a simple Bayesian dynamic network with directed graph, which was extensively applied in the areas of speech recognition. In particular, HMM is more applicable for continuous gesture detection where hand gesture is complex and context-sensitive [34]. One issue regarding continuous signals analysis is that, the training and recognition of HMM is computationally expensive. Discrete HMM was applied instead in the usual hand gesture recognition to circumvent the computational issue.

Table 2.2: Hand gesture classification models: pros & cons

Model	Pros	Cons
Template Matching	Fast and easy; not affected by illumination etc.	Performance degrades for large training data
HMM	Easy tuning of training and validation, model clarity	weak distinguishing performance between classes
SVM	Easy to implement, robust outcome	For large training data, can not hold on to performance
Deep Learning	Less effort on training data; model takes care of everything	huge volume of training samples required

2.4.4 Deep Learning

Deep Learning is a booming and emerging wing of ML field, that consists of models having multi-processing layers, and learns data representations at varied layers of abstraction. It resembles the synergy of human nervous system, and has the ability of being robust, highly fault-tolerant, adaptive, highly parallel, anti-interference and mobile learning [34]. Furthermore, unlike traditional ML algorithm, deep learning requires little tuning by hands, and enables the computational power of handling enormous amount of data. Two most popular deep learning models are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). There has been a growing trend of applying deep learning techniques for human gesture and sign language recognition systems. In what follows, we describe CNN and its components in more detail.

2.5 Convolutional Neural Network (CNN)

Artificial Intelligence (AI) has been bridging gaps between performance of machines and human capabilities. Researchers and industry experts alike have been working

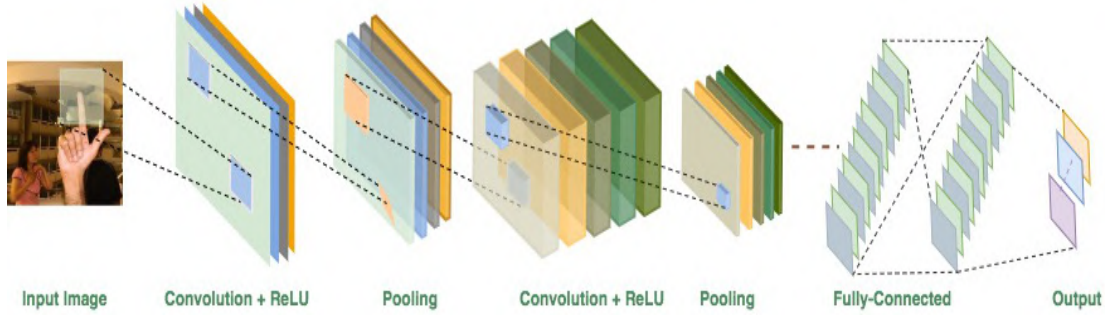


Figure 2.6: Basic CNN building blocks

on various aspects in this field to make significant contributions. Computer Vision (CV) is one of many such progressive areas. The core objective in this area is to make machines capable of viewing the world like humans. Therefore, a lot of efforts have been made in image analysis, recognition and classification. The progress in CV along with deep learning has been perfected over time. In particular, CNN has been the state-of-the-art in almost all core CV algorithms. Since our proposed model in this thesis is extensively based on CNN, we dedicate a full subsection on this popular network architecture. CNN can automatically learn spatial hierarchy of features from low-level to higher-level patterns in an adaptive manner through a technique called backpropagation. The network is constructed mathematically and has several core layers like convolution, pooling, activation and fully-connected layers through which it can process a grid pattern like images. The whole deep learning model is inspired by the structure of human visual cortex [36]. Convolution and pooling layers are responsible for building spatial feature maps whereas a fully-connected layer relates the feature maps into final output layer which is usually a classifier (Fig. 2.6).

2.5.1 Perceptron

A perceptron refers to a special type of supervised learning model where its architecture and behavior is quite similar to a brain neuron. It is often regarded as the most basic version of neural network. A single layer perceptron is easy to explain and it acts as a linear binary classifier. In Fig. 2.7, a feature vector $\mathbf{x} = [x_1, x_2, x_3 \dots x_m]$ is used as input and the single layer perceptron predicts the probability as an output of a particular event or class. Each input in the feature vector is associated with its corresponding weight ($\mathbf{W}$) which determines the contribution of a certain input feature requires in the summation operation (Eq. 2.9).

$$V = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_mx_m \quad (2.9)$$

$$V = [\mathbf{W} \ b][\mathbf{x} \ 1]^T$$

In simple terms, a set of input elements are set as more significant than the others by assigning them greater weights so that they have a greater influence in calculating an alternate value V . A bias $b = w_0$ is also added to this summation operation. The

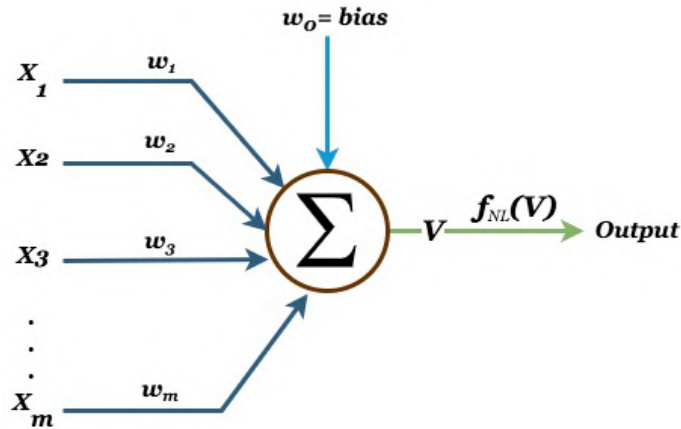


Figure 2.7: A single layer perceptron

output of a summation function is then passed through a nonlinear activation function

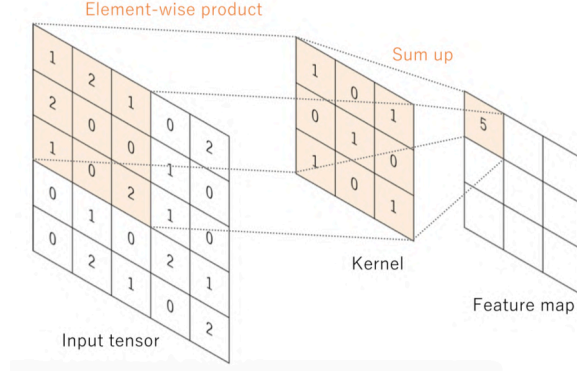


Figure 2.8: Convolution operation with a 3×3 kernel [3]

f_{NL} . Since the final output is the probability of a class, usually a *sigmoid* function is used to limit the output value between 0 and 1.

2.5.2 Convolution Layer

Convolution layers are the core parts in a CNN model. These layers are nothing but stacks of mathematical steps. In digital image processing, an image channel (e.g. RGB channels) is stored as a 2D array, also called a tensor. Different types of kernels are applied on an image to extract important spatial features to generate feature maps. A kernel/filter is a small grid of optimizable parameters which is applied at each image position to calculate a desired spatial operation (Fig. 2.8). Output maps of one layer are fed into the consecutive layer, and features are mapped progressively in a hierarchical manner. At each training epoch, the CNN model optimizes its parameters in order to make the losses (differences between outputs and true labels) as minimum as possible through the backpropagation optimization learning algorithm and gradient descent.

A) *Convolution:*

Convolution is a linear operation which is applied for extracting features of a particular layer. Usually, an element-wise operation between kernel and input tensor is worked out at each tensor point and added up to generate output tensors or activation maps (Fig. 2.6). This process is repeated with a list of predefined filters/kernels for a specific convolution layer and each feature map is stacked for using in the next layer. Each convolution layer learns specialized local patterns of the input tensors. These patterns have two important properties: rotation invariant and spatial hierarchy. After a convolution layer learns a particular pattern, the model can recognize it in another tensor, even if the feature is rotated (rotation invariant). This makes a CNN model require less training data to learn a pattern, and helps in generalization. Next, an early convolution layer would learn small segments of local patterns like edges, and a following layer would learn broader patterns made up of features from previous layers, and so on. This helps CNN efficiently learn complex and abstract concepts which are spatially hierarchical. Usually the filters are of size 3×3 , but sometimes filter size of 5×5 or 7×7 are also used.

B) *Zero Padding:*

The convolution process explained in the previous section does not mention one issue which is the center of each filter cannot overlap the border element (pixel) of the input tensor. As a result, the output maps will have smaller width and height compared to the incoming input tensor. Usually a technique called

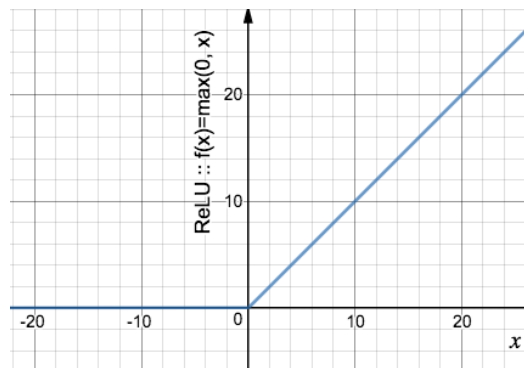
zero-padding is applied to keep the integrity of the output size. If no padding is applied, then each of the following activation maps would be smaller in size after each convolution.

C) *Stride*:

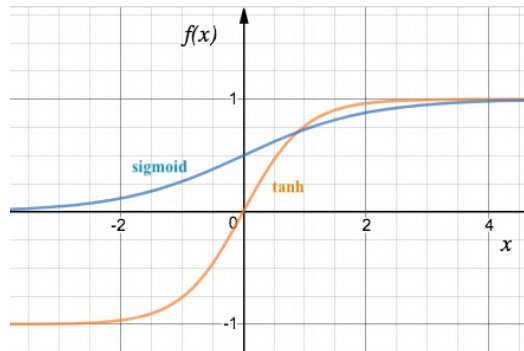
The separation between two consecutive filter positions is defined as a stride.

The most preferable choice for a stride is one. However a larger stride is also used to perform downsampling.

D) *Activation Function*:



(a) ReLU



(b) *Sigmoid* and tanh

Figure 2.9: Activation functions

The outcomes of a convolution layer are processed through an activation function, which adds non-linearity. The most commonly used activation function is

Rectified Linear Unit (ReLU) (Fig. 2.9a). Previously, *sigmoid* and *tanh* (hyperbolic tangent) were also common (Fig. 2.9b). ReLU is a piecewise linear function with the form, $f(x) = \max(0, x)$.

2.5.3 Pooling Layer

A pooling layer reduces the spatial dimensions of a feature map. This downsampling operation aims to incorporate invariance to distortion and small shifts, and also decreases the volume of learnable parameters. As a result, the training phase of the model becomes more efficient and faster. A pooling layer has a kernel size, stride and zero-padding as hyperparameters similar to convolution operations, however in a pooling layer, the learnable parameters are not present.

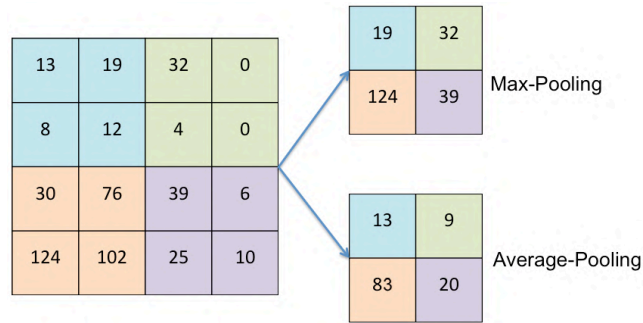


Figure 2.10: Max-pooling and average-pooling example

Max and average pooling are the two variants of pooling operation (Fig. 2.10). In max-pooling, the largest value is extracted from each patch of a feature map. Average pooling, on the other hand, finds an average value of all the values in a patch. Besides, max-pooling acts as a noise suppressor by discarding noisy activations, and also contributes to feature reduction. Average-pooling has a little less ability compared to max-pooling; it can merely contribute in dimensionality reduction to suppress noise.

2.5.4 Fully-Connected Layer

As part of the CNN, a fully-connected layer takes the outputs of the previous convolution/pooling layers and finally generates a classification decision. This process may be performed in a couple of stages. First, the output taken from previous layers is flattened into a vector and linked to dense layers (one or more fully-connected layers), where each individual input is connected to each output through a learnable weight. The last dense layer usually has the same output nodes as the total number of classes. Each dense/fully-connected layer is followed by an activation function (usually ReLU). The activation function at the last dense layer is usually unique from the others. A suitable activation function is chosen based on the nature of the task. For a multi-class task, a softmax function is applied for normalizing output into a probability of a particular target class. Each probability value ranges between zero and one, and all the probabilities must sum to one. For binary classification task, a *sigmoid* activation function is used at the last fully-connected layer.

2.5.5 Training Convolutional Neural Network

Training a CNN is a mechanism where right sets of kernels are found in the convolution layers and a good set of optimum weights are assigned in the fully-connected layers so that the accumulated loss between predicted outputs and ground truth values becomes minimal on a training dataset. After each round of training (a.k.a. epoch), an update of network is performed (e.g. weights update) with backpropagation. A proper loss function and an optimization algorithm (e.g. gradient descent) play vital roles in the

training process of CNN. During a forward phase on a training data, the performance of a CNN model under certain weights and kernels is calculated with a loss function, and important intermediate values or inputs are cached for utilizing in the backward phase. In a backpropagation, each layer receives a gradient value and returns a new gradient value based on the loss value. These two consecutive phases helps to keep the training of a CNN model organized.

2.5.6 Error and Loss Calculation

A loss function helps to better train a neural network. Given an input and a ground truth label, it measures the difference between these two values. The choice of loss functions and optimizer play an important role in a deep learning model. Before we elaborate on mathematical formulation of a loss function, also known as cost function, we try to give a structural and behavioral representation of a single-layer perceptron in a neural network. In most of the neural networks, the error is calculated by taking the difference between the output value of the perceptron and the ground truth value as follows (2.10) [37]:

$$E(x) = Y_{true} - Y_{predicted} \quad (2.10)$$

This error $E(x)$ is used to calculate the loss/cost $J(w)$ (Fig. 2.11) using different functions such as hinge loss and cross-entropy loss. Different cost functions give different error values for the same output value, thus have a significant effect on the performance of the network model. One of such widely used functions is the mean-squared error, that is measured by taking the squared difference. Besides, different

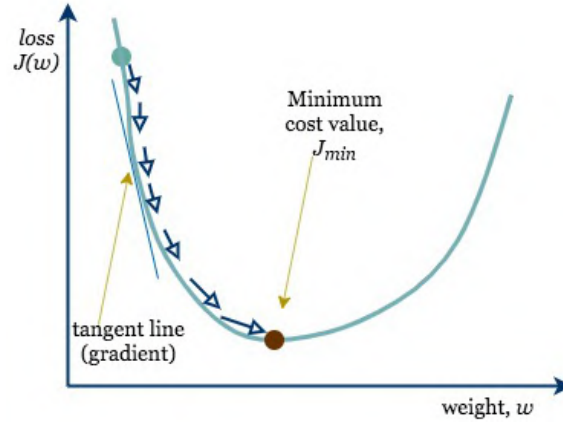


Figure 2.11: Cost curve

loss functions serve different purposes based on the type of tasks (e.g. regression, classification). The loss $J(w)$ is a function of internal parameters such as weight and bias. For accurate prediction performance, this calculated loss value is minimized through backpropagation and an optimization function. The optimization function measures the gradient by calculating partial derivative of the cost function with respect to w . In this process, weights are updated in an opposite manner of calculated gradient. This update cycle is repeated until the minimum of the cost function is reached (Fig. 2.11).

2.5.7 Optimization Algorithms

Optimization algorithms are used to minimize the loss function. In other words, they are used in order to find optimum and faster outcomes by updating internal parameters such as bias and weights.

A) *Gradient Descent:*

Gradient descent is a popular optimization technique that updates learnable parameters (weights and biases) of the model for minimizing its loss value. The

derivative of the cost (loss) function shows the direction in which the curve has the steepest rate of change, and consequently each internal parameter is modified in the opposite direction of the gradient with a step size called learning rate.

$$\begin{aligned} W &= W - \alpha \frac{\partial}{\partial W} J(W) \\ b &= b - \alpha \frac{\partial}{\partial b} J(b) \end{aligned} \tag{2.11}$$

At first, the weight matrix W is randomly given initial values. By using gradient descent method, optimal W and b (bias) are obtained as per the above equations (Eq. 2.11), in order to minimize the training loss. In these equations, α stands for the learning rate, which is an important hyperparameter, initialized before starting training.

B) *Mini-Batch Gradient Descent*

One big issue with gradient descent is that it updates the trainable parameters only after it goes over the entire training dataset. This is not helpful for the system memory when the data size is very big. This problem can be tackled by applying a mini-batch gradient descent algorithm where the entire training dataset is divided into mini-batches of 32, 64, and so on. Therefore *batch_size* becomes a new hyperparameter. Sometimes the mini-batch gradient descent is also referred to as Stochastic Gradient Descent (SGD). Several improved methods like Momentum, RMS prop and AdaM have also been proposed [38].

2.5.8 Overfitting

One fundamental concern in deep learning is the tension between generalization and optimization [39]. Optimization is the mechanism of tuning a model to achieve the best possible outcome on the training dataset. Generalization, on the other hand, refers to how accurate the model performs when an unseen dataset is fed to it after training. At the initial phase of training, both optimization and generalization scores show a correlation, both show a gradual decrease in training loss. The model underfits during this period, as there is still room for improvements and the network is yet to learn more patterns from the training data. After several epochs, the model stops improving and the validation result begins to deteriorate and the model starts overfitting. That means, the model starts to learn unnecessary patterns that are specific to the training data, but unhelpful or misleading when it comes for the unseen test data. Few proposed solutions to reduce overfitting include regularization (e.g. dropout, weight decay), data augmentation, batch normalization and reduction of network size.

In a deep neural network, overfitting poses a fundamental threat. One way to combat this problem is by incorporating a regularization technique called dropout. The key idea here is that at each training phase, a probability is put forth, for whether individual nodes are dropped out or kept. This ultimately reduces the network size from the usual one. As a result, a large overfitting model now repeatedly becomes smaller than the base model but simultaneously trains the large model without causing overfitting [40].

2.6 Popular CNN Architectures

In recent times, the AI community has witnessed a good number of CNN architectures. Those models are so deep that visualizing a complete model has become very difficult. Here, we present some prominent CNN models with custom blocks from the literature.

A) *AlexNet*

The network structure of AlexNet consists of eight sequential layers: five convolutional and three dense layers [41]. AlexNet is the extended version of LeNet5 [42] with few more stacked layers, and it generates 1,000 class labels. This model was the first to implement ReLU as an activation function (Fig. 2.12).

B) *VGG16 And VGG19*

The researcher at Visual Geometry Group (VGG) developed the VGG16 model that has 13 convolutional layers, followed by three fully-connected/dense layers [43]. The architecture used ReLU just like AlexNet, however this network is deeper than AlexNet with more layers, and it uses small kernels (Fig. 2.13).

The model is built with around 138M parameters and the disk size it consumes

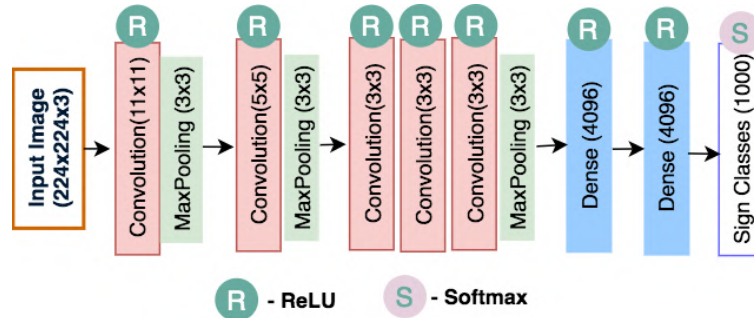


Figure 2.12: AlexNet (2012) architecture

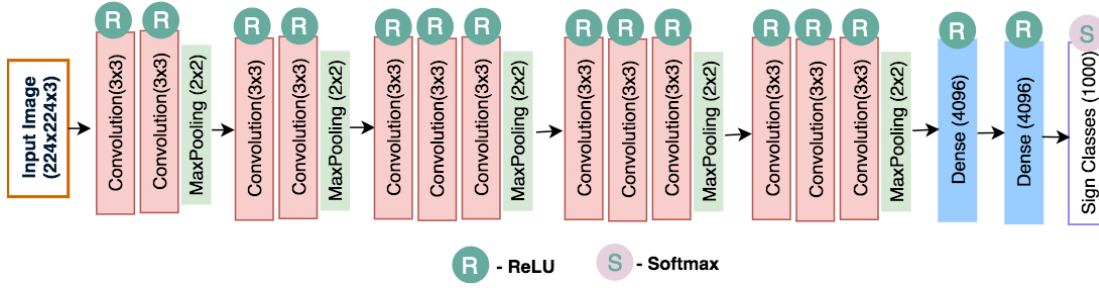


Figure 2.13: VGG16 (2014) architecture

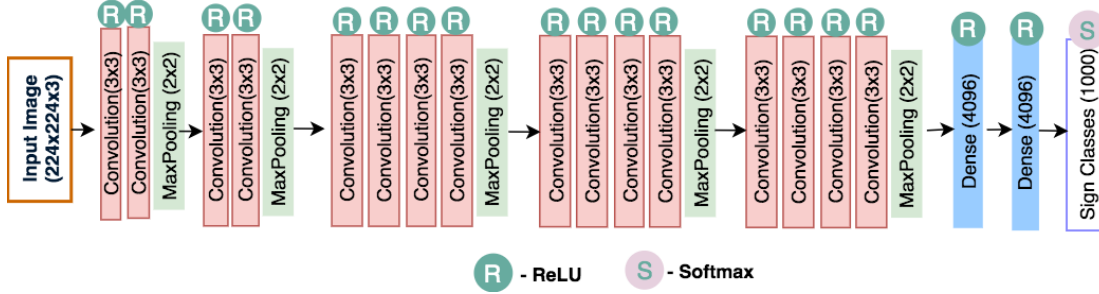


Figure 2.14: VGG19 architecture

is about 500MB. Another model, VGG19 is a deeper variant of VGG16 with one extra convolution layer in the last three convolution blocks (Fig. 2.14).

C) Inception

Inception architecture showed very good performance at comparatively low computational overhead [44]. It has been updated and matured over several versions (from V1 to V4). The final version, V4 is similar to V3 [45] except it incorporates residual connections. Inception-V4 architecture consists of around 39M weights. On the other hand, V3 has around 24M weight values. This version has improvements to loss function, optimizer, and batch normalization in the subordinate network. It has reduced the number of input dimensions to the following layer, and thus it avoids bottlenecks. This updated version uses factorization techniques in order to speed up computations [45]. To be more precise, this

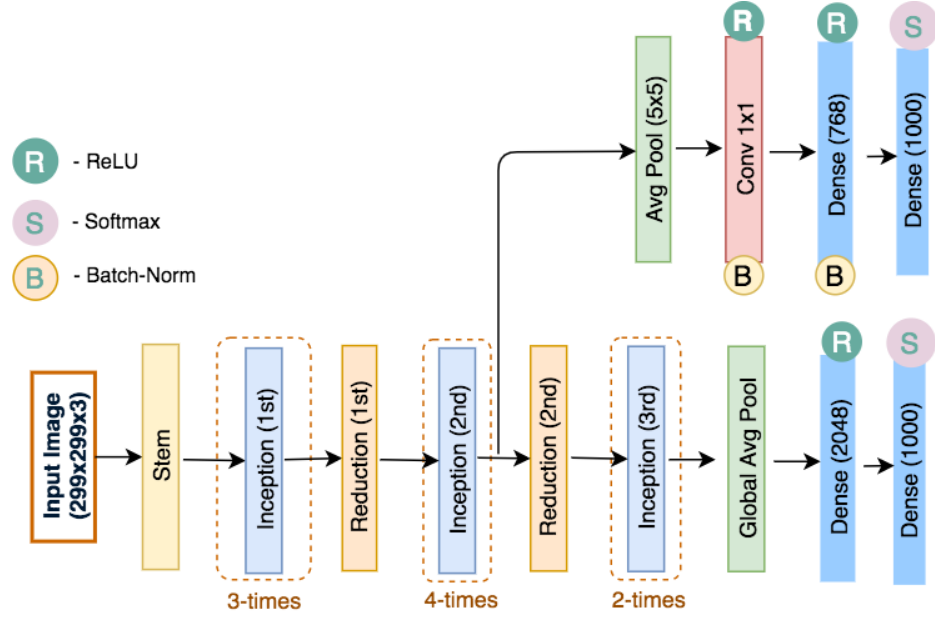


Figure 2.15: Inception block architecture

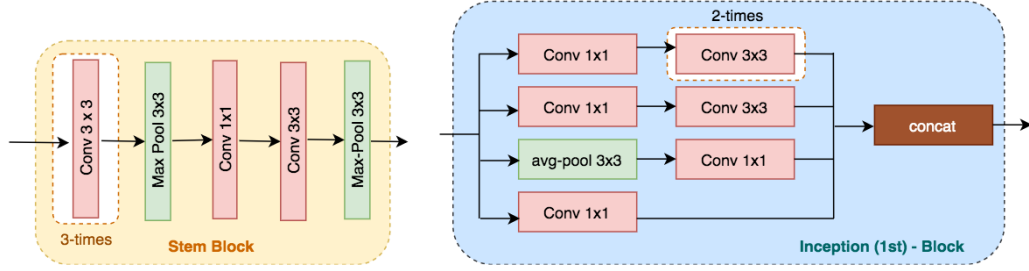


Figure 2.16: Inception stem and 1st block architecture

architecture factorize convolutions of size $n \times n$ into convolutions of of size $1 \times n$ and $n \times 1$ (Fig. 2.17). A new version replaced 5×5 and 7×7 convolutions into several 3×3 convolutions (Fig. 2.16).

D) *Xception*

Xception [46] is an improvement from Inception. This architecture replaced Inception modules with layers of separable convolutions. Xception has the same number of parameters as Inception-V1. Xception uses depthwise separable convolutions that outperforms Inception-V3 [45] which is one of the best models

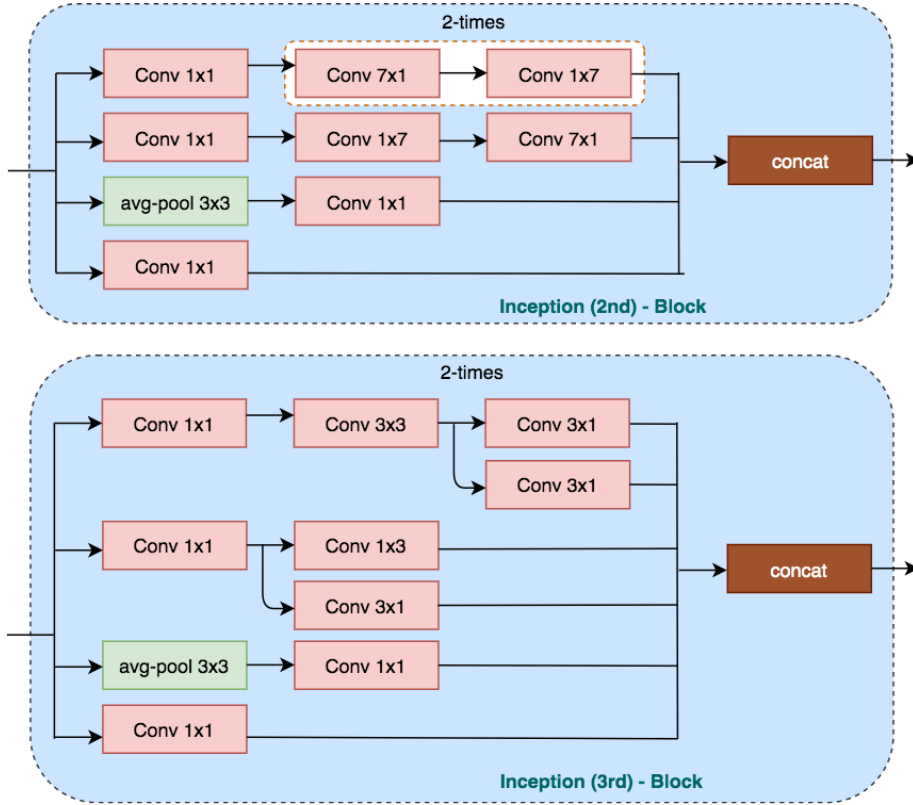


Figure 2.17: Inception 2nd & 3rd block architecture

on ImageNet dataset. The basic difference between Inception and Xception is the order and orientation of convolution layers. Under Inception hypothesis, a convolution layer tries to learn kernels in a three dimensional space, with a single channel dimension and two spatial width and height dimensions. Therefore, a particular convolution kernel maps both the spatial correlations and intra-channel correlations. More elaborately speaking, an Inception unit first sees the cross-channel correlation with a bunch of 1×1 convolutions, then maps the incoming data into 3 or 4 different smaller spaces, and finally with 3×3 or 5×5 kernels it maps all the interrelationships in those reduced 3D spaces (Fig. 2.18). Xception, on the other hand, can be considered as an extreme version of Inception architecture with the aim to completely separate the two

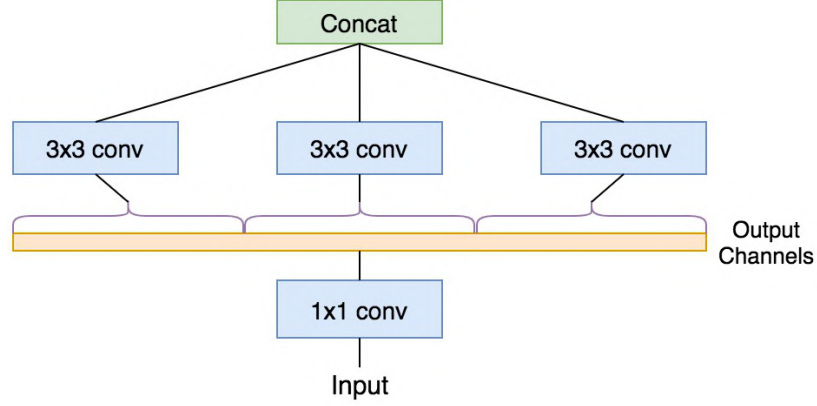


Figure 2.18: Simplified inception module

kind of correlations (spatial and cross-channel). This is performed by using a particular kind of convolution known as depthwise separable convolution or just *separable – convolution* (In *keras* it is known as *SeparableConv2D*).

E) *ResNet-50*

From the architectures described in the previous sections, it is evident that the performance increases as the number of layers is increased. However, as the depth of the network is increased, the performance gets stuck and degrades quickly. This issue was first addressed with ResNet (an earlier version that uses skip/shortcut connections) designed by Microsoft Research while designing for a deeper architecture. ResNet also incorporated batch normalization [47]. However, the model ResNet50 has about 26M parameters. The core of the model consists of convolution and identity blocks (Fig. 2.19). It also uses skip connections in a fine-tuned way. The convolution layers of the model goes as deep as 152 layers, and still it preserves the generalization capability of the model.

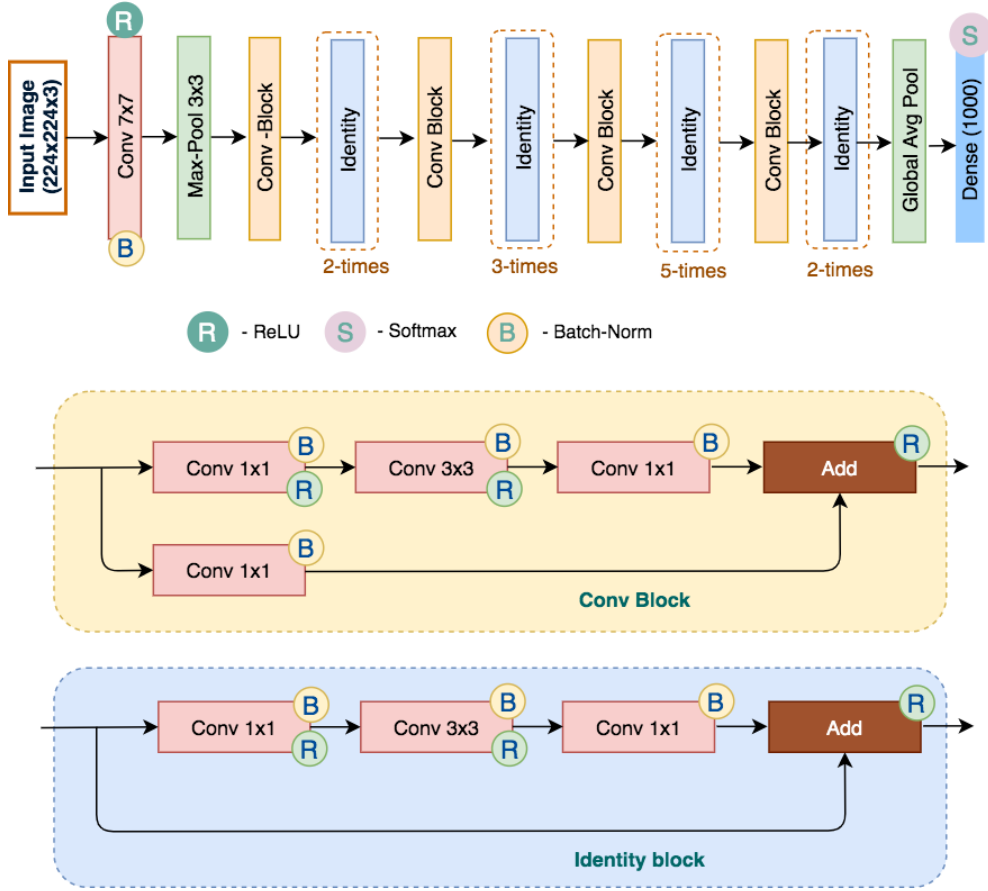


Figure 2.19: ResNet-50 architecture

F) *ResNeXt-50*

ResNeXt50 [48] is the upgraded version of ResNeXts and has about half million less parameters compared to ResNet50. Compared to ResNeXts, this architecture has added parallel branches/towers within each particular model like an Inception network. The total number of towers are counted as 32.

2.7 Transfer Learning

In transfer learning technique, instead of learning from scratch, one may take advantage of the previous learning patterns that have been trained while solving a completely

different problem. In CV, transfer learning is basically applied through the incorporation of prominent pre-trained models. A pre-trained model is a well-established model created and trained on a big dataset by someone else in order to solve a problem similar to the one at hand. The model may not always be completely accurate for the application at hand, but it saves a lot of efforts. The common practice is to use popular models from published works (e.g. MobileNet, VGG, ResNet, and Inception).

While reusing a pre-trained model, the classifier layer needs to be removed first, then a new classifier is added that fits the new purpose. Next, the model needs to be fine-tuned. Basically, there are three strategies from which one is picked: train the whole model, freeze the base convolutional model, or train some specific layers while leaving other layers frozen. In the first case, only the architecture of the pre-trained model is used, and the entire model is trained on a new and large dataset from the scratch. The main idea of the next strategy is to hold the base model in its intact form, only a new classifier is attached to its outputs. The pre-trained model here acts as the core feature instrument, thus reducing the need for high computational power. This approach is also favorable when the size of the dataset is comparatively smaller, and/or the problem at hand is very similar to the pre-trained model. The third and the final strategy is a very realistic one which is to adjust the weights of the pre-trained network by training some layers and leaving the rest as frozen or intact. As far as convolutional layers are concerned, we know that the lower layers extracts general features that are not problem specific, whereas higher layers deal with problem specific features. Therefore, judgment must be made about how much weights we want to adjust by unfreezing some layers and training them. For example, a high-

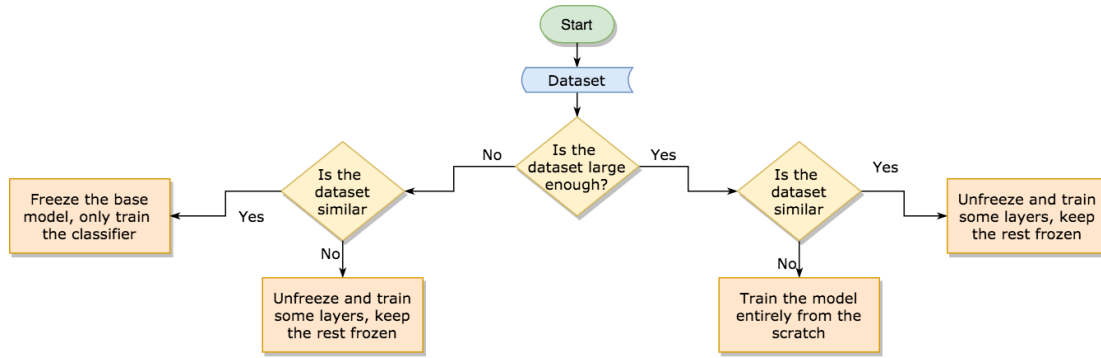


Figure 2.20: Pre-trained model tuning strategy

dimensional small dataset requires only few layers to update their weights in order to tackle the overfitting issue. A large dataset with fewer parameters may unfreeze most of the layers and update the model with accurate weights for the new task, fearing less about the overfitting issue. Figure 2.20 shows a decision-making process in this regard.

2.8 Existing Segmentation Methods

This section describes three popular semantic segmentation architectures: Fully Convolutional Networks (FCN) [4], U-Net [5], and Fully Convolutional DenseNet (FC-DenseNet) [6].

A) *Fully-Convolutional Networks*

Fully Convolutional Networks (FCN) integrate ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classifiers into fully-convolutional networks. Next, they are augmented to make dense prediction by applying pixel-loss and (in-network) up-sampling [4]. Usually an FCN processes an input of any arbitrary size, and generates a corresponding spatial dimensional output, a resam-

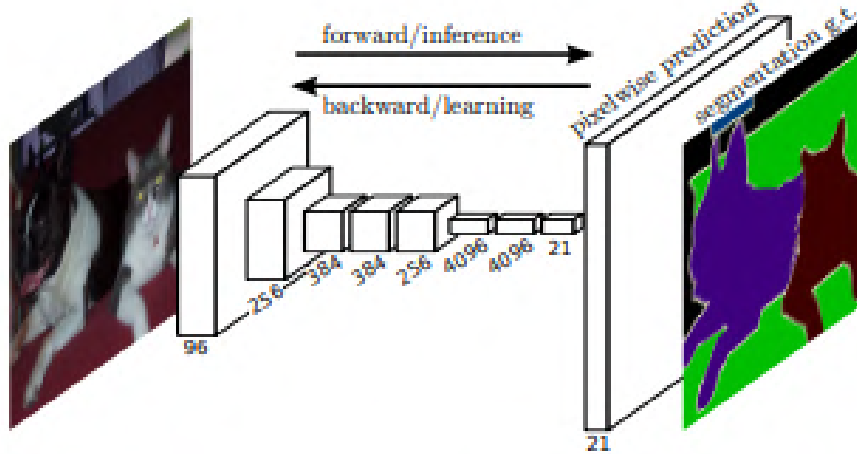


Figure 2.21: FCN architecture [4]

pled one mostly. Through fine-tuning, the model is trained for segmentation (Fig. 2.21). Skip-connections are concatenated between layers in order to integrate coarse, local and semantic display information.

B) *U-Net*

Initially U-Net was modeled for segmenting biomedical images. In those special images, each cell level classification is very crucial. The main challenge in this field is that the training image samples are not adequate. Therefore, the model is designed to address those issues with great care. U-Net is built with fully-convolutional layers and is modified in such a way that it can work on comparatively fewer training samples to produce more accurate segmentation [5]. Due to the small training set, the model utilizes data augmentation by using elastic deformation, so that the integrity of the bio-data is kept. The network architecture of U-Net is built upon two core stages: contract on (left) and expand on (right). The contracting or encoder path is built with two convolution layers, each of which is followed by a ReLU and a downsampling max-pooling

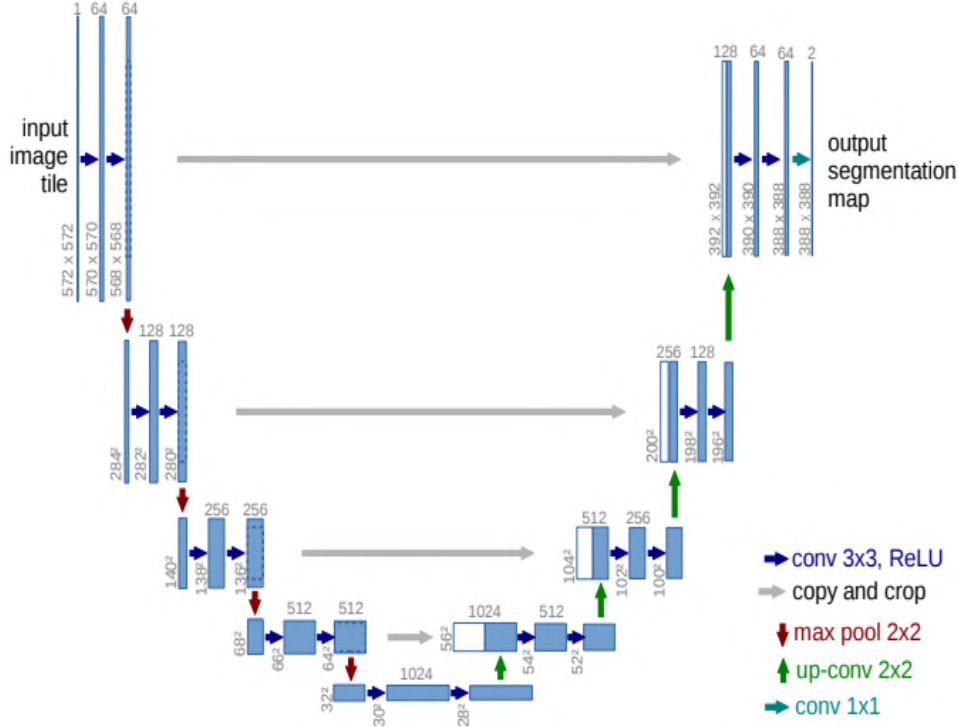


Figure 2.22: U-Net architecture [5]

layer (Fig. 2.22). At each downsampling phase, the feature channels become double in size. The expansive or decoder phase, on the other hand, consists of an up-scaling of the feature channels, followed by up-convolution that reduces the channels by half. The end layer is 1×1 convolution that maps the feature vectors into the desired number of segment classes.

C) *Fully-Convolutional DenseNets*

This architecture is built from Dense Connection Block (DB). The model consists of two paths like most other semantic segmentation structures. The first path is a downsampling with two down transitions, and the second path is an upsampling one with also two transitions but upwards (Fig. 2.23). A circular chain represents a concatenation and an arrow means network connectivity ar-

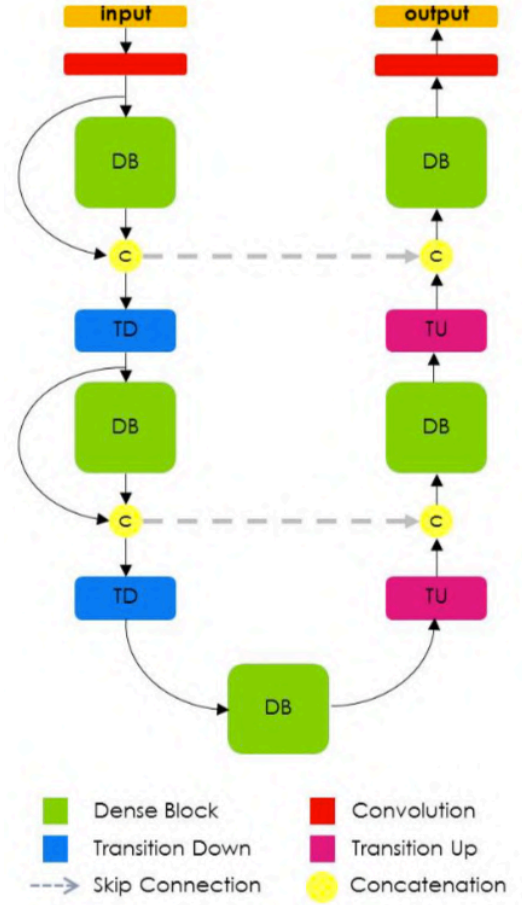


Figure 2.23: FC-DenseNets architecture [6]

rangement. In the model, skip-connections are established and showed with two horizontal arrows [6]. The feature vectors from the left path are coupled with the corresponding feature vectors to the right path, and the two connectivity patterns in both paths are separate. The left path that has the input to a DB block is connected with corresponding output, thus it contributes to a linear growth of the total feature maps; while the right path does not.

CHAPTER 3

RELATED WORKS

CNN-based semantic segmentation models have become the most prominent methods in recent years. We review works on semantic segmentation and hand segmentation in the first two sections of this chapter. In what follows, we explore researches on sign gestures segmentation and recognition under two categories: static and dynamic.

Figure 3.1 demonstrates our literature review areas of preference.

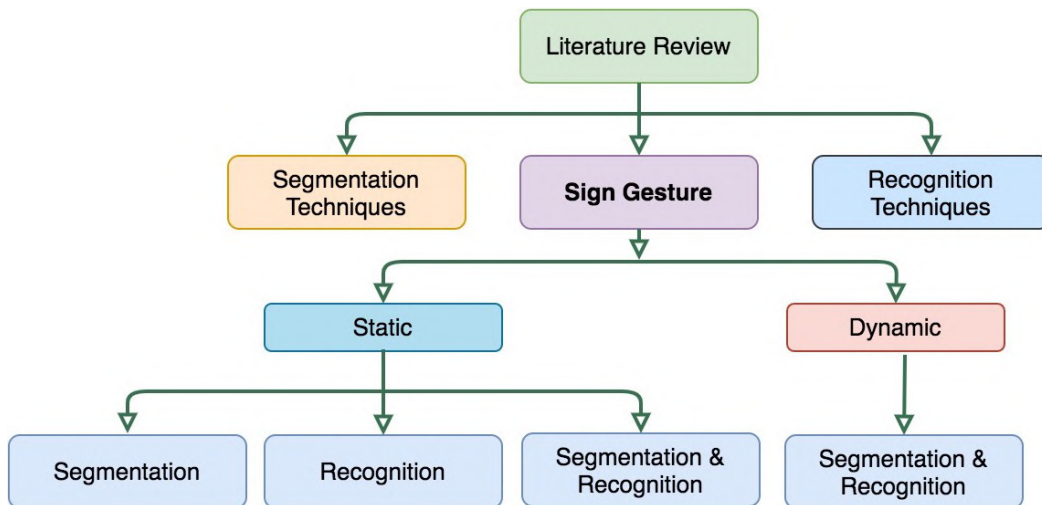


Figure 3.1: Literature review areas of preference

3.1 Semantic Segmentation

FCN-based semantic segmentation models ([49], [4], [50]) showed promising progress on many benchmarks despite the issue with max-pooling layers that resulted in loss of spatial data. FCN makes proper utilization of spatial dense prediction. In the design of such networks, common classification architectures such as AlexNet, GoogLeNet and VGGNet, were used in the literature. In order to combine semantic information, a skip architecture was followed in all those models. Based on the computational budget, the width and depth of those networks were carefully crafted. We also came across a list of models that adopted the Spatial Pyramid Pooling (SPP) technique [51]. Models, such as DeepLab [7], [52] and PSPNet [53], implemented SPP at different grid scales.

3.2 Hand Segmentation

Segmenting hand region from a gesture image is very challenging when the background is complex and blurred. This is one of the hot topics and is still an open interesting problem. In the literature, we put our focus mainly on hand segmentation, gesture with complex background and other conditions, and gesture recognition models. Many novel approaches in image segmentation have been investigated thoroughly. A good number of segmentation models ([12], [5], [7], [13]) are showing incredible performances in object segmentation as summarized in Table 3.1.

SegNet is a fully-convolutional deep network architecture that can understand a scene in an image and then segment it accordingly. The model is powerful in terms

of memory requirement and execution time while making inference. Furthermore, the trainable hyperparameters are comparatively very small and the model is able to perform with low resolution image features for pixel-wise classification. However, the model is primarily for road scene understanding; making it efficient for gesture recognition might require complex tuning.

RefineNet is mostly like SegNet, however it is showing superior performance due to its multi-path refinement network. It captures rich background through a connected residual pooling method. Therefore, RefineNet becomes the state-of-the-art in object recognition.

As one of the objectives of this thesis is to find a suitable hand gesture segmentation model, it is helpful to observe the work authored by Ghanem et al. [54] which successfully applied both SegNet and RefineNet for hand gesture segmentation. It tried to solve the issue with hands occluded with face in a new proposed dataset containing 4,384 annotated image frames. The experiment results showed that RefineNet outperformed SegNet with a formidable margin over the occluded image dataset.

In [55], the authors argued that traditional methods for hand segmentation do not meet high performance in terms of pixel-wise accuracy. The authors built a fresh dataset and proposed a model for semantic segmentation to work on the dataset. The model integrated FCN architecture with a segmentation block. Training was performed with a pre-trained VGG16 network as backbone.

The authors in [56] proposed hand detection and segmentation in a dynamic video. The paper fine-tuned the existing segmentation method, RefineNet, and used this model with two proposed datasets. The authors argued that the available dataset was

developed in a controlled laboratory setting. One of the datasets collected egocentric hand videos from YouTube and contains hands in a complex setting. The other dataset, HandOverFace, contains occluded hand objects. The paper also used two other popular datasets which are EgoHands [57] and GTEA. In order to perform activity recognition, the paper trained a CNN model using hand masks generated by a RefineNet-based model. To measure the segmentation accuracy, the authors annotated a subset of EgoHand dataset and achieved an *mIOU* score of 58%. The main issue with this paper is that the contribution to come with new segmentation model was somewhat minimal since fine-tuning using RefineNet is very common in this field.

We found another egocentric activity recognition in [58]. Here, the authors used a pre-trained CNN model for recognition and a convolutional Long Short Term Memory (LSTM) model for segmentation. The model used spatial attention maps from video frames to locate segmentation objects and used GTEA bench-marked dataset for comparing their segmentation performance.

Skin-color segmentation has been a prominent practice in the area of hand gesture recognition systems. However, some researches also used other segmentation methods like thresholding [59] and background subtraction [60]. Ghanem’s work [54], just described earlier, raised another important issue regarding sign gesture, which is hands occluding with face. Several researches worked on occlusion in sign language.

Table 3.1: Summary of segmentation methods used in the literature

Ref	Year	Method	Tools	Strength	Challenge
[28]	2018	Skin-color	Dynamic detector based on face-skin tone in YCbCr space	Resolves occlusion and misclassification	Face-pixels affect accuracy. Euclidean Distance for recognition
[61]	2019	Skin-color	Multi-view frames	Covers bimanual gesture, tackles self-occlusion	Only 6 gestures covered
[62]	2017	Skin-color	Faster-RCNN for Region Selection & Patch-based CNN skin detector	Clutter Environment Detection	Does not tackle blur, complex background, etc.
[63]	2020	OR-SkipNet	Semantic segmentation for skin regions	Higher localization accuracy	Model did not compare with standard architectures
[60]	2015	Hybrid	Thresholding & Skin-color & Adaptive background subtraction	Comprehensive features for hand object	Small dataset (only 3 video samples), Model not generalized
[12]	2017	SegNet	Semantic Segmentation	Computationally efficient, less memory required	Road-scene segmentation only, less efficient for gesture recognition
[13]	2017	RefineNet	Chained residual pooling with CNN	Combines high-level semantics and low-level features for high resolution.	Computationally costly
[59]	2017	Thresholding	Binary image fed to CNN	Denoising autoencoder takes care of corrupted gesture	Inefficient in complex background, occlusion

While most of the papers discussed above recognize single handed sign gesture, the authors in [61] described technique to deal with gestures which were bimanual in nature. The article used three cameras from different directions, in order to tackle the complex issue of the self-occlusion problem in this dexterous type of gestures. However, it may cause conflict among different recognized gesture images from multi-views. Therefore, the paper used a fusion method of outcomes from the multi-view points. The authors also employed both color and shape features, then trained the classifiers with gestures of the bimanual dataset. Afterward, a fusion method was applied with weighted sum to ensemble outcomes predicted from individual classifiers. The paper claims better performance of multi-view gestures over single-view results. The approach may be applicable to interactive applications, like multimedia video game or sign language. However, the experiment consisted of only six gestures. Therefore, the outcome of the experiment may not be robust for diverse gestures. The solution to the occlusion problem claimed by the authors may even lack credibility to a number of complex hand orientations. Another occlusion solution was proposed in [28], where

the authors designed a SLRS that simply converted signs of arabic words into corresponding texts. For segmentation, a face-color tone based skin detection was used. The article proposed a tracking technique of skin-blob in order to find and follow the hands. However, the system was evaluated only with 30 individual words to claim a recognition accuracy of 97%. Furthermore, it is similar to [64] except that the authors claimed that they took care of the occlusion problem in a better and economically viable way. But the size of dataset was very small, and the model did not apply any sophisticated classification algorithm other than Euclidian distance. Table 3.1 gives a summary of several segmentation models in the literature.

3.3 Static Sign Gesture Segmentation and Recognition

Rastgoo et al. [22] used a deep learning based approach, Restricted Boltzmann Machine (RBM), in order to carry out the recognition of sign language from static images. The article evaluated a generative RBM model for gesture recognition. The model used both RGB-D and RGB as two modalities in the input images in three distinct forms: original, cropped and noisy-cropped. For detecting hands, five cropped segments of the input images in three distinct forms were fed into CNN model. Then, three forms of segmented hand were created and fed to three RBMs. Finally, the outputs from these models were fused and fed to a final RBM for recognition. This multi-modal approach was trained with the ASL alphabets and digits from four available datasets. The robustness of the model was evaluated against noise.

The article [65] used a YOLO network architecture for hand segmentation and a

DNN for sign alphabet recognition. The authors used multiple variants of their model to derive an optimum one. They trained their model with a combination of two ASL datasets: one publicly available [66] and one custom-developed. The model achieved a recognition accuracy of 98.07%.

Sharma et. al. [67] used a deep Eigen space based CNN model in their ASL Recognition system. Binary mapping with RGB-D images were used for hand segmentation. The paper used a dataset by [68] and achieved a recognition performance of 91.4%.

Song et al. [69] described a model for converting sign language to emotional speech (Mandarin and Tibetan). The main motivation for this paper seems to be the fact that emotions can contribute to communication accuracy. Therefore, the model incorporated the recognition of facial expressions. According to the model, first a context-dependent tag along with related emotional labels of the sign language were formed from the identified sign language and face expression. Meanwhile, an HMM backed speech fusion was trained via a neutral and an emotional corpus. Afterwards, the speech synthesis was performed as per the labels and tags of sign text so that the sign language and face expressions were converted into emotional speech. The authors claimed that the experiment showed the Estimated Mean Opinion Score (eMOS) score to be 4.0 for the transformed Mandarin/Tibetan emo-speech. The paper used the PAD 3-D model for assessing expression and synthesizing the speech. The paper made the conclusion that the conversion performance was good, since the Euclidean distance of the PAD scores were small.

The article [70] used CNN in two stages and fused the outcomes from the network. In the first phase, hand regions were identified by performing segmentation at pixel

level; the next phase helped determine the hand gesture. The paper claimed that its model to be robust against complex background or different illumination.

Xie et al. [71] proposed an Inception-V3 based method for recognizing static gestures of 24 alphabet in ASL. The authors claimed that their system can extract features automatically for gesture classification. They used two pre-trained Inception-V3 models: one for RGB and one for RGB-D, then fused them for recognition. The paper used publicly two public datasets and achieved a combined recognition accuracy of 91.35%.

The paper [72] applied six image descriptors for 36 hand gestures recognition, and compared the accuracy and CPU speed in a comprehensive manner. The authors suggested that for better recognition result, the sizes of neural network and feature vector played a crucial role. The feature vector used by the paper had minimum computational overhead and was used for real-time recognition.

Roy et al. [62] used a dual-modal deep learning based method for detecting hand and face by applying patch-based skin-color segmentation. The paper used CNN based model over Faster-RCNN to suppress false positive detection outcomes. However, the model seemed to be time consuming and less robust against low illumination, blur, shadows, etc.

Elsayed et al. [60] proposed a hybrid method using adaptive complex background subtraction and thresholding with respect to skin-color. However, the model may lack generalization, only three video samples were applied with the hybrid.

The authors in [73] used skin-color segmentation for detecting palm, face, etc. from webcam images in real-time. The gesture features were obtained by a pre-trained

CNN. The authors discussed about only four classes, but they did not explicitly discuss about the sign language performance. This was a good step for segmentation, however the gesture detection block did not seem to be matured enough for dark gestures/skin-color due to false negative detection.

Denoising autoencoder is a special variant of autoencoder; its network is applied to learn the restoration of noise inflicted input data (image) in the output layer [74]. Such type of networks gain more useful features from training samples compared to the ordinary autoencoder. The authors in [59] used a stacked denoising autoencoder to learn complex sign gesture classification. Using a public ASL dataset, the paper applied a convolution neural network in three stages with different hyperparameter setups. However, the model did not deal with common issues such as occlusion and complex background.

The paper [75] presented a weighted HMM/kNN model to collect signer-independent gesture features; where complexity was present in the training data due to its enormous volume from a range of different people. The method included two distinct blocks, one for classification, and the other for recognition. In the first part, kNN was applied to learn the training data. Different weights were assigned to various sign language features based on contributions. For the recognition block, the weighted results from the first block were taken as input along with temporal data and fuzzy inference to HMM. The overall performance of the model was not very compelling. The paper claimed their work was signer-independent, the dataset image included only the palm region and a broad variations in samples were missing.

In an ideal scenario, skin-color segmentation should be easy to achieve perfect

accuracy due to its dissimilarity to background-color. However, in real life we face adverse conditions like poor illuminations, skin-like other pixels, black-skin and different backgrounds etc. While most of the proposed methods in the literature used a costly preprocessing phase in their models, the authors in [63] offered a deep CNN on top of an exterior residual skip network. This semantic segmentation approach helped transfer the edge information directly from an initial layer to the end layer and thus helped enrich the features with spatial data. The model seemed similar to SegNet [12] and ResNet [47].

The work in [76] gave a gesture recognition model for RGB/RGB-D ASL images by fine-tuning a VGG19 architecture. The proposed model applied feature concatenation by using two separate VGG19 models: one trained with RGB images and the other trained with depth images. Since the pre-trained base model was for 1000 classes, the authors of the paper replaced the fully-connected layer with 24 classes. The model showed an accuracy of 94.8% on ASL dataset.

Table 3.3 give a summary of several gesture recognition models and their experimental results.

Table 3.2: Summary of static sign gesture segmentation and recognition in the literature

Ref.	Year	Classification	Segmentation	Feature Extraction	Dataset	Acc (%)
[70]	2018	Fusion of two CNNs	CNN base hand segmentation	ASPP	HGR1 OUHANDS	83.10 93.75
[65]	2021	DNN	YOLO network	DNN	ASL by [66]	98.07
[22]	2018	Faster R-CNN	Five crops of hand	Restricted Boltzmann Machine (RBM)	Massey-G12 Surrey NYU ASL-Finger	99.31 97.56 90.01 91.83
[67]	2020	Eigen space CNN	Binary mapping of RGB-D images	CNN filters	ASL by [68]	91.04
[75]	2015	KNN	HMMs	OpenCV library	Private ASL	46.48
[71]	2018	Inception-V3	RGB/RGB-D processed in separate networks	CNN filters	ASL by [77]	91.35
[72]	2018	Configurable 3-Layered perceptron network	grayscale and binary palm images	Using six image descriptors	Massey-G12	98.61
[73]	2018	Deep Learning classifier	Skin-color/Face	CNN Transfer Learning for extracting features	Private ASL	94.00

3.4 Static Sign Gesture Recognition

Barbhuiya et. al. [78] applied modified AlexNet and VGG16 for feature extraction of sign alphabets images. A multi-class SVM was used for final recognition of signs. The authors explored different system-based performance analysis and achieved a final accuracy of 99.82% on a publicly available dataset.

Suresh et al. [79] applied CNN for classifying six sign classes from 3,060 images. The proposed model used two distinct optimizers such as SGD and AdaM to get the almost identical results. The dataset barely was sufficient for the deep learning model and the authors did not provide some hyperparameter values like *batch_size*, *epoch*, *learning_rate*, etc. Their model used only two convolution layers and it was trained on only six sign gestures.

The paper [33] used SIFT and an SVM classifier for recognizing BSL words and alphabets. The authors achieved a recognition accuracy of 88.87% for BSL words and

Table 3.3: Summary of static sign gesture recognition in the literature

Ref.	Year	Classification	Feature Extraction	Dataset	Acc (%)
[78]	2020	SVM	Pretrained AlexNet and VGG16	Massey-G12	99.82
[79]	2019	Two CNN models	Convolution filters	Private hand gestures	99.50
[76]	2019	Fine-tuned VGG19	Convolution filters	ASL by [77]	94.80
[33]	2015	SVM	SIFT	BSL word BSL alphabets	88.89 88.34
[80]	2019	Pretrained AlexNet	Convolution filters	Moeslund [81]	98.40

88.34% for letters. They compared their model with a kNN-based classifier.

Ozcan et al. [80] fine-tuned a pretrained AlexNet model and used it for recognizing hand gestures. The model achieved superior recognition performance of 98.40% on Moselund’s gesture dataset [81].

3.5 Dynamic Sign Gesture Recognition

The paper [82] showed a hybrid model built with 3D-RNN and CNN, that learns additional data along with sequential information for gesture recognition from continuous videos. A dataset of ASL videos were fully-annotated on temporal domain. The continuous clips with equal semantic sense were singled out by using sliding window for segmentation. The ASL dataset consisted of two categories: sequential and sentence videos. In the first category video, a person performed a number of certain ASL words; the other category video had a person performing ASL sentence with multiple sign words. However, the experiment achieved merely a 69.2% accuracy.

Abid et al. [83] offered another dynamic recognition method of sign language for

home-interactive systems. The model was built on two distinct sub modules: image processing (IP) and formal grammar (stochastic linear). The first module of the system recognized hand gesture from a video; the other module analyzed the gesture sequences and checked the correctness of the syntax. Furthermore, the paper claimed that the system can also make predictions about the missed out sign gestures.

We have also observed a number of promising works on Arabic Sign Language (ArSL). In [84], the authors built their recognition system of ArSL by using CNN for 3D vision on their dataset, which contained 25 Arabic sign words from a number of individual signers. Each gesture was represented as a class and its surrounding defined its own 3D CNN model that helped extracting temporal-spatial feature sets along with the motion data encoded in sequential frames. The outcome was not free from overfitting problem, as the size of the dataset was significantly small for a CNN model.

Table 3.4: Summary of dynamic sign gesture segmentation and recognition in the literature

Ref.	Year	Classification	Segmentation	Feature Extraction	Dataset	Acc (%)
[85]	2019	SVMs	SkinMask, YCbCr and fusion of two CNNs	Morphological ops	Private ASL word	97.28
[86]	2017	Fusion (Kinect, LMC)	Face and hand key points	3D coordinates features outputs from Kinect and LMC	Private ASL (single) ASL (bimanual)	96.05 94.27
[69]	2018	SVM, HMM and CNN	Facial and hand area	DNN	Cohn-Kanade JAFFE	90.70 94.70
[84]	2018	3D-CNN	Depth map with stabel sign images	Convolutional kernels	Private ArSL	94.00
[82]	2018	RNN along with CNN	C3D models (special, motion, depth)	Semantic equality-based sliding windows	Private ASL sentence	69.20
[83]	2014	BOFs, SVM	Palm gesture clips	3D histograms of descriptors (HOG)	Private ASL sentence	98.65

Rahim et al. [85] proposed a segmentation technique as a combination of Skin-Mask and YCbCr, with an aim to identify arms. The feature fusion of CNN was used to extract corresponding features. The model took input signs word from live video capturing from a webcam, segmented hand gestures and finally extracted features by applying CNN feature fusion for final gesture recognition with an SVM classifier. Though the average accuracy of the recognition was about 97.28%, the dataset contained samples of only three persons, and almost all the samples of a particular sign word had identical background making them look-alike. Furthermore, it seemed that the authors chose contrasting backgrounds for the ease of segmentation with skin-color and YCbCr.

CHAPTER 4

PROPOSED METHODOLOGY

In this chapter, we explain the details of the proposed methodology for hand gesture segmentation and recognition. The following section provides a high-level overview of our proposed system framework. Section 4.2 explains the details of the segmentation phase and Section 4.3 explains the details of the recognition phase. Next, Section 4.4 presents the whole system architecture that combines the two phases. The creation of a ground-truth dataset is explained in Section 4.5. Finally, Section 4.6 describes other data preparation and techniques used in this thesis for enhancing the results.

4.1 Proposed System Framework

This thesis proposes a novel network architecture dubbed as S3RNet, for the recognition of sign gestures with semantic segmentation. Figure 4.1 demonstrates our proposed system framework. Two core components of the framework are semantic segmentation and sign recognition. The work-flow of our system is marked with numbers. After we create a ground-truth gesture dataset, we apply pre-processing methods

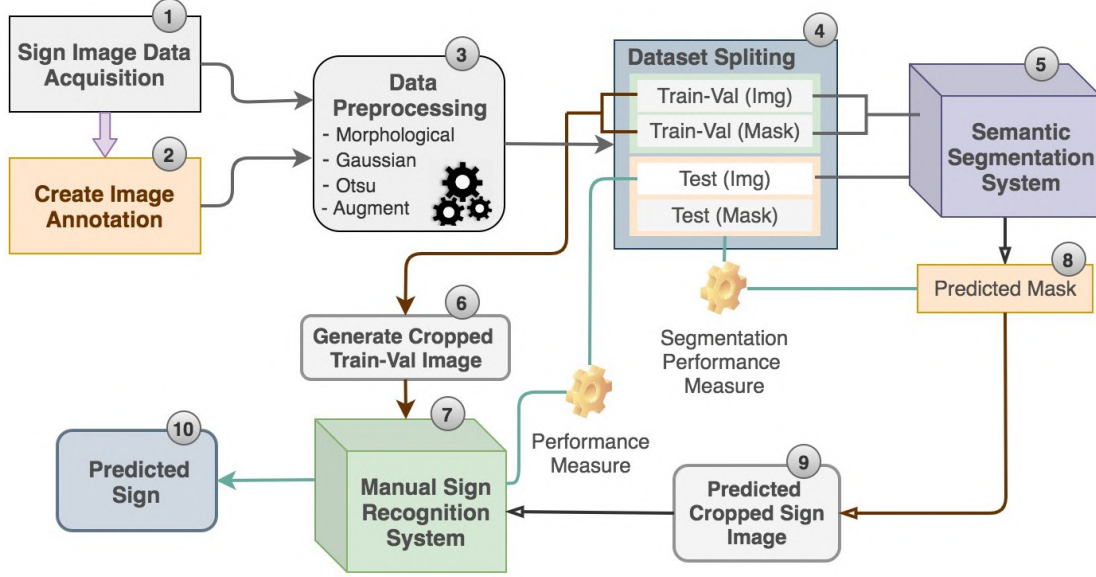


Figure 4.1: Proposed system framework

such as smoothing, morphological operations, thresholding, and augmentation. Data augmentation is applied through systematic arguments tuning. Then, the gesture dataset is split into training, validation and testing sets which are used to construct and evaluate our proposed models. The performance of the semantic segmentation is measured by comparing between the predicted masks it generates for the testing set to the ground-truth masks. Likewise, the performance of the recognition component is measured by comparing the predicted sign labels of our model when we feed it with the predicted mask-cropped sign gesture images that are generated for the testing dataset. Considerable efforts are made to explore different CNN-based segmentation and recognition models. For the recognition block, we design one transfer learning-based model dubbed as TRNetV16, and two custom CNN models dubbed as B4CN and WB3CN. Since we could not find a suitable ground-truth dataset for hand gesture segmentation, we hand-crafted an existing dataset (NUS-HII) with complex

backgrounds to create ground-truths. More details about various components of the proposed framework are explained in the following subsections.

4.2 Semantic Gesture Segmentation Model

Our proposed custom-based semantic segmentation component and its design detail is provided in the final part of this section. The building blocks and core functionality of this segmentation component is demonstrated in Figure 4.6. Before describing this component, we better explain a couple of related concepts and techniques for better understanding. Hence, we begin Section 4.2.1 by defining semantic segmentation in basic terms. In what follows, we describe two important building blocks of the segmentation component such as Encoder-Decoder and Skip-Connection (Section 4.2.2 and 4.2.3, respectively). The network architectures of transfer learning-based and custom-based semantic segmentation are also described in Section 4.2.4 and 4.2.5.

4.2.1 Semantic Segmentation

Semantic segmentation, also known as pixel-wise segmentation, is an emerging field in HCI. The main functionality in semantic segmentation is to group all the pixels of an input image into predefined segment classes. This technique has a wide range of applications covering from scene recognition, autonomous driving, and inferring relationships between objects.

Unlike image classification, in semantic segmentation, the output of the network is also an image. For example, in the outdoor photo shown in Figure 4.2, semantic



Figure 4.2: Labels in a semantic segmentation

segmentation will split pixels with separate IDs for people, sky, ground and plants. This segmentation technique is not like object detection, as it does not anticipate any objects with bounding-boxes. Also no distinctions have been made between several instances of the same class. For example, in Fig. 4.2, multiple persons in the photo are labeled as person. In semantic segmentation, the objective is to analyze an input image of size $W \times H \times C$, where W and H are the width and height and C is the color channels, and produce a $W \times H$ matrix that contains anticipated IDs of corresponding classes to all pixels. Recently, deep learning based semantic segmentation has been proven to be the state-of-the-art for object segmentation. Thus, the early segmentation methods based on basic features have fast been overwhelmed by this modern segmentation technique. In this thesis, our segmentation component processes a sign image with a complex background and generates a predicted segmentation mask having only two classes representing hand and background.

4.2.2 Encoder-Decoder Architecture

We use Encoder-Decoder, a specialized CNN, as the backbone architecture for our proposed semantic segmentation component. The basic building blocks of a CNN architecture are provided in the background Section 2.5. Nonetheless, how it is different

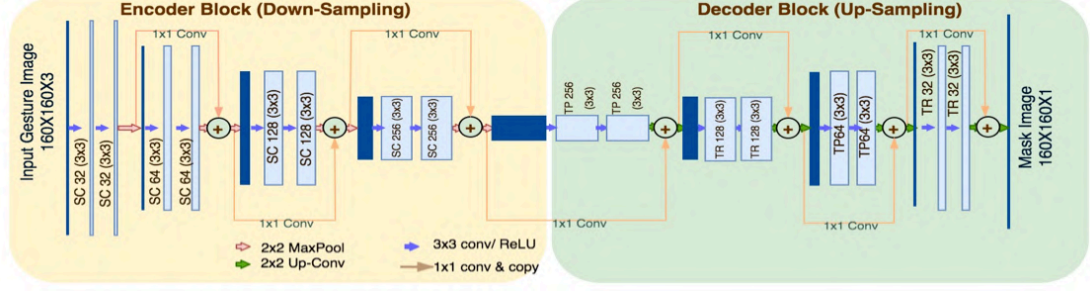


Figure 4.3: Encoder-decoder structure of the proposed S3RNet model

from regular CNN or how it is engineered for segmentation purpose is the core objective of this section. The architecture of a CNN-based segmentation model consists of convolution layers, activation functions, batch normalization, and max-pooling layers (Fig. 4.3).

At an initial layer, the network encompasses small regions with low-level information in an image whereas at an upper level the neurons learn patterns for a broad region of the image. Therefore, as more network layers are added in the encoder block, the size of the input image gradually decreases while the channel size keeps on increasing. The pooling layers are responsible for the gradual downsampling. For the image classification task, fully-connected layers are applied in order to map the spatial data to a fixed length class vector. This removes all the spatial information. However, the spatial data of the image is required for semantic segmentation, therefore fully-connected layers are not used in this network, thus making it fully-convolutional. The convolution layers connected with pooling layers generate a low-dimensional tensor that consists of high-level spatial information. Now high-resolution segmented output images must be generated from the high-level spatial data. In order to achieve that goal, more convolution slabs along with upsampling layers need to be added in the

decoder block so that the size of the spatial tensor becomes bigger. As the resolution is increased, the number of channels is reduced with the low-level spatial data. This whole model is known as an encoder-decoder structure. The encoder part is responsible for downsampling the input, and the decoder layers contribute in upsampling (Fig. 4.3). As the network is trained for semantic segmentation, the encoder generates a tensor output consisting of feature maps. The decoder, on the other end, considers this data and generates a segmentation map.

4.2.3 Skip Connection/Short Circuit

Another important building block of segmentation is the skip-connection or convolution slabs. When the layers of encoder and decoder are stacked together, there is a possibility of losing low-level details. As a result, the segmentation boundaries in the maps generated by the decoder may not be appropriate. In order to gain the lost information, the decoder layers need to access the low-level information generated during encoding in different layers. This is accomplished by skip-connections, where the intermediate generated data of the encoder layers are directly linked to the inputs to corresponding intermediate decoder layers.

4.2.4 Transfer Learning-based Segmentation

We applied DeepLabV3+ [7] in our transfer learning-based model (DLSR) for semantic segmentation as shown in Figure 4.5. This model is basically an encoder-decoder architecture. However, in the encoder part, it applies Atrous Spatial Pyramid Pooling (ASPP) with different rates to capture contextual information in the feature maps.

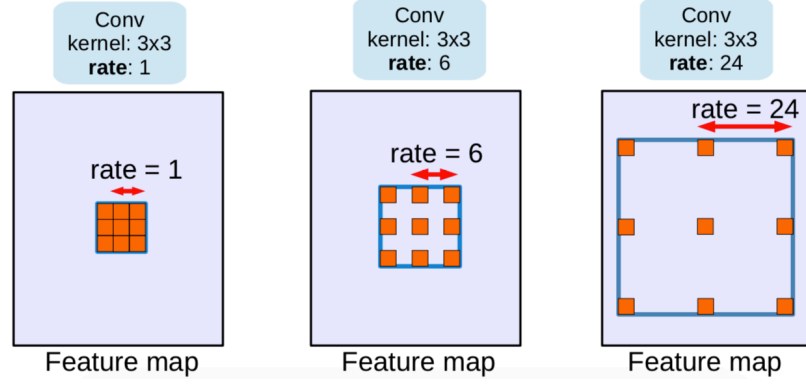


Figure 4.4: Atrous convolution at different rates [7]

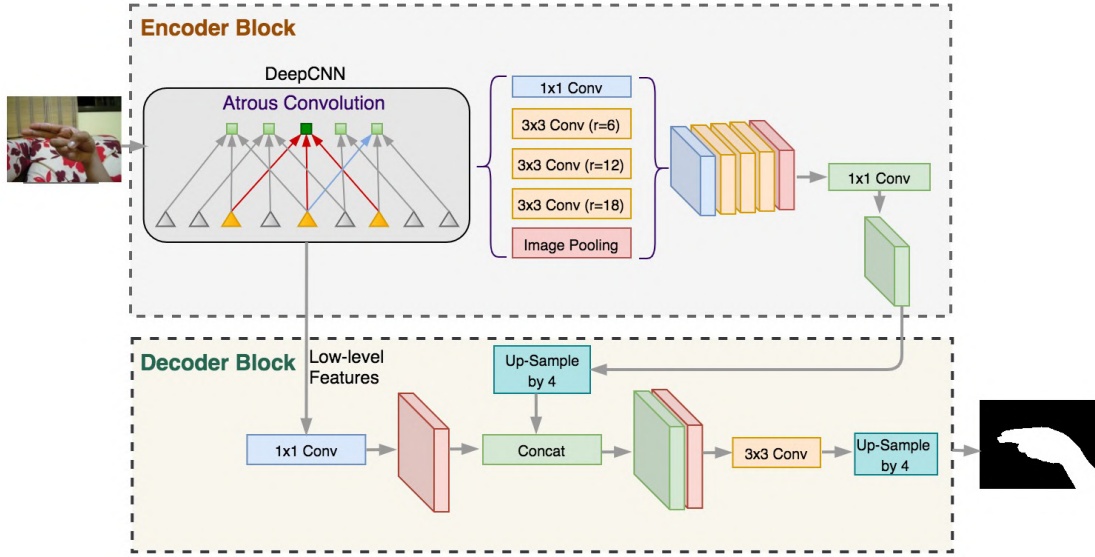


Figure 4.5: Semantic-segmentation using DeepLabV3+ [7]

As shown in Figure 4.4, for a 2D image, the atrous convolution can be defined mathematically by the following equation [7]:

$$\mathbf{y}[\mathbf{i}] = \sum_{\mathbf{k}} \mathbf{x}[\mathbf{i} + \mathbf{r} \cdot \mathbf{k}] \mathbf{w}[\mathbf{k}] \quad (4.1)$$

where $\mathbf{i}$ is the location along $\mathbf{y}$ -axis, $\mathbf{w}$ is a filter, $\mathbf{x}$ is the input feature map, and $\mathbf{r}$ is the atrous rate, which is related to the stride and plays a crucial role as the input image is sampled with different r . In CNN, repeated actions like max-pooling and use of strides in successive layers gradually reduces the spatial dimension of the final feature

maps. Atrous convolution is applied to achieve the same objectives with some added benefits such as maintaining field-of-view in successive layers. In the decoder phase, the feature maps are upsampled bilinearly by a multiple of 4 and then fused with low-level features. Finally, 3×3 convolutions are used in order to get more refined features followed by a similar upsampling. In DLSR, as the network-backbone, we used the weights of an Xception_65 model which is pretrained on PASCAL VOC 2012 dataset. The atrous rates used during training were 6, 12 and 16. First, converted our gesture dataset in PASCAL VOC format and then trained the model for 30000 training steps with a batch size of 1.

4.2.5 Custom-based Semantic Segmentation

Another semantic segmentation component based custom module is proposed as shown in Fig. 4.6. It is designed with a deliberate goal that the architecture generates more accurate semantic segmentation masks even for a small number of training samples. This model is motivated by depthwise separable convolutions of Xception [46] while designing encoder-decoder networks of the custom-based semantic segmentation module. As shown in Fig. 4.6, a *depthwise* convolution first maps the spatial (width and height) correlations independently for each output channel space, and then a 1×1 *pointwise* convolution maps intra-channel relationships. Our hand-crafted annotated dataset consists of about 2,000 sign images. The entry block of the encoder consists of a convolution with 32 filters. Each of the next three levels of the encoder uses two consecutive *SeparableConv2D* with filter sizes 64, 128 and 256 respectively. The convolution of each level of both encoder and decoder is followed by a batch normal-

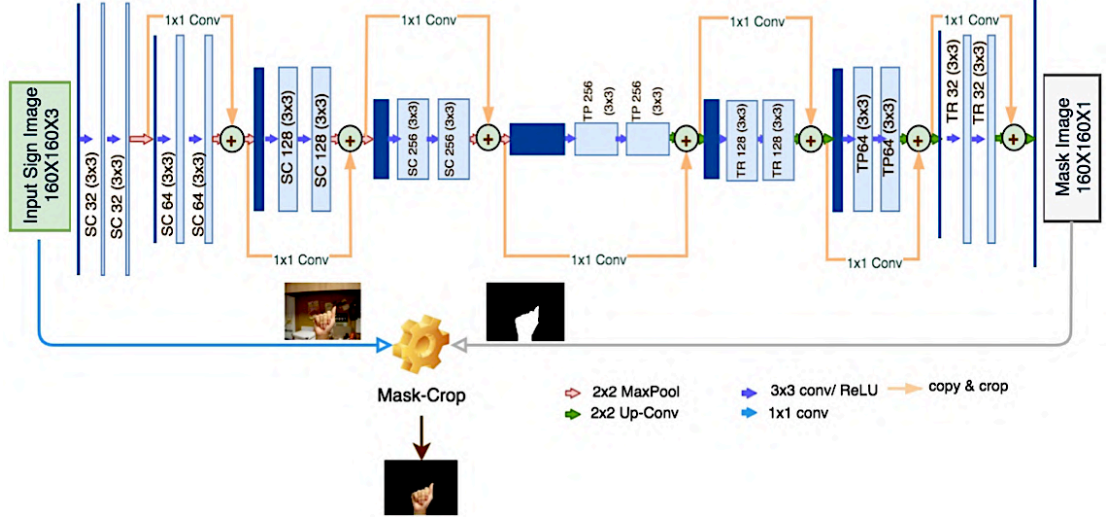


Figure 4.6: Custom-based semantic segmentation component

ization layer and a ReLU activation layer. After two consecutive *SeparableConv2D*, a max-pooling operation is performed in the encoder phase (downsampling). Likewise, after two consecutive *Conv2DTranspose* an up-sampling operation is performed in the decoder phase. A residual skip-connection is made between layers of corresponding encoder and decoder. The input image dimension to the model is $160 \times 160 \times 3$ and the predicted output mask dimension of the segmentation block is $160 \times 160 \times 1$.

4.3 Manual Sign Recognition Models

As our segmentation component generates semantic masks, we need a recognition block for final sign gesture classification. Here, we propose one transfer learning based model (TRNetV16), and three custom-designed CNN models WB3CN, B4CN, and MC-SLR. We elaborate on each of these models in the following subsections.

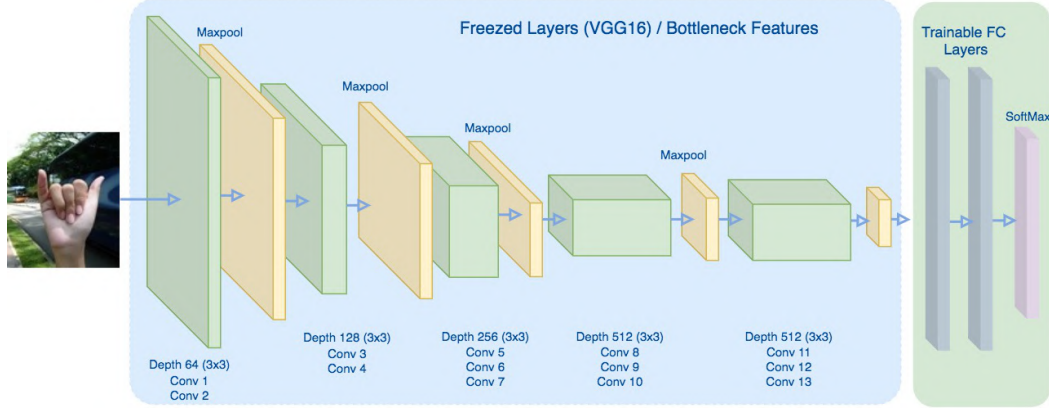


Figure 4.7: TRNetV16 architecture

4.3.1 TRNetV16

Our first recognition model dubbed as TRNetV16 utilizes the top layers of the pre-trained VGG16 architecture [43] which is trained on ImageNet dataset. TRNetV16 excludes the fully-connected layers of the pretrained model in order to extract features of the correct dataset. We first use the internal 16 layers of VGG16 model to generate bottleneck features (Fig. 4.7). The resulting features are utilized to train the fully-connected block for the final sign gesture recognition.

4.3.2 WB3CN

Our first custom-CNN recognition model is WB3CN (Fig. 4.8a). It uses four convolution layers, two max-pooling layers, and two fully-connected layers. After two consecutive convolution layers, a 2×2 max-pooling layer is appended. No batch-normalization is applied after any convolution layer. Three dropout layers are appended, of which two are added after a pair of the convolution layers with 25% dropout ratio. The last dropout is added after the fully-connected layer with a ratio of 50%.

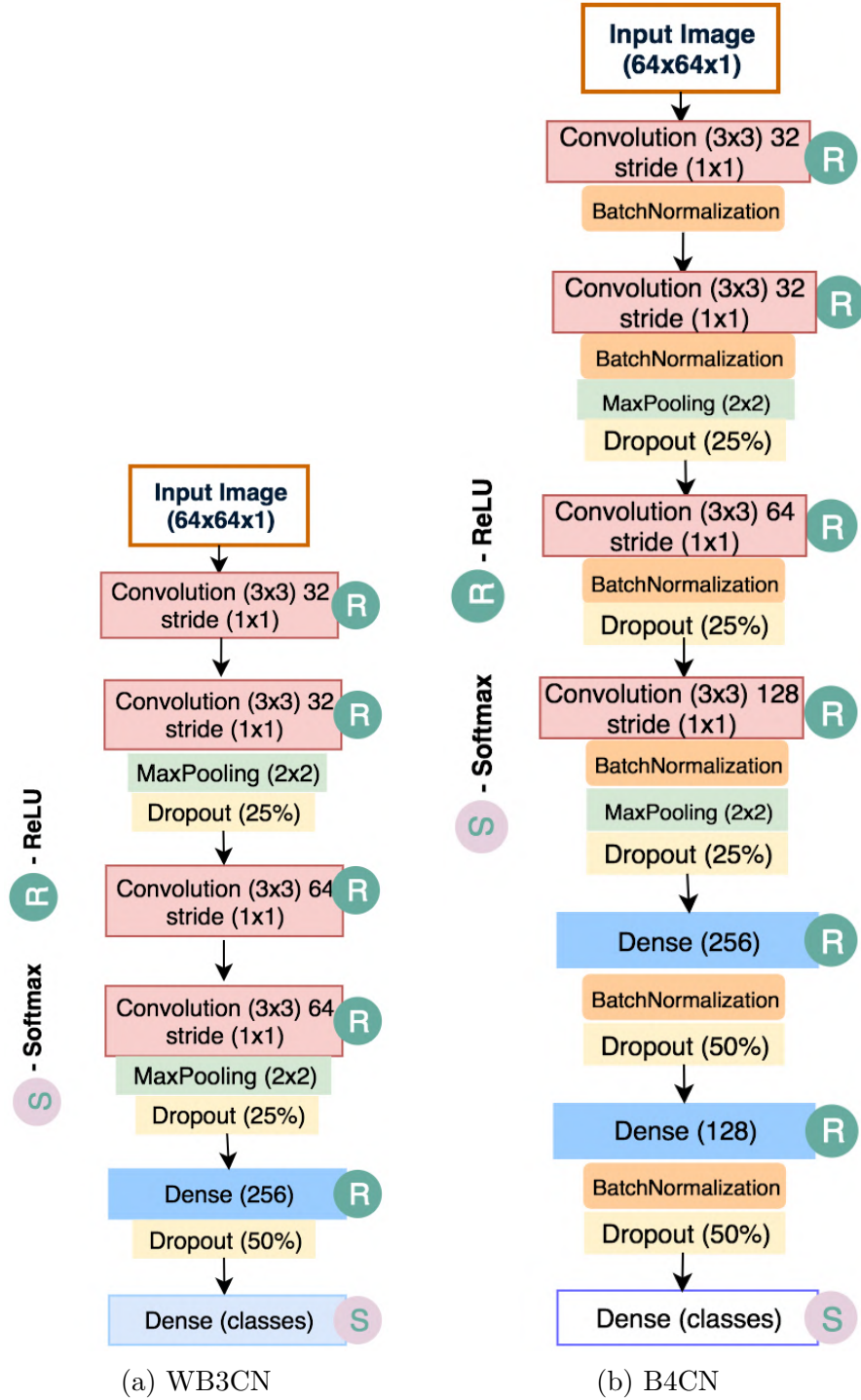


Figure 4.8: Two custom-based recognition model architectures

Our proposed model segments a sign gesture image before processing for recognition. Therefore, we choose simple CNN architecture for the recognition module. The first pair of conv-layers uses 32 filters whereas the second pair uses 64 filters. All the outputs of the convolution layers are passed through a ReLU activation function, except for the last fully-connected layer which uses a SoftMax for final classification.

4.3.3 B4CN

Our next model B4CN (Fig. 4.8b) is different from WB3CN in terms of network layer type and network parameter. It uses four convolution layers, two pooling layers, and three fully-connected layers. Unlike WB3CN we apply batch-normalization after each successive layer. Except for the first and the last layer of the network, we add a dropout later after each convolution and dense layer. A convolution layer is appended with a 25% dropout whereas a dense layers is appended with a 50% dropout. All the outputs of the convolution layers are passed through a ReLU activation function, except for the last fully-connected layer which uses a SoftMax for final classification

4.3.4 MC-SLR

After proposing two custom CNN models as explained in the previous sections, this thesis also explores the scenario where it is necessary to recognize multi-culture signs, instead of a particular ethnic group. We propose MC-SLR model (Fig. 4.9) in this regard. This model operates in two distinct recognition modes such as single-mode recognition and dual-mode recognition. Since the model deals with at least two multi-national, in single-mode a single CNN classifier is applied to determine the sign gesture

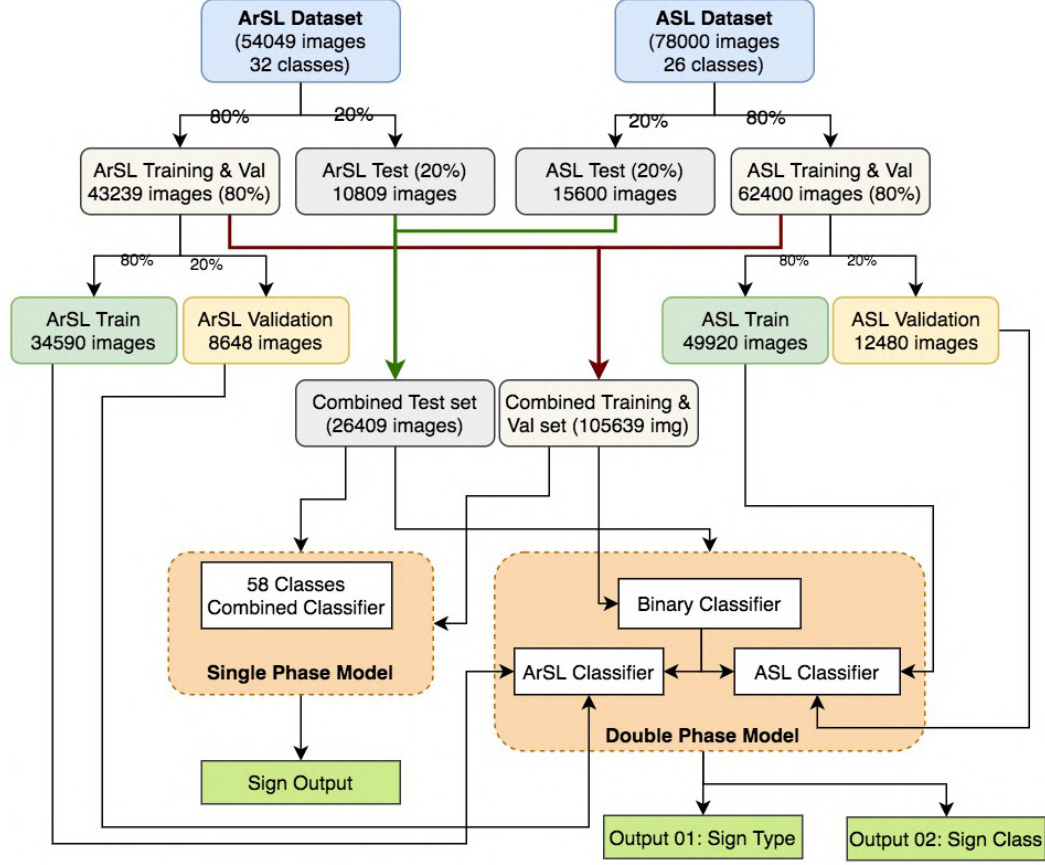


Figure 4.9: MC-SLR system framework

class. On the other hand, in a dual-mode the model performs recognition in two phases. First it determines the language by using a CNN based binary classifier, and next the model recognizes the particular sign class under that particular language. Our choice for selecting a combination of models in the dual-phase model includes the custom-designed WB3CN and B4CN. We also apply two more popular models such as LeNet and AlexNet.

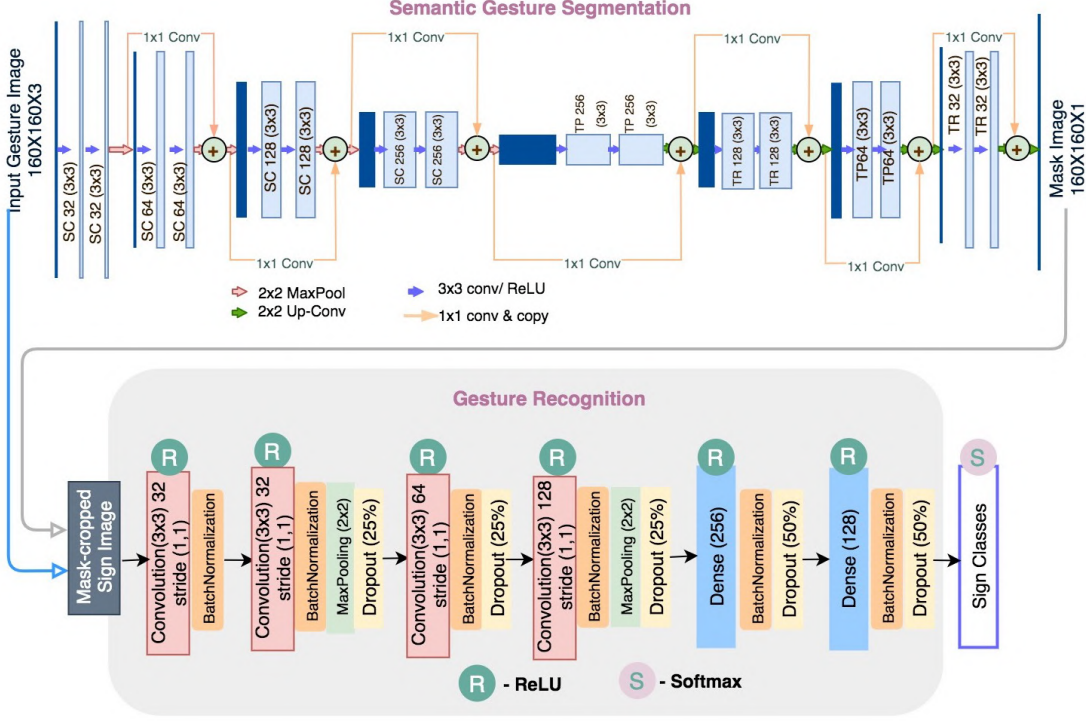


Figure 4.10: S3RNet architecture

4.4 Integrated Model

Finally, we integrated the segmentation and recognition components in two system variants. The first one is called S3RNet as shown in Fig. 4.10. The upper part of this model is the semantic segmentation component, with an encoder-decoder architecture. It is composed of a linear stack of *SeparableConv2D* layers with corresponding skip connections as explained in Section 4.2.5. The recognition module processes the input image and masks to generate the cropped sign which is then passed to the classifier for recognition. We choose WB3CN architecture as our sign classifier. The description of WB3CN is already provided in Section 4.3.2. Nonetheless, the choice for this particular recognition model or for the segmentation block empirical.

As an alternative architecture, we propose another transfer learning-based model

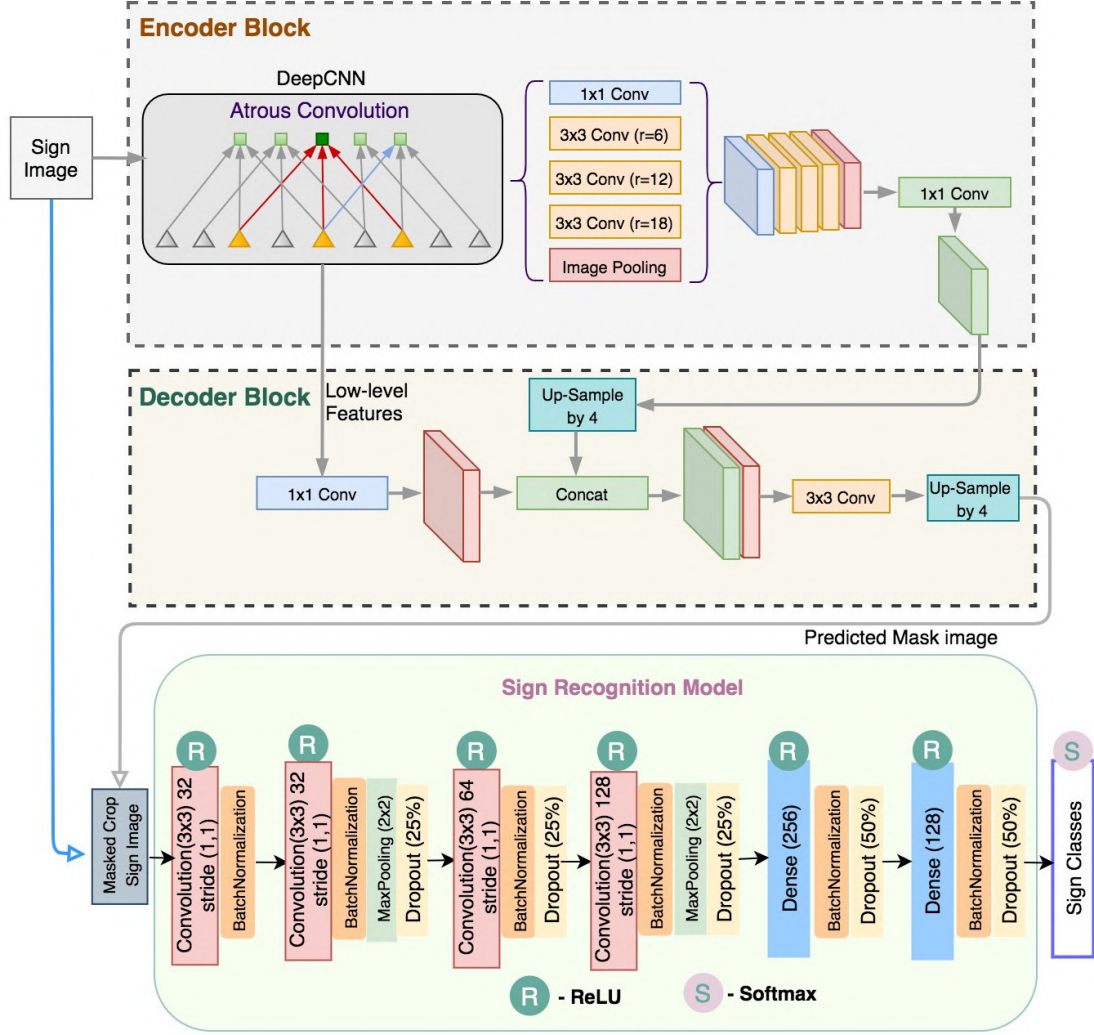


Figure 4.11: Alternate model: DLSR

dubbed as DLSR as shown in Fig. 4.11. Similar to S3RNet, the recognition module of DLSR is also WB3CN. However, the semantic segmentation is based on transfer learning as explained in Section 4.2.4 in order to handle the scarcity of datasets.

4.5 Creation of Ground-truth Dataset

A segmentation model requires ground-truth segmented images along with RGB images. Each segmented class is assigned a unique ID. In the segmentation set, the pixel

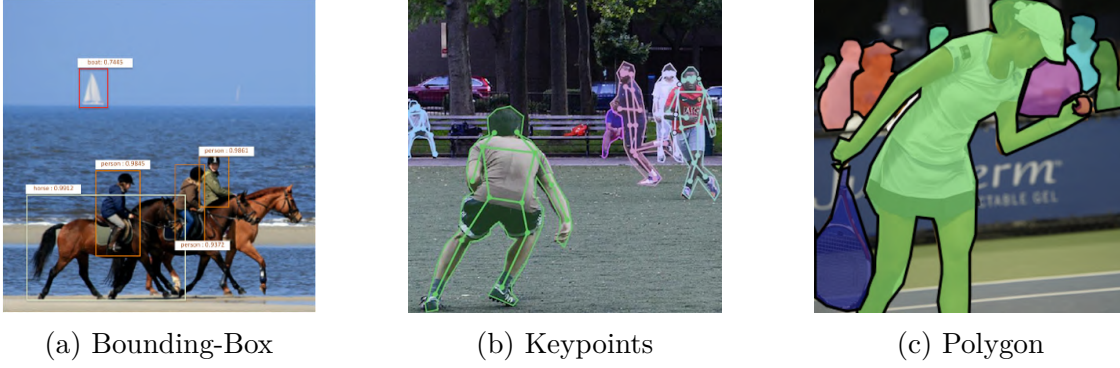


Figure 4.12: Sample images of three annotation methods ¹

value in an image stands for the class ID of the associated pixel. This general format is applicable for most of the segmentation datasets. For segmentation maps, *bmp* or *png* format is more suitable over *jpg*, as the later format is lossy and prone to change its pixel values. Finally, the size of both the segmentation image and corresponding input image must be the same. The filename should be the same for both types of images. Later, a data augmentation technique is applied in order to increase the size of the dataset.

The main tasks for creating an annotated dataset is labeling the input images. There could be many types of image annotation such as bounding-box (Fig. 4.12a), keypoints (Fig. 4.12b), lines and splines, and polygon segmentation (Fig. 4.12c).

A) *Formats of Annotation:*

A variety of image annotation formats are available. Few commonly used annotation formats include Common Objects in Context (COCO), Pascal VOC and YOLO. COCO stores annotations using JSON for all annotation types such as object/keypoint detection, panoptic/stuff segmentation, image captioning and

¹<https://cocodataset.org/>



Figure 4.13: Labeling with *MakeSense* tool

panoptic segmentation. Pascal VOC generates an XML file from the annotations. In YOLO annotation, a text file is created for each image. In our experiments for segmentation, we choose COCO annotation format.

B) *Creating Ground-truths for NUS-HII*

We have used an annotation tool named *MakeSense* for creating COCO JSON of the annotated dataset (Fig. 4.13). The original dataset consists of about 2,000 images of hand sign gestures with very complex backgrounds. The entire images were not annotated at once, rather they were divided into several parts. Later, by using *COCOAssistance* we have merged all the data parts into one single annotated dataset. We have also generated segmented masks, and binary ground-truth images from the annotation files.

4.6 Other Tools and Techniques

A) *Stratified Cross Validation:*

Cross validation plays an important role for preparing data into training and testing parts to be applied in a deep learning model. In cross validation more than one split, a.k.a. folds, are performed. In simple k -fold cross validation, the

dataset is divided into k parts, and each time $k - 1$ folds are used for training while the other fold is used for testing. However, while splitting data into several folds, it would be helpful if each of the folds represents the whole data, that is, in each part the proportion of different groups of data is maintained. Enforcing this type of distribution is termed as stratified k-fold cross-validation.

B) *Data Augmentation:*

Deep learning requires big data for proper recognition. By applying an image data augmentation technique, it is possible to expand the volume of a training dataset. This will also help to increase the model robustness by exposing it to diverse samples of the same sign. Therefore, smaller datasets can be enlarged by using a library such as *ImageDataGenerator* in *keras*. Some parameters for this tool include shift ranges (both width & height), *shear_range*, *zoom_range*, *brightness_range*, and flipping.

C) *Otsu's Binary Thresholding:*

Semantic segmentation model produces predicted mask imagesets where pixel values are not uniform for a particular object class. Therefore, we apply Otsu's thresholding module for separating sign objects from the background. Usually in image segmentation, the selection of unsupervised optimal values for thresholding is challenging. For our case, the values are known for a particular object as the histogram is bimodal distribution. Due to the distinct variation between the sign object and the background, the Otsu's thresholding gives better thresholding results [87]. Since our mask images are bimodal, Otsu's method finds

a value for t as threshold that reduces the weighted intra-class variance as per Equation 4.2. In other words, it basically finds t value that exists within two peaks in the histogram such that variances are minimal for both classes.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad (4.2)$$

where, $\sigma_w^2(t)$ is the intra-class variance. q_1 and q_2 are two probabilities for those two classes separated by t -the threshold value. Also $\sigma_1^2(t)$ and $\sigma_2^2(t)$ are those two class variances.

D) *Batch Normalization:*

Batch normalization is a solution for the vanishing gradient problem in a deep neural network. The vanishing gradient is the scenario where the gradient declines in an exponential manner when the weights are propagated down to the previous layers. As a result, the weights and biases of initial network will not be modified effectively. Usually, those initial network layers are crucial to learning the core parts of the input data, and hence, it may result in poor accuracy. The easiest way to combat this issue is by using batch normalization or using an activation function (e.g. ReLU). Besides, batch normalization allows using higher learning rate and thus the learning process can become faster.

E) *Early Stopping:*

One issue regarding training a neural network is that the choice of training epochs is not an easy task. Overfitting may occur if the number of epochs is set to be very high whereas underfitting may occur otherwise. A technique called Early Stopping is applied by setting an arbitrary larger epochs size, and the

training stops when the performance of the model no longer improves. In other words, early stopping makes the training halt when either the training loss starts to jump up or the validation accuracy tends to drop. The proposed model of this thesis applied early stopping with a reasonable *patience_variable* value against a larger epoch size.

F) *Hyperparameter Tuning with Grid Search:*

Hyperparameters are those special values which are not like network weights and are not learnt directly; they are passed as arguments to a deep learning model. In the grid search technique, the model searches the feasible hyperparameter domain and finds the parameter values that result in the best validation score. This thesis uses *GridSearchCV* library that considers all possible combinations of parameters in an exhaustive search, to find the best combination of *batch_size*, *epoch*, *optimizer*, *learning_rate*, and *momentum*.

CHAPTER 5

EVALUATIONS AND DISCUSSIONS

5.1 Introduction

In this chapter, we carried out a series of experiments with our proposed models. Those experiments can be categorized into two groups such as sign recognition and sign segmentation. Under the first group, we ran experiments using WB3CN, B4CN, TRNetV16, and MC-SLR. Our objective was to evaluate and select the best sign recognition model to be used in the second stage of our proposed S3RNet model. Under the second category, we first ran segmentation experiments and then combined both segmentation and recognition. The results showed that our proposed model can be regarded as an effective solution for static sign gesture segmentation and recognition. The following section gives an overview of the datasets used in our experiments. Next, we describe the experimental setup and performance measures of our experiments. The last section provides experimental results and discussions.

5.2 Datasets

In our experiments, we used four publicly available sign gesture datasets: one dataset was on Arabic Sign Language (ArSL), two datasets on American Sign Language (ASL), and the last one on general sign gestures. An annotation dataset was created from NUS-HII for sign gesture segmentation. Several experiments were conducted with the four datasets along with this segmented one. Table 5.1 shows a summary of the datasets and the details are described in the following subsections.

Table 5.1: Summary of datasets

Dataset	#Signs	#Signers	#Images/Sign	#Total	Cited In
ArASL-PMU	32	40	1,293 - 2,114	54,049	[88], [89], [90], [91], [92]
Massey-G12	36	5	65 - 70	2,515	[22], [72], [93], [94], [95]
GRASSK-ASL	29	Not given	3,000	87,000	[66], [96], [97], [65], [98]
NUS-HII	10	40	200	2,000	[99], [100], [101]

5.2.1 ArASL-PMU Dataset

The first dataset, ArASL-PMU [89], consists of 54,049 labeled images of 32 signs of Arabic alphabets where 40 participants of different age groups contributed. A sample of images are shown in Fig. 5.1. Each of the 32 classes does not have the same number of images. The authors of the dataset applied image augmentation to increase the dataset size. The dataset also has some issues such as intensity variations, dark-background/foreground and slightly complex background. The collected sign language images are resized to a dimension of 64×64 and finally turned into gray-scale images.



Figure 5.1: Sample sign gestures of ArASL-PMU dataset

5.2.2 Massey-G12 Dataset

This ASL dataset was created by Massey University and the dataset has a total of 2,515 images of five subjects; not each subject contributed equal number of gesture images (subject 1: 900, subject 2: 895, subject 3: 180, subject 4: 180, and subject 5: 360). Images in the dataset include different illumination conditions and a large variety of hands; a sample is shown in Fig. 5.2. They were cropped at their segmented

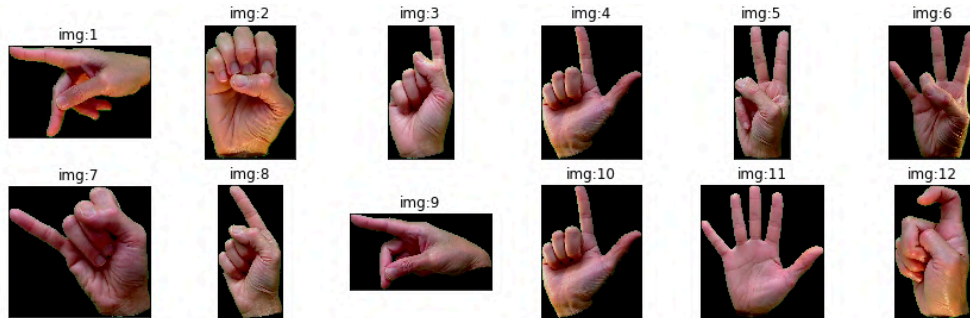


Figure 5.2: Sample sign gestures of Massey-G12 dataset

bounding-boxes in different sizes. Almost all the gestures have dark backgrounds and the hands cannot be seen beyond wrist level. The images do not show a large angle of rotation except within a limited range. The dataset was augmented with rotation, zoom, and crop.

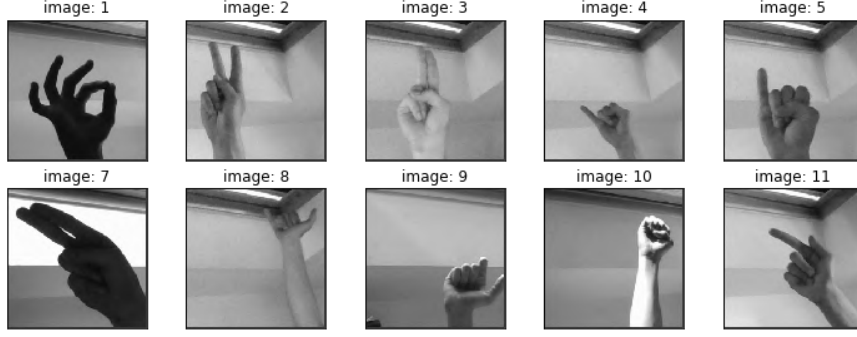


Figure 5.3: Sample sign gestures of GRASSK-ASL dataset

5.2.3 GRASSK-ASL dataset

This dataset is based on ASL alphabets and is developed by [66]. It has a total of 87,000 images of 26 classes with an equal number of images for each class (i.e. a balanced dataset). The images in the dataset include different illumination conditions and a large variety of hands; a sample is shown in Fig. 5.3. They have both dark and light background with different view points and limited angle of rotation where hands are not seen beyond the elbow level.

5.2.4 NUS-HII Dataset

This dataset is offered by the National University of Singapore. It is the second version of its type having 10 classes of hand gestures, which are captured in and around the NUS campus, with complex natural backgrounds and various hand sizes and shapes. This makes it very challenging for both segmentation and classification. A sample of images is shown in Fig. 5.4. The variation in illumination, background objects, skin-color, and hand size/shape are so evident that even the human eyes may fail to recognize the sign gestures correctly. The NUS-HII signs were performed by forty

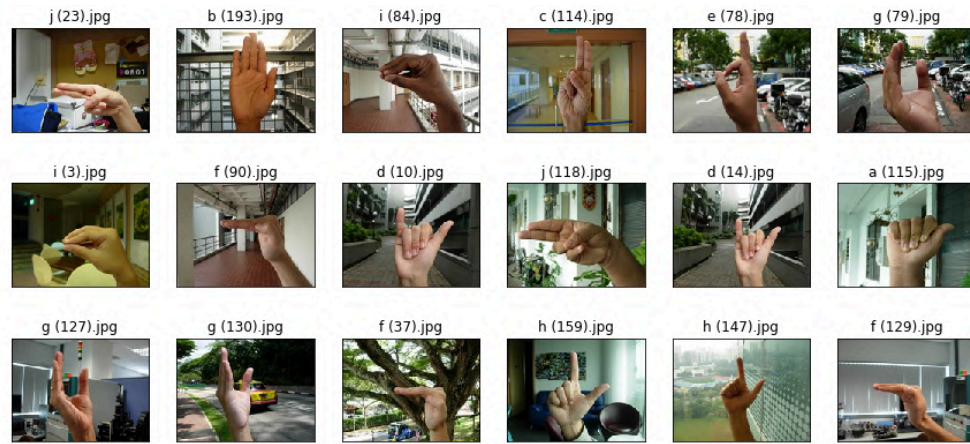


Figure 5.4: Sample sign gestures of NUS-HII dataset

subjects having various ethnicity, genders and age range from 22 to 56 years. For each subject there are 10 hand postures with five variations which are represented with letters a to j accordingly. The dataset contains a total of 2,000 sign gesture images in RGB format.

5.3 Experimental Setup and Performance Measures

The hardware used in all experiments is a MacbookPro computer (Intel Core(TM) i7-7700 3.6 GHz, 16.0 GB RAM). The convolutional neural network models were implemented with Python version 3.6 and *keras/Tensorflow*. For all of the experiments, about 20% of each dataset was reserved for testing, the remaining 80% samples were used for training and validation. The training-validation split ratio was 80% for training and 20% for validation. We resized images by applying linear interpolation from the *OpenCV* library. In recognition experiments, all images were resized into 64×64 grayscale images. The reason behind choosing smaller resolution is due to the fact that, for the sign gesture recognition in this thesis, a bigger network with

a large amount of image features will not improve performance [102], [103]. Rather, our fine-grained model is capable of performing better with low resolution recognition of segmented gestures. For semantic segmentation experiments, we resize the sign images into 160×120 color images.

We applied early-stopping during training with suitable *patience* in almost all the experiments. We chose *validation_loss* as the criteria for halting the training and saving weights of the best performing model. All the training samples were normalized before feeding into the model, and the target classes were fed as *onehot* vector. For each testing phase, we restarted the kernel and loaded the saved model. Next, we fed the test data into the loaded model and measure the performance in two decimal places. For the recognition experiment, we recorded the loss and accuracy of each training and validation at each epoch. We set epoch size considerably high enough so that the training does not halt except due to early stopping criterion. For each recognition experiment, we also measured the average precision, recall, F1-score and accuracy of each class. For the experiments of semantic segmentation, we measured the mean intersection of union (*mIOU*) score between the predicted and ground truth masks.

5.4 Recognition Models on Imagesets

This section describes experiments with the three recognition models using both raw and masked datasets. Detailed results for each model are given in Appendix A including confusion matrices and performance reports.

We ran a series of recognition experiments on the four datasets: ArASL-PMU, Massey-G12, GRASSK-ASL, and NUS-HII. For each individual dataset, we applied our three proposed recognition models, which are TRNetV16, WB3CN, and B4CN. In this particular experiment section, our objective was to make relative comparisons among the models based on their performances with respect to each dataset. We keep in mind that we give credit to a recognition model if it performs better classification for the gesture images with relatively less complex backgrounds. This is because the recognition model will deal with sign images after being processed by our segmentation block. The comprehensive experimental results of the recognition models against those datasets are recorded in Table 5.2.

For Massey-G12 dataset, all the three proposed recognition models showed promising performances. The most performing model for this dataset was WB3CN with a recognition accuracy of 99.67%. For the remaining two models, they recorded recognition accuracies of 99.34% and 99.60% respectively. These recognition performances of are better than what was achieved by Rastgoo et al. [22] and Chevtchenko et al. [72], which had 99.31% and 98.61%, respectively. Figure 5.5 shows the accuracy and loss curves for experiments with Massey-G12 dataset. All the models show smooth convergence during training. The validation curves show similar trends except for B4CN model. The other performance scores can be found in Appendix A (A.36, A.37, A.38).

For the ArASL-PMU dataset, the accuracies obtained are 96.62%, 95.77%, and 96.48% for TRNetV16, WB3CN and B4CN, respectively. In the literature, we found Latif et al. [88] achieved a slightly better accuracy of 97.6% on the same dataset. Figure 5.6 integrates and normalizes the accuracy and loss curves of the three models

on ArASL-PMU dataset for training and validation. The graphs suggest that all the three models can provide decent accuracy gain and gradual decrease in loss during the training. However, we observed irregular behavior in validation curves for B4CN model (Fig. 5.6d and 5.6c). The other performance scores obtained in detailed can be found in Appendix A (A.39, A.40, A.41).

For the GRASSK-ASL dataset, the three models performed very well with superior accuracy above 99%, which is better than other datasets. The best model was B4CN achieving 99.93% accuracy. The accuracy and loss curves during training and validation can be found in Fig. 5.7. The curves also reveals that the WB3CN model runs for longer epochs and has better validation performance. The other performance scores can be found in Appendix A (A.42, A.43, A.44).

Moreover, we applied the proposed models for the NUS-HII dataset that contains sign gesture images with complex backgrounds. Table 5.2 shows that the three models performed relatively poorly for this particular dataset compared to the other datasets. The complex nature of the backgrounds makes it harder for recognition. However, the performances of the three models are better than Serre et al. [99] which obtained an accuracy of 75.71% for this dataset. Confusion matrices and other performance scores can be found in Appendix A (A.45, A.46, A.47, A.48, A.49). Figure 5.8 shows the model behavior during training, where B4CN is converging faster.

Finally, we tested the proposed recognition models on a cropped version of the NUS-HII dataset using the ground truth masks. Figure 5.9 shows a sample of the masked sign gesture images of this dataset. The reason of selecting this dataset is due to the impact of the challenging background on the model performance. The

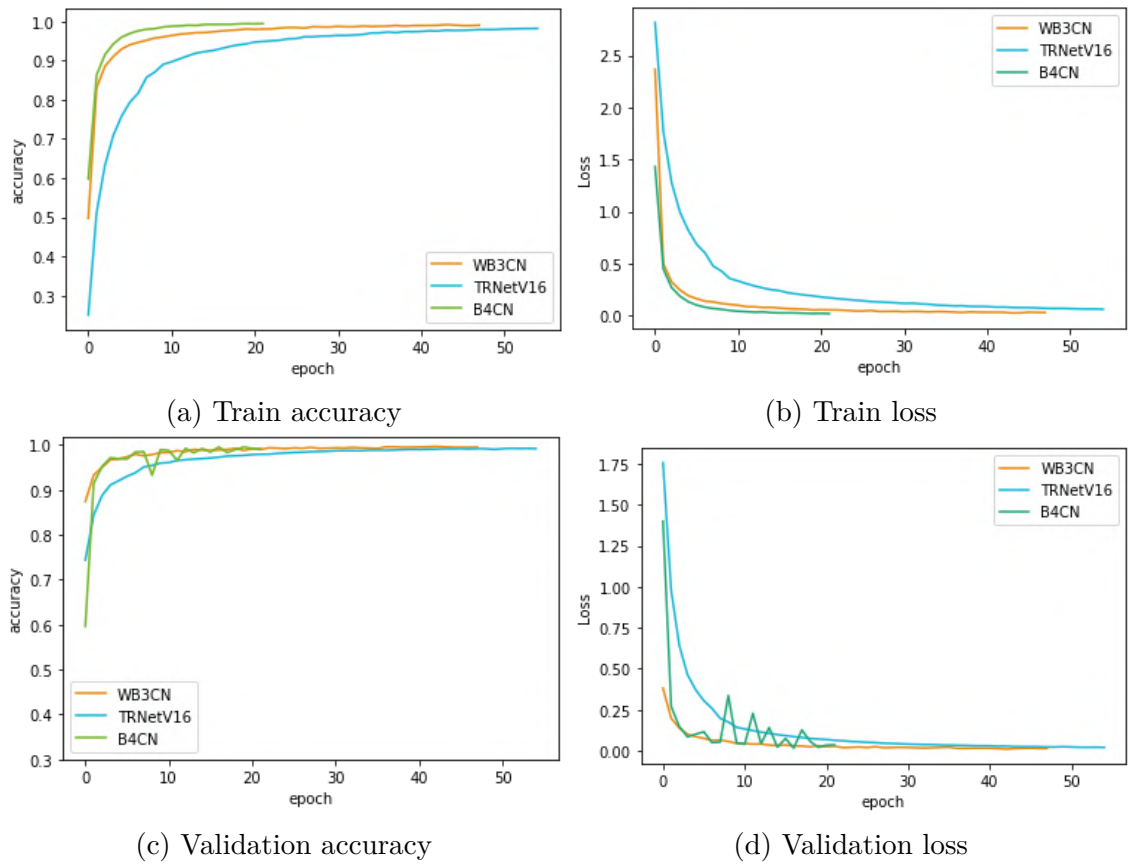


Figure 5.5: Learning curves for different models on the Massey-G12 dataset

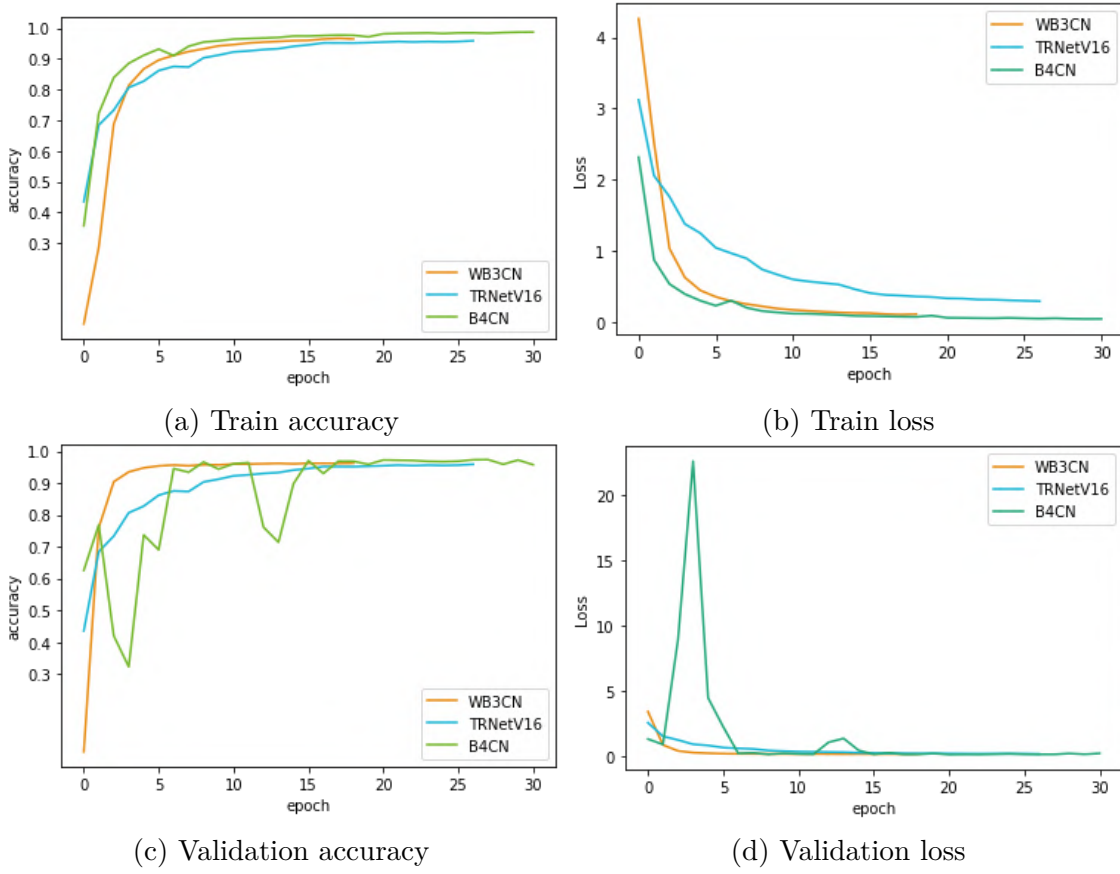


Figure 5.6: Learning curves for different models on the ArASL-PMU dataset

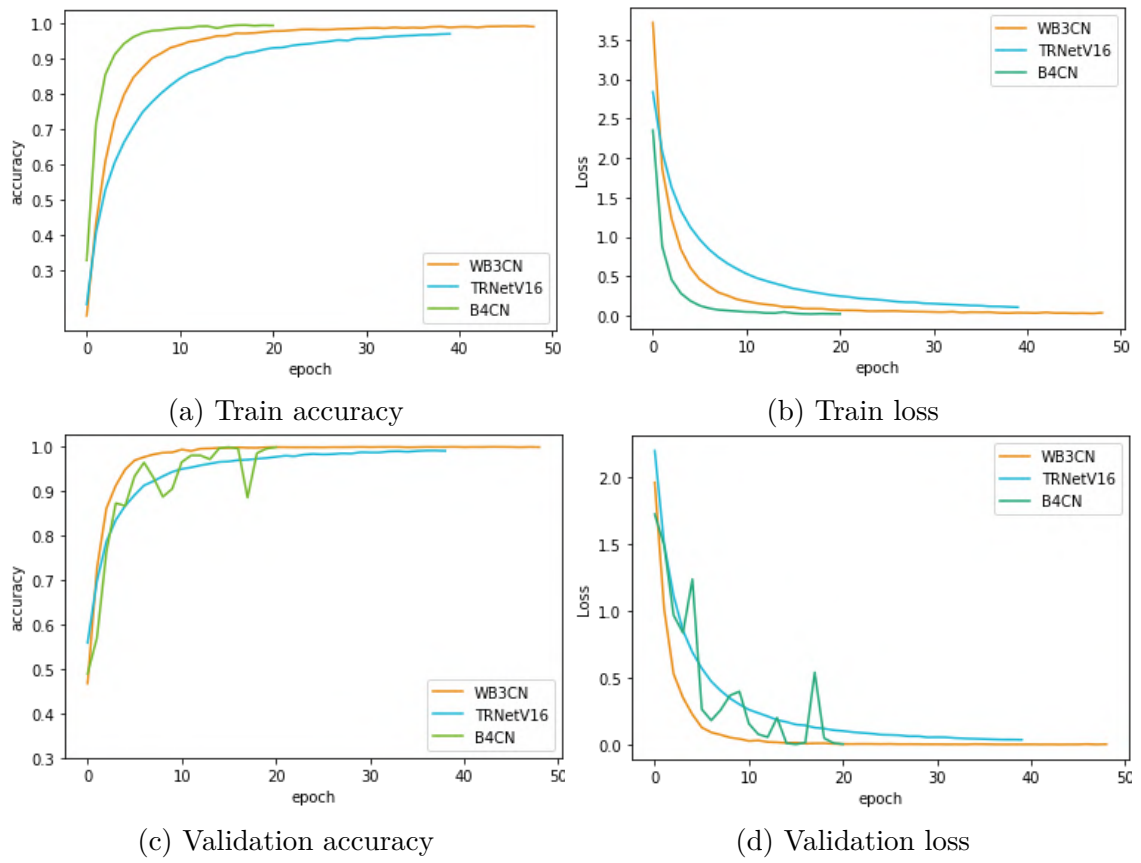


Figure 5.7: Learning curves for different models on the GRASSK-ASL dataset

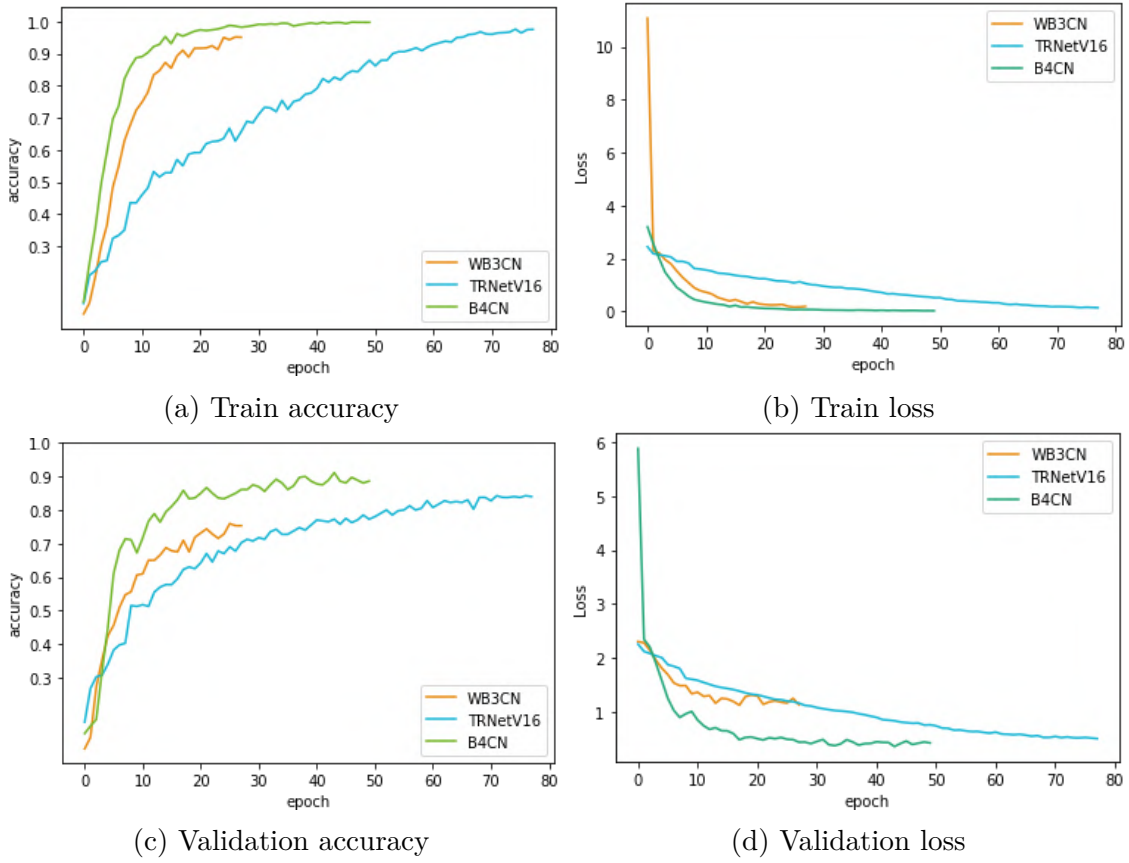


Figure 5.8: Learning curves for different models on the NUS-HII dataset

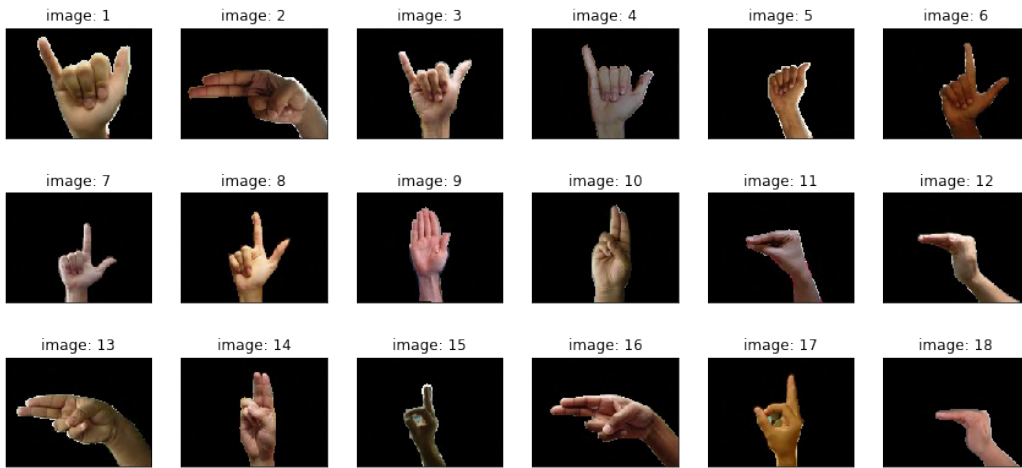


Figure 5.9: A sample of NUS-HII images cropped with ground truth masks

Table 5.2: The percentage performance of SLR for different CNN models on different datasets (approx. two decimal places)

Dataset	Model	Prc	Rec	F1	Acc
Massey-G12	TRNetV16	99.33	99.26	99.29	99.34
	WB3CN	99.73	99.41	99.55	99.67
	B4CN	99.81	99.40	99.59	99.60
ArASL-PMU	TRNetV16	98.04	95.99	97.00	96.62
	WB3CN	98.12	95.77	96.92	95.77
	B4CN	97.56	96.48	96.98	96.48
GRASSK-ASL	TRNetV16	99.47	99.01	99.24	99.25
	WB3CN	99.99	99.98	99.98	99.98
	B4CN	99.99	99.99	99.98	99.93
NUS-HII	TRNetV16	87.86	68.50	76.60	80.01
	WB3CN	80.34	69.59	74.27	75.99
	B4CN	93.05	87.00	89.25	89.49
NUS-HII*	TRNetV16	97.06	96.75	96.88	97.05
	WB3CN	99.01	98.00	98.48	98.50
	B4CN	89.57	69.75	76.95	70.18

NUS-HII* is cropped with ground truth masks

results of our recognition models for the masked imageset are shown in Table 5.2 with significant improvement in the accuracy for TRNetV16 and WB3CN. However, the performance of B4CN degraded due to the incorporation of batch normalization. For the cropped NUS-HII dataset, confusion matrices and the other performance scores obtained in detailed can be found in Appendix A (A.50, A.51, A.52, A.53).

Based on the previous experiments, we will select WB3CN and discard the other models for further experiments on segmentation and recognition. The reason of discarding TRNetV16 is due to its complexity. B4CN was also discarded because it has significantly low performance for masked images compared to WB3CN.

5.5 Combining Segmentation with Recognition

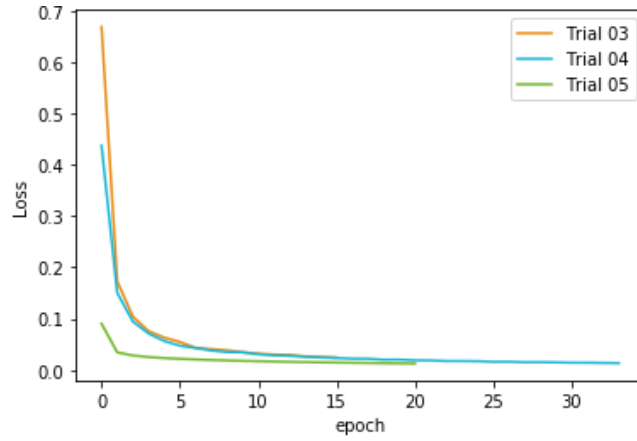
This section initially describes several segmentation experiments for model tuning with NUS-HII dataset and the ground-truth masks. Next, a list of sensitivity analysis experiments with different augmentation parameters are described and analyzed. The last part of this section demonstrates the combined segmentation and recognition experiments with different models. Detailed results for each experiment are given in Appendix A.

In the segmentation model tuning phase, we ran five experiments with different model parameters on the NUS-HII dataset. Our objective was to determine the best performing (higher $mIOU$ score) segmentation component to be integrated with the two proposed models: S3RNet and DLSR. Four of the experiments used the dataset without augmentation and the last one applied augmented dataset. The experimental results for each trial are recorded in Table 5.3. As per the results, the fourth model variant showed an $mIOU$ score of 93.24%. This segmentation score was the highest before applying data augmentation. We applied this particular segmentation model in the next phase of experiments. The last experiment in this round (Trial 05) applied the augmented dataset and showed an $mIOU$ score of 95.17%. The loss curves during training and validation for the last three experiments are shown in Fig. 5.10.

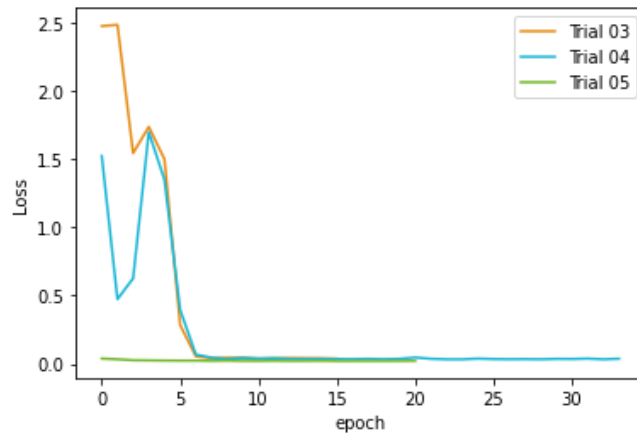
In the sensitivity analysis phase, a series of experiments were carried out by tuning each individual augmentation argument in *ImageDataGenerator* of *keras*. For each round, we measured the segmentation score in terms of $mIOU$ and the recognition

Table 5.3: The percentage performance of segmentation model while tuning training parameters (approx. two decimal places)

Trials	Aug.?	Variance	Patience	Training Parameters	<i>mIOU</i> (%)
01	NO	Symmetric	5	416 (26×16)	92.29
02	NO	Asymetric	3	7 (7×1)	72.58
03	NO	Symmetric	5	512 (16×32)	91.15
04	NO	Symmetric	10	544 (34×16)	93.24
05	YES	Symmetric	5	336 (21×16)	95.17



(a) Training loss



(b) Validation loss

Figure 5.10: Segmentation model(S3RNet) tuning trials' results

accuracy in two decimal places. The comprehensive experimental results of S3RNet model for those parameters are recorded in Table 5.4. DLSR has inbuilt DeepLabV3+ as the segmentation component and the segmenter is fine-tuned by its developer, so we did not run experiments to observe its sensitivity with the augmentation arguments.

In the first sensitivity analysis experiment, *shear_range* was set to $[0 - 45.0]^\circ$ and the model S3RNet with augmented dataset showed an *mIOU* score of 91.97% and a recognition accuracy of 88.17%. The recognition suffered more compared to segmentation in this particular experiment. We argue that the huge *shear_range* value was the reason behind this low recognition rate. A *shear_range* of $[0 - 45.0]^\circ$ deformed the gestures shapes and regions, so the recognition model found it hard recognizing many samples. The next two experiments in this round with two other *shear_range* values: $[0 - 20.0]^\circ$ and $[0 - 30.0]^\circ$ suggest that there is a negative linear relation between recognition accuracy and *shear_range*. Figure 5.11 shows some good and bad segmentation performance of S3RNet for samples of sign gesture images with a *shear_range* of $[0 - 45.0]^\circ$. Detailed performance scores, confusion matrices and samples of segmentation performances can be found in Appendix A (A.12, A.13, A.15, A.16, A.17).

In our next experiment, we used *width_shift_range* = 0.5, which means we let an image shift horizontally at most 50% to the right or to the left. This caused a significant part of gesture images to be omitted from right or the left. As a result, the segmentation accuracy was somewhat decent but the recognition accuracy suffered a big setback (55.60%). Another observation was that when the region of interest became smaller due to shift in object towards one side, the *mIOU* score also suffered slightly. Figure 5.13 shows some good and bad segmentation perfor-

mance of S3RNet for samples of sign gesture images with a *width_shift* value of 0.5. The other performance scores and confusion matrices can be found in Appendix A (A.8, A.9, A.10, A.11).

Based on the outcome of the four experiments describe above, we argue that in the final round of our data augmentation process, we should set lower values for arguments which cause sever deformation (e.g *shear_range*) or removal of major part of the sign hand (e.g. *width_shift_range*).

In the following experiment, we applied *brightness_range* of $[0.2, 1.8]$, which means the brightness of an augmented image can vary from 80% darker to at most 80% brighter than the original image. The outcome of the experiment further revealed that *brightness_range* has no significant impact on the segmentation or the recognition performance, as they are almost the same as before. Similar outcome can be observed for an input range of $[0.1, 1.9]$. Effect of *brightness_range* for some random images and their corresponding predictions can be found in Appendix A(A.1).

On the other hand, flipping along both axis showed the lowest scores for both segmentation and recognition.

Experiments with *rotation_range* also showed similar negative linear relation with respect to recognition accuracy. Segmentation performance for this particular parameter showed somewhat stable *mIoU* scores. Figure ?? shows some good and bad segmentation performance of the model for samples of sign gesture images with a *rotation_range* of $[0 - 25.0]^\circ$. The detailed performance scores and confusion matrices of these experiments are recorded in Appendix A (A.2, A.3, A.4, A.5, A.6, A.7).

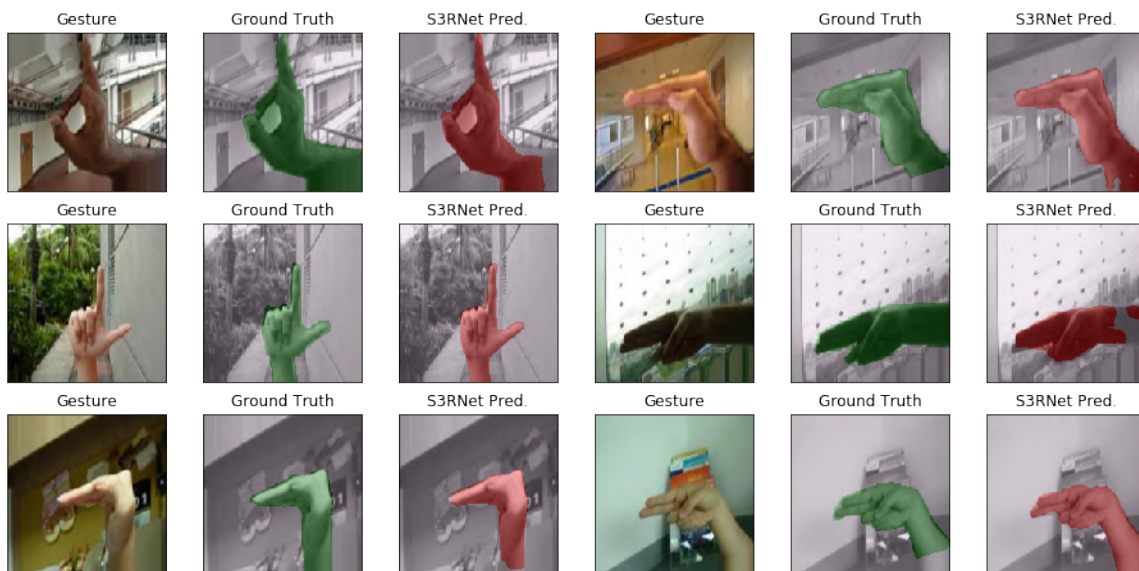
All the partial augmentation experiments suggest that the combined data aug-

Table 5.4: The percentage performance of sensitivity analysis of S3RNet for augmentation parameters (approx. two decimal places)

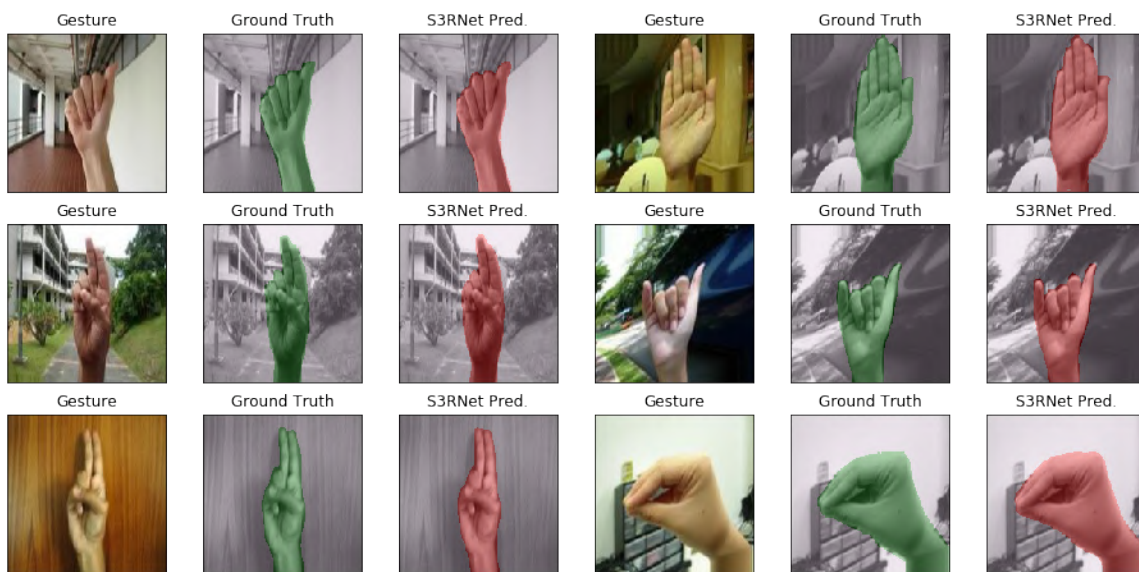
Argument		<i>mIOU</i> (%)	Prc (%)	Rec (%)	F1 (%)	Acc (%)
Name	Value					
Shear Range	[0 - 45.0]°	91.97	90.65	88.08	88.39	88.17
	[0 - 30.0]°	94.18	97.02	96.85	96.86	96.85
	[0 - 20.0]°	94.81	99.19	99.18	99.17	99.17
Width Shift	0.50	88.37	68.04	54.53	58.32	55.60
	0.3	94.76	97.43	97.00	97.15	97.12
Flip (H/V)	H	85.73	63.89	63.90	63.44	63.89
	V	77.13	59.65	55.50	56.23	55.80
Brightness Range	[0.2, 1.8]	94.93	99.60	99.60	99.60	99.59
	[0.1, 1.9]	94.68	99.48	99.48	99.48	99.47
Rotation	[0 - 30.0]°	93.78	91.66	90.28	90.60	90.54
	[0 - 25.0]°	94.32	96.02	95.69	95.70	95.67
	[0 - 20.0]°	94.69	98.72	98.55	98.62	98.57

mentation should have arguments where width shift should be much less than 0.5, shear range be around $[0 - 25.0]^\circ$ and rotation be less than 30° . Data augmentation were performed with the arguments: $shear_range = [0 - 20.0]^\circ$, $rotation_range = [0 - 30.0]^\circ$, $width_shift_range = 0.2$, $brightness_range = [0.2, 1.8]$, and $height_shift_range = 0.1$.

The performance of S3RNet and DLSR with the final augmented dataset are shown in Table 5.5. This table also shows the results of other models. The best performance for the augmented dataset was shown by our proposed S3RNet model with an *mIOU* value of 95.17% and a recognition accuracy of 99.74%. Performance of S3RNet on the normal dataset showed a segmentation score of 92.29% (*mIOU*) and the corresponding recognition accuracy was about 98.61%. When the model was trained on the augmented dataset, it became more robust with a diverge range of training samples and showed better performance for both segmentation and recognition.

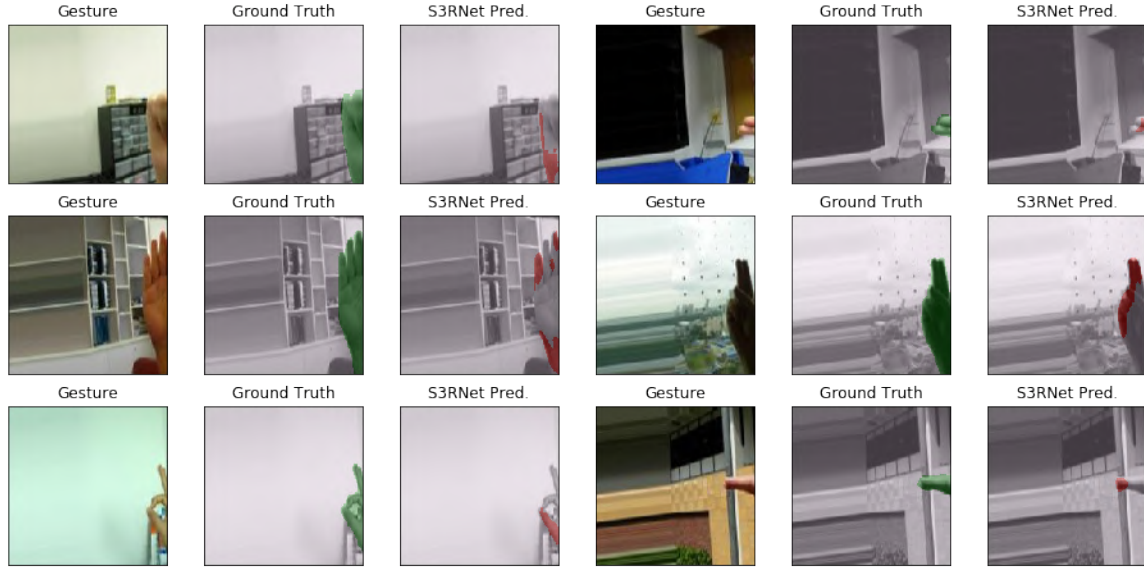


(a) Bad segmentation prediction ($mIOU = 77.62\%$)

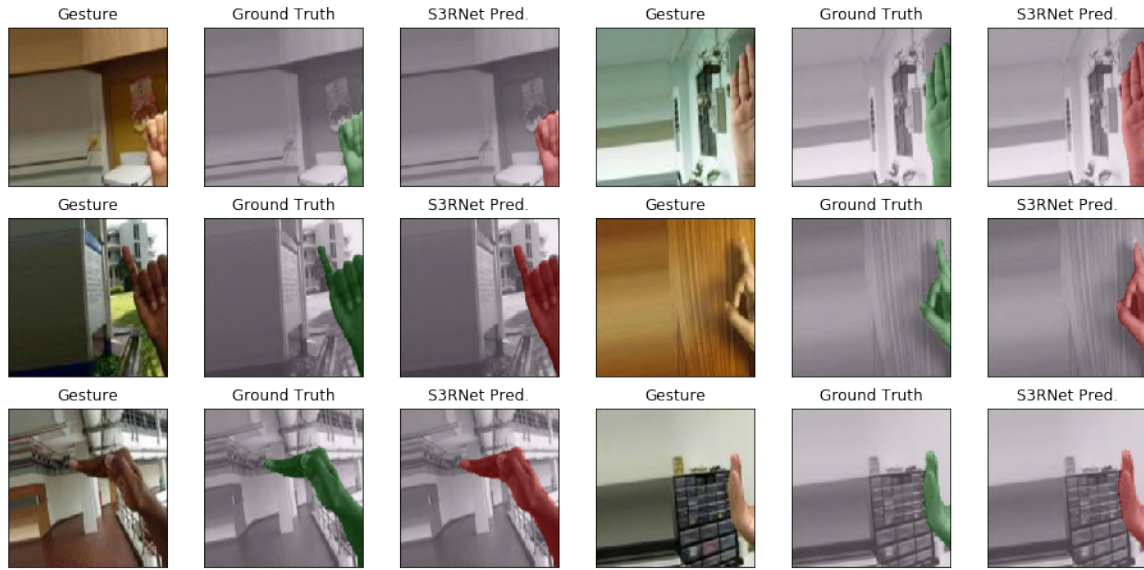


(b) Good segmentation prediction ($mIOU = 95.37\%$)

Figure 5.11: Sensitivity analysis for shear range $[0 - 45.0]^\circ$

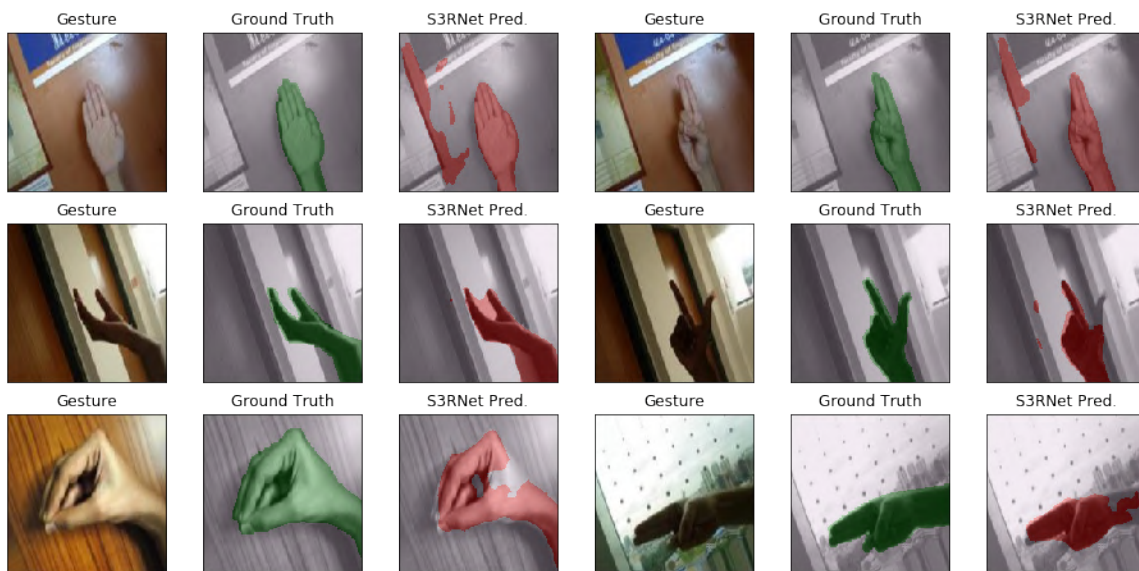


(a) Bad segmentation prediction ($mIOU = 25.37\%$)



(b) Good segmentation prediction ($mIOU = 93.23\%$)

Figure 5.12: Sensitivity analysis for width shift $[0.5]$



(a) Bad segmentation prediction ($mIOU = 67.67\%$)



(b) Good segmentation prediction ($mIOU = 95.4\%$)

Figure 5.13: Sensitivity analysis for rotation range $[0 - 25.0]^\circ$

Table 5.5: The percentage performance of segmentation and recognition for different CNN models (approx. two decimal places)

Model	Aug?	Size (MB)	<i>mIOU</i> (%)	Prc (%)	Rec (%)	F1 (%)	Acc (%)
UNet [5]	NO	1.9	60.47	72.76	60.00	63.30	62.5
FCN32 [4]	NO	537.2	88.04	84.33	78.50	77.96	80.00
FCN16 [4]	NO	537.2	91.64	96.67	96.00	96.24	96.49
MobileNet-V2 [104]	NO	23	79.41	92.74	87.79	89.95	88.34
DLSR	YES	439	80.45	94.90	91.51	93.09	92.01
	NO	439	84.63	96.92	94.27	95.58	94.47
S3RNet	YES	8.4	95.17	99.78	99.70	99.74	99.72
	NO	8.4	92.29	98.61	98.50	98.51	98.50

We compared our results with some contemporary models: UNet [5], FCN [4], and ModileNet-V2 [104]. We ran two variants of FCN model. FCN16 showed comparatively better results for both segmentation and recognition. The detailed performance scores and confusion matrices of these experiments are shown in Appendix A (A.18, A.19, A.20, A.21, A.22, A.23, A.24, A.25, A.26, A.27, A.28, A.29, A.30, A.31).

5.6 Cross-Language Recognition

This section describes cross-language experiments using two national sign alphabet datasets. For these two datasets, we applied our two proposed models (WB3CN & B4CN) and two other popular models: AlexNet [41] and LeNet [42]. For each model, the network architecture were designed according to Fig. 4.9 of Chapter 4. The comprehensive experimental results of the recognition models in single phase and dual phase are shown in Table 5.6.

In single phase, except for AlexNet, all the remaining models showed lower performance compared to dual phase. AlexNet showed an accuracy of 94.09%. In single

phase, accuracy obtained are 96.43%, 98.79% and 86.21% for LeNet, WB3CN and B4CN, respectively. Of all four models, WB3CN performed better whereas B4CN showed the lowest performance.

In dual phase, all the models showed good recognition scores. However, AlexNet accuracy dropped slightly from 94.09% in single phase to 93.32% in dual phase. Accuracy obtained for LeNet, WB3CN and B4CN in this phase are 96.51%, 97.98% and 98.15%, respectively. The accuracy and loss curves for single phase models can be found in Fig. 5.15. The curves for dual phase models are shown in Fig. 5.16 and Fig. 5.14

For the datasets, LeNet outperforms AlexNet in terms of accuracy in both of the phases. Out of the four models, LeNet has lower model size and training time. As per the training and validation performance curves, the gaps between training and validation accuracy/loss curves tend to widen for larger epoch. However, due to early stopping callback, the executions do not go far beyond the *patience* value for respective executions and the best weights are restored. For the two datasets, model performance decreases for deeper CNN model. Out of the four models, AlexNet has deeper architecture and it shows poor performance in both phases. WB3CN outperforms all the three models in both phases.

The model B4CN shows rocky changes in validation curves, though for the training curves it shows much stable and quick convergence. The reason is it used batch-normalization at each layer, which might result in poor performance in the early validation phases, as the biases and weights of initial network did not get modified effectively. This model stabilizes gradually as the training epoch increases. Another observation

Table 5.6: The percentage performance for different CNN models on cross-sign datasets for two phases (S/D) (approx. two decimal places)

Model	Phase	Params ($\times 10^6$)	Time (s)/epoch	Prc	Rec	F1	Acc
AlexNet	S	57.38	13,841	94.69	93.91	94.24	94.09
	D	114.2	14,039	95.42	92.73	94.01	93.32
LeNet	S	1.641	283	96.78	96.32	96.51	96.43
	D	4.915	553	97.57	96.23	96.88	96.51
WB3CN	S	2.849	1,161	99.04	99.69	98.86	98.79
	D	8.12	1,160	98.99	97.76	98.36	97.98
B4CN	S	5.682	2,012	92.32	85.52	87.21	86.21
	D	17.03	2,244	98.70	98.15	98.41	98.15

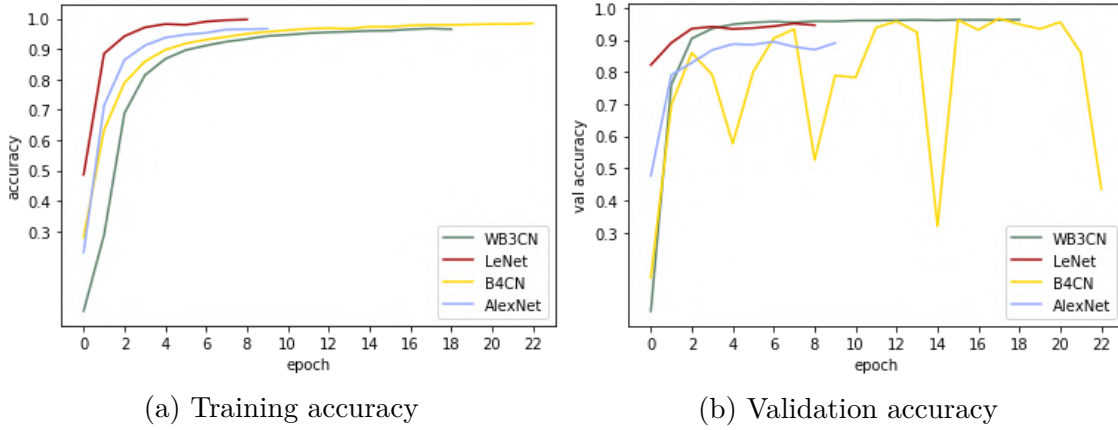


Figure 5.14: ArASL-PMU: 2P model results

is, the model B4CN uses a dropout ratio of 50% in two consecutive dense layers, which could adversely affects the performance. Finally, the model WB3CN shows descent performance during training and validation compared to B4CN. Other performance scores and confusion matrices can be found in Appendix A (A.32, A.33, A.34, A.35).

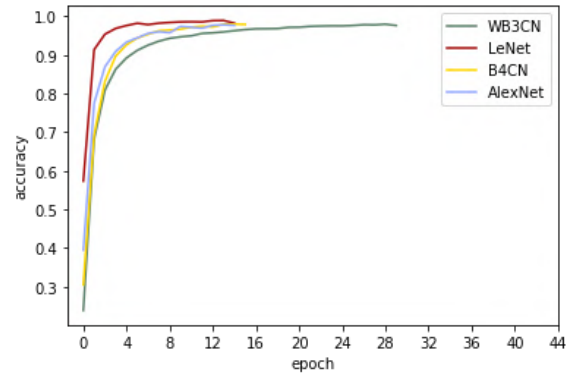
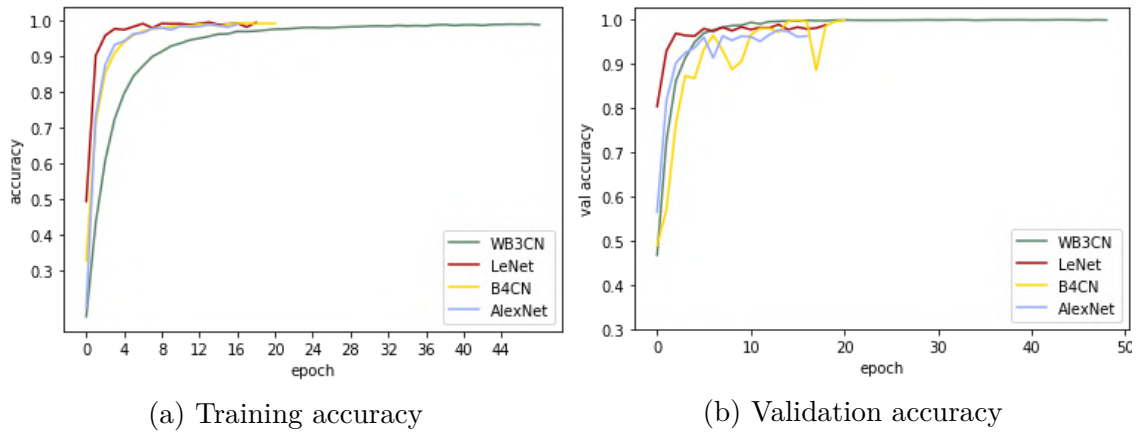


Figure 5.15: Training accuracy for single phase models



(a) Training accuracy

(b) Validation accuracy

Figure 5.16: GRASSK-ASL: 2P results

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

This chapter begins with a summary and conclusion of our thesis work on hand gesture and sign language segmentation and recognition. Then we briefly outline some future research directions and goals.

HCI is gaining more traction in research. As a sub-category, sign gesture recognition also begs more works in the field. The goal is to make the communication barrier as minimal as possible between normal hearing and speech-impaired people. Deep learning based recognition solutions can serve the purpose to deal with some challenges for better recognition performance including complex background, variation in illumination and blur. A good segmentation technique can combat these issues. Previously, researchers in this field applied several traditional techniques for sign gesture segmentation. However, the challenges could not be solved completely with traditional tools. This thesis explored and compared various solutions and pro-

posed a novel model dubbed as "S3RNet" which is suited for static gesture segmentation and recognition. The model integrates a semantic segmentation block and a custom CNN-based classifier for static sign image recognition block. The semantic segmentation module has been designed following the encoder-decoder architecture with residual connections among the corresponding convolution layers and motivated by depthwise separable convolution. We have fine tuned our model in a series of experiments and observations. The recognition block of our model has also been fine-tuned and chosen based on empirical outcomes. For that, we have designed three distinct sign-classifiers including two custom CNN-based models (WB3CN and B4CN) and one transfer learning-based model (TRNetV16) which is pre-trained with ImageNet dataset. Four publicly available datasets: ArASL-PMU, Massey-G12, GRASSK-ASL and NUS-HII have been used to evaluate these models. Based on the results for these datasets and the three recognition models, the most suitable classifier was chosen for the integrated S3RNet model. Alternatively, a second integrated model (DLSR) has also been offered in this thesis. DLSR is quite identical to S3RNet model except that it applies a transfer learning-based semantic segmenter.

In order to evaluate the segmentation engine, a custom-developed ground-truth dataset has been developed from annotating NUS-HII dataset. To make the models work more robust, data augmented has been applied with this dataset in a series of experiments on a trial and error basis. We have fine tuned the arguments value for data augmentation by observing the impacts on the segmentation and recognition performances altogether. These sensitivity-analysis experiments demonstrated the robustness of S3RNet model. We have benchmarked our results with some con-

temporary models: UNet, FCN, and ModileNet.

As an addition to our core work, we propose a new deep sign recognition model where two or more different nationality sign datasets are combined and normalized to make a two-phased recognition engine from of each sign to its corresponding ethnic language. This cross-sign recognition approach has been implemented with different variants of CNN models. We claim this cross-sign model is the standard procedure to be followed for the recognition of any sets of cross-language datasets. The results from the cross-sign experiments have shown convincing recognition performance without major intra-language recognition mismatch for multi-ethnic gestures.

6.1 Limitation and Future Work

The sign gesture datasets used in this thesis do not contain occlusions. Rather, variation in illumination and blur are included as specific requirements for dataset, in addition to complex background. In our future research, we have plan to design and work on a dataset that has occluded gestures. Also we aspire to advance our research in two other emerging fields: dynamic activity recognition and complex sign sentence recognition. Both of these fields deal with dynamic video data. Our primary goal would be to design a time efficient object tracking model that can recognize the meanings of activities and take actions accordingly. In order to advance our current work, we plan to work on developing best-in-class model for real-time segmentation and recognition of egocentric dynamic gestures. Here also, our main focus would be to optimize the overall architecture.

APPENDIX A

SUPPLEMENTARY MATERIAL

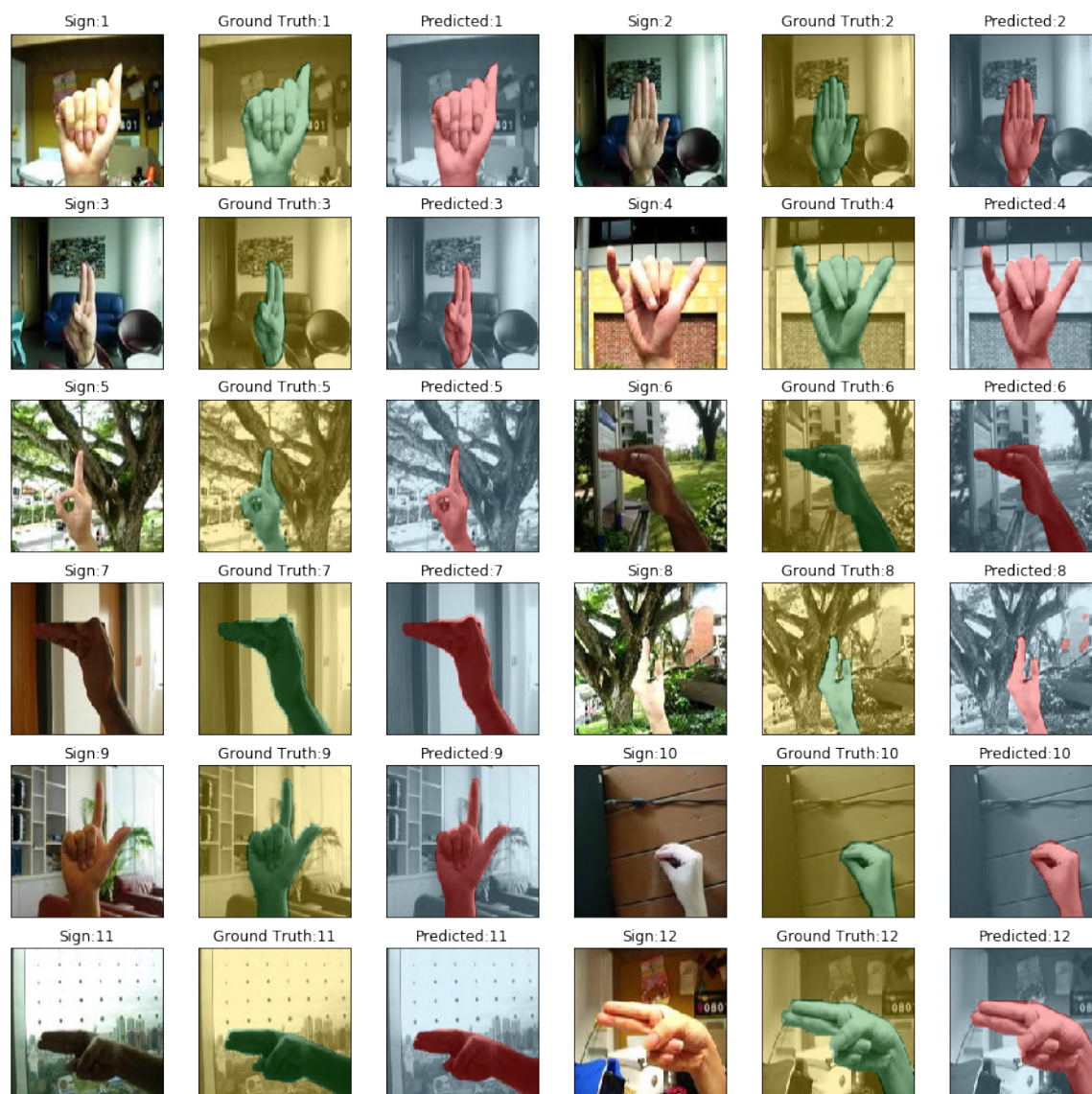


Figure A.1: Sensitivity analysis for brightness range $[0.2, 1.8]$

	precision	recall	f1-score	support
a	0.9227	0.8650	0.8929	400
b	0.9974	0.9500	0.9731	400
c	0.9508	0.9175	0.9338	400
d	0.9713	0.8475	0.9052	400
e	0.7702	0.9550	0.8527	400
f	0.9758	0.8075	0.8837	400
g	0.8929	0.9800	0.9344	400
h	1.0000	0.8450	0.9160	400
i	0.8404	0.9475	0.8907	400
j	0.8449	0.9125	0.8774	400
micro avg	0.9077	0.9028	0.9052	4000
macro avg	0.9166	0.9027	0.9060	4000
weighted avg	0.9166	0.9028	0.9060	4000
samples avg	0.9028	0.9028	0.9028	4000

Figure A.2: Performance measures: rotation range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII

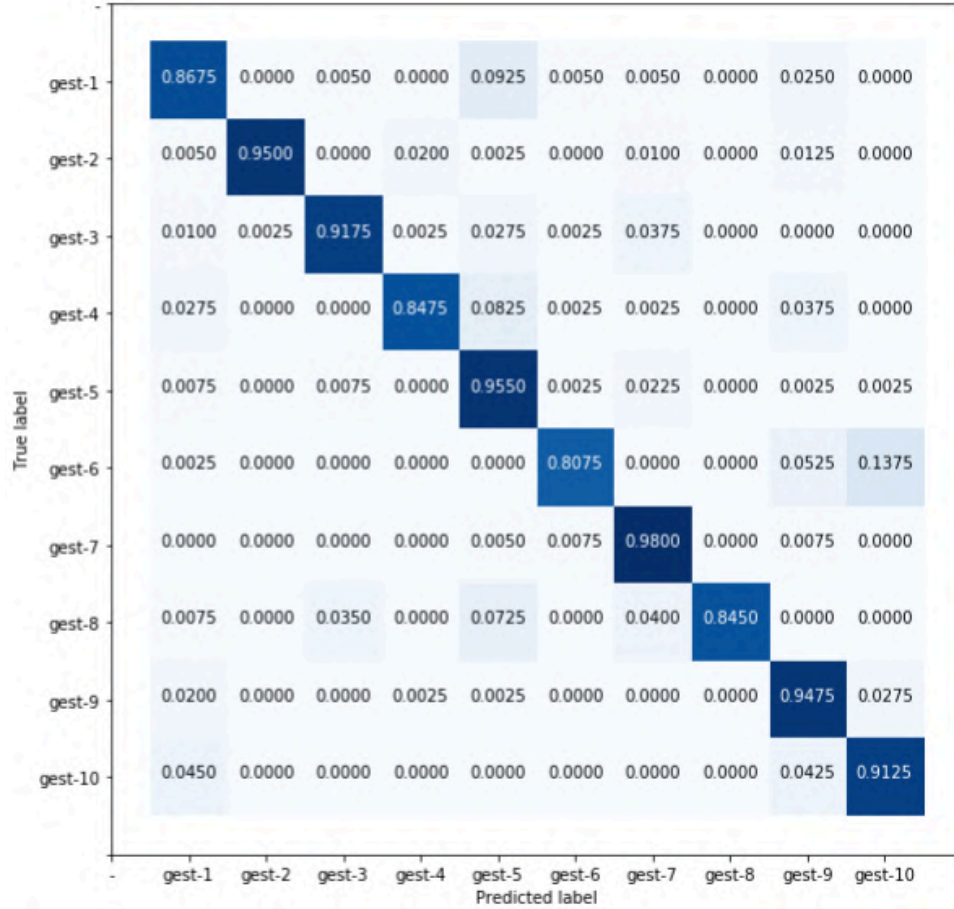


Figure A.3: Confusion matrix: rotation range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.9638	0.9325	0.9479	400
b	1.0000	0.9700	0.9848	400
c	0.9747	0.9625	0.9686	400
d	0.9893	0.9225	0.9547	400
e	0.8717	0.9850	0.9249	400
f	0.9945	0.9025	0.9463	400
g	0.9545	0.9975	0.9756	400
h	1.0000	0.9275	0.9624	400
i	0.9403	0.9850	0.9621	400
j	0.9133	0.9750	0.9432	400
micro avg	0.9577	0.9560	0.9568	4000
macro avg	0.9602	0.9560	0.9570	4000
weighted avg	0.9602	0.9560	0.9570	4000
samples avg	0.9560	0.9560	0.9560	4000

Figure A.4: Performance measures: rotation range $[0 - 25.0]^\circ$ with S3RNet on NUS-HII

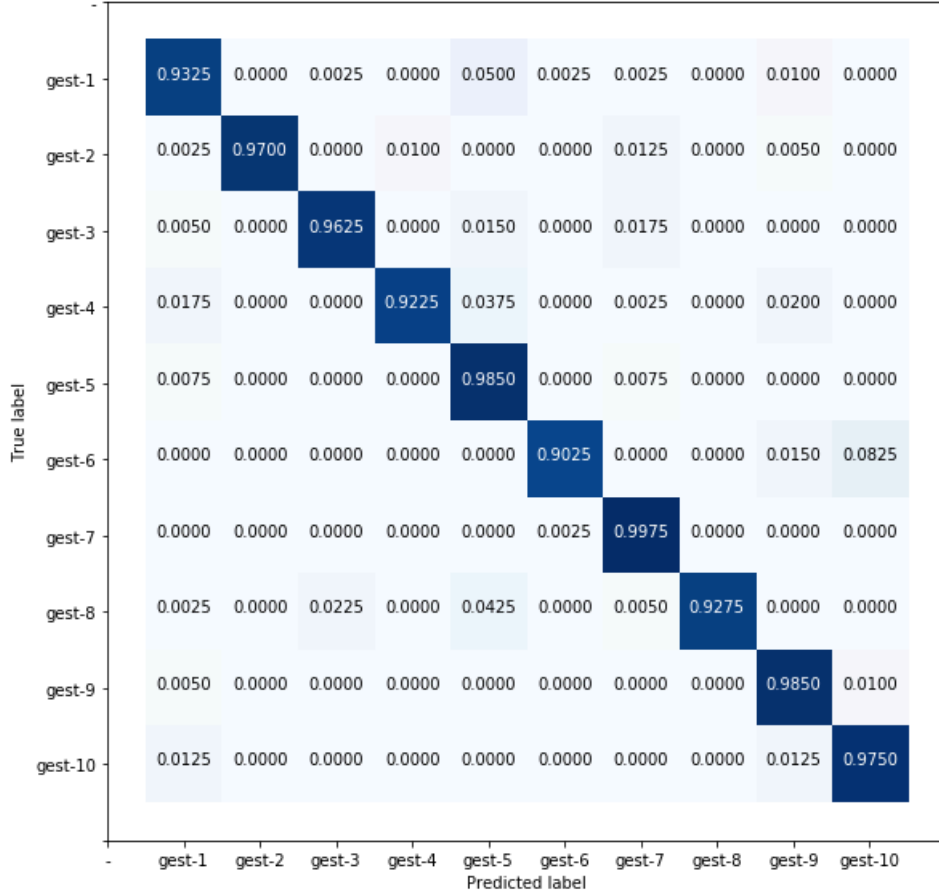


Figure A.5: Confusion matrix: rotation range $[0 - 30.0]^\circ$ with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.9874	0.9825	0.9850	400
b	1.0000	0.9925	0.9962	400
c	0.9924	0.9800	0.9862	400
d	0.9949	0.9700	0.9823	400
e	0.9455	0.9975	0.9708	400
f	1.0000	0.9650	0.9822	400
g	0.9950	1.0000	0.9975	400
h	1.0000	0.9725	0.9861	400
i	0.9925	0.9975	0.9950	400
j	0.9638	0.9975	0.9803	400
micro avg	0.9867	0.9855	0.9861	4000
macro avg	0.9872	0.9855	0.9862	4000
weighted avg	0.9872	0.9855	0.9862	4000
samples avg	0.9855	0.9855	0.9855	4000

Figure A.6: Performance measures: rotation range $[0 - 20.0]^\circ$ with S3RNet on NUS-HII

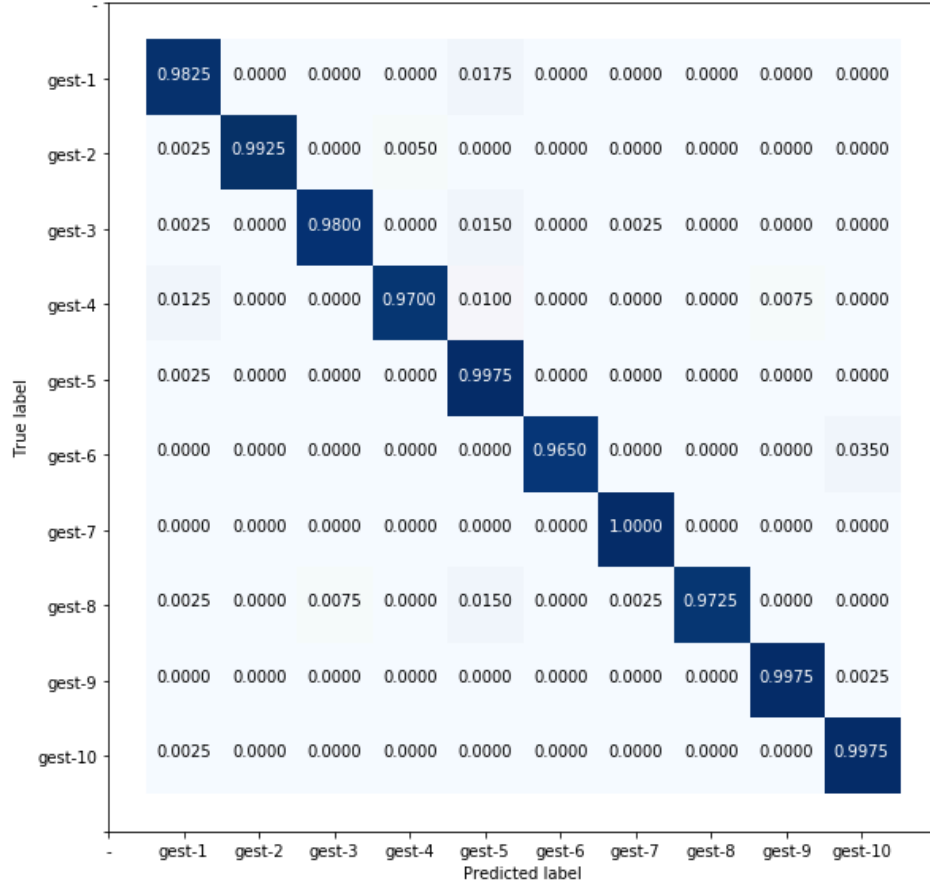


Figure A.7: Confusion matrix: rotation range $[0 - 20.0]^\circ$ with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.6467	0.5125	0.5718	400
b	0.7946	0.5125	0.6231	400
c	0.3992	0.7825	0.5287	400
d	0.5882	0.5000	0.5405	400
e	0.9950	0.4975	0.6633	400
f	0.6040	0.5225	0.5603	400
g	0.9318	0.5125	0.6613	400
h	0.3261	0.4875	0.3908	400
i	0.7475	0.5625	0.6419	400
j	0.7705	0.5625	0.6503	400
micro avg	0.5966	0.5453	0.5697	4000
macro avg	0.6804	0.5453	0.5832	4000
weighted avg	0.6804	0.5453	0.5832	4000
samples avg	0.5453	0.5453	0.5453	4000

Figure A.8: Performance measures: width shift [0.5] with S3RNet on NUS-HII

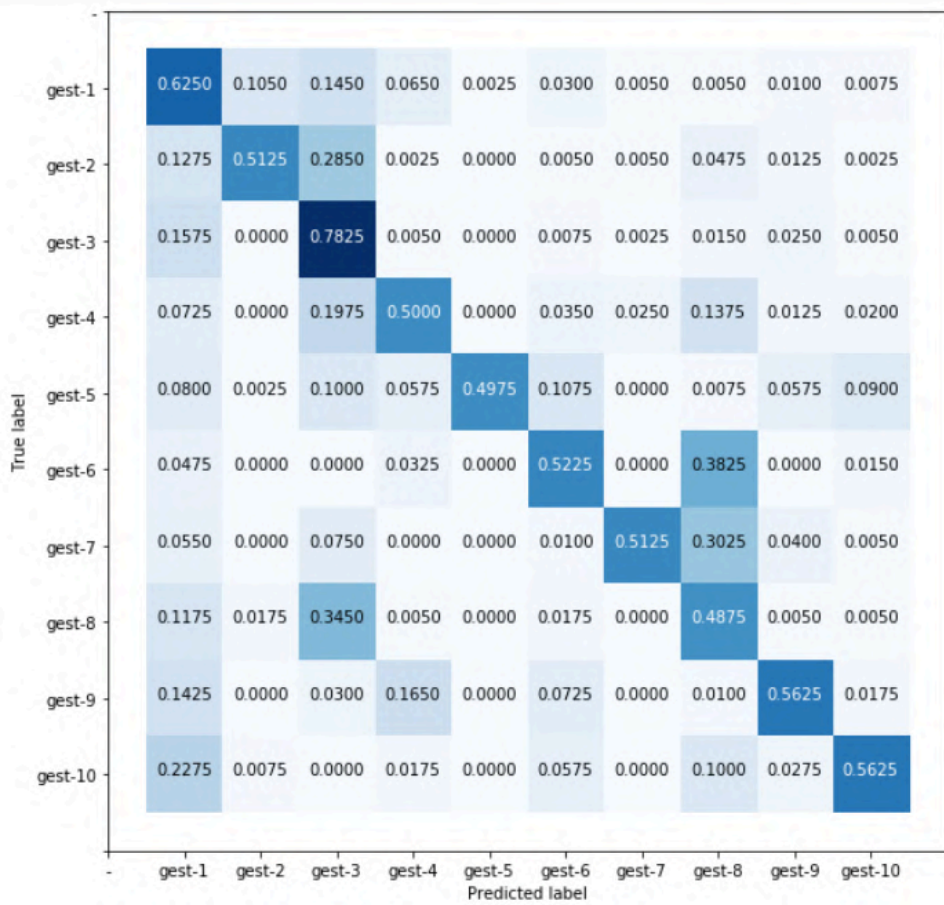


Figure A.9: Confusion matrix: width shift [0.5] with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.9974	0.9625	0.9796	400
b	0.9751	0.9775	0.9763	400
c	0.8803	0.9925	0.9330	400
d	0.9924	0.9750	0.9836	400
e	0.9973	0.9325	0.9638	400
f	0.9431	0.9950	0.9684	400
g	0.9899	0.9775	0.9836	400
h	0.9947	0.9450	0.9692	400
i	0.9728	0.9850	0.9789	400
j	1.0000	0.9575	0.9783	400
micro avg	0.9724	0.9700	0.9712	4000
macro avg	0.9743	0.9700	0.9715	4000
weighted avg	0.9743	0.9700	0.9715	4000
samples avg	0.9700	0.9700	0.9700	4000

Figure A.10: Performance measures: width shift [0.3] with S3RNet on NUS-HII



Figure A.11: Confusion matrix: width shift [0.3] with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.9732	1.0000	0.9864	400
b	1.0000	0.9975	0.9987	400
c	0.9974	0.9750	0.9861	400
d	1.0000	0.9725	0.9861	400
e	0.9756	1.0000	0.9877	400
f	1.0000	0.9800	0.9899	400
g	1.0000	1.0000	1.0000	400
h	1.0000	1.0000	1.0000	400
i	0.9901	1.0000	0.9950	400
j	0.9827	0.9925	0.9876	400
micro avg	0.9918	0.9918	0.9918	4000
macro avg	0.9919	0.9918	0.9917	4000
weighted avg	0.9919	0.9918	0.9917	4000
samples avg	0.9918	0.9918	0.9918	4000

Figure A.12: Performance measures: shear range $[0-20.0]^\circ$ with S3RNet on NUS-HII



Figure A.13: Confusion matrix: shear range $[0-20.0]^\circ$ with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.6078	0.9800	0.7502	400
b	0.9949	0.9700	0.9823	400
c	0.9724	0.9700	0.9712	400
d	0.9843	0.7825	0.8719	400
e	0.9838	0.9125	0.9468	400
f	0.9750	0.6825	0.8029	400
g	0.9875	0.9850	0.9862	400
h	0.9704	0.9850	0.9777	400
i	0.7591	0.8825	0.8162	400
j	0.8297	0.6575	0.7336	400
micro avg	0.8830	0.8808	0.8819	4000
macro avg	0.9065	0.8808	0.8839	4000
weighted avg	0.9065	0.8808	0.8839	4000
samples avg	0.8808	0.8808	0.8808	4000

Figure A.14: Performance measures: sear range $[0 - 45.0]^\circ$ with S3RNet on NUS-HII

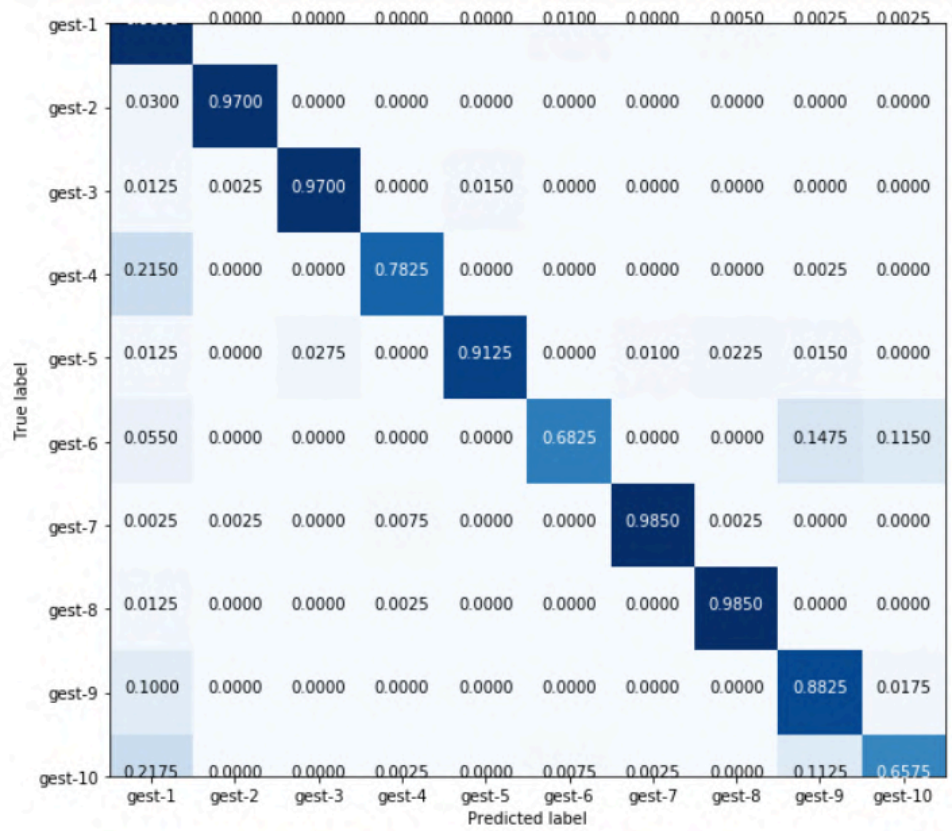


Figure A.15: Confusion matrix: sear range $[0 - 45.0]^\circ$ with S3RNet on NUS-HII

	precision	recall	f1-score	support
a	0.9087	0.9950	0.9499	400
b	1.0000	0.9975	0.9987	400
c	0.9949	0.9775	0.9861	400
d	1.0000	0.9425	0.9704	400
e	0.9779	0.9975	0.9876	400
f	0.9890	0.9025	0.9438	400
g	1.0000	1.0000	1.0000	400
h	1.0000	0.9950	0.9975	400
i	0.9039	0.9875	0.9438	400
j	0.9271	0.8900	0.9082	400
micro avg	0.9687	0.9685	0.9686	4000
macro avg	0.9702	0.9685	0.9686	4000
weighted avg	0.9702	0.9685	0.9686	4000
samples avg	0.9685	0.9685	0.9685	4000

Figure A.16: Performance measures: shear range $[0-30.0]^\circ$ with S3RNet on NUS-HII

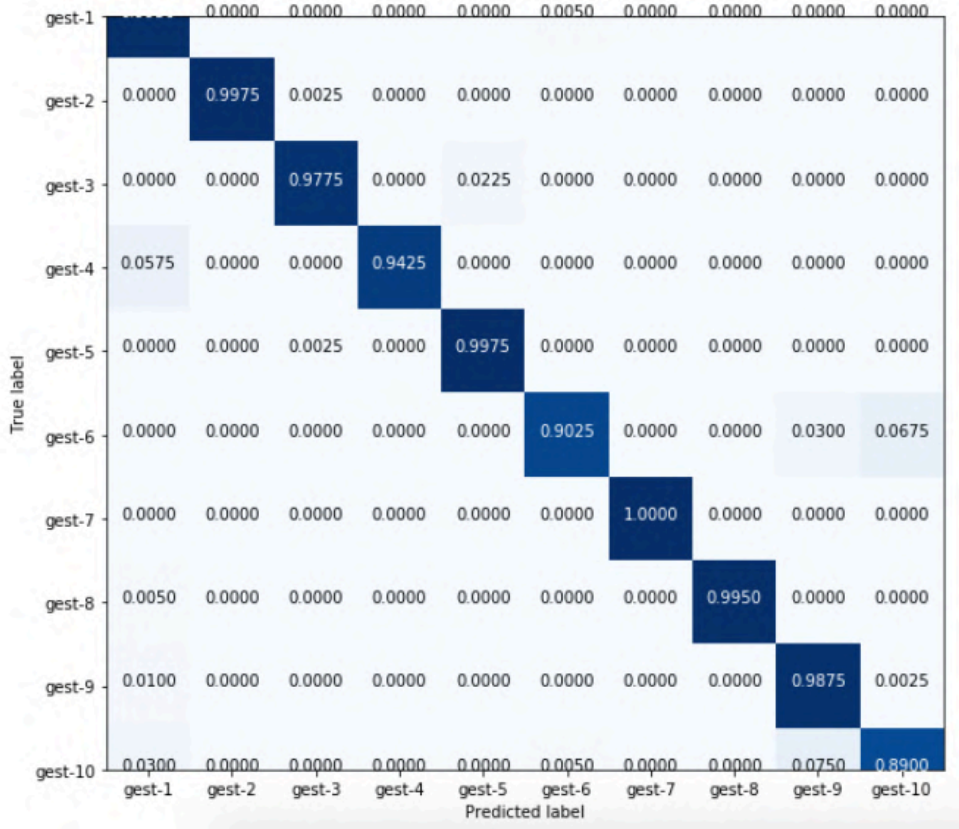


Figure A.17: Confusion matrix: shear range $[0-30.0]^\circ$ with S3RNet on NUS-HII

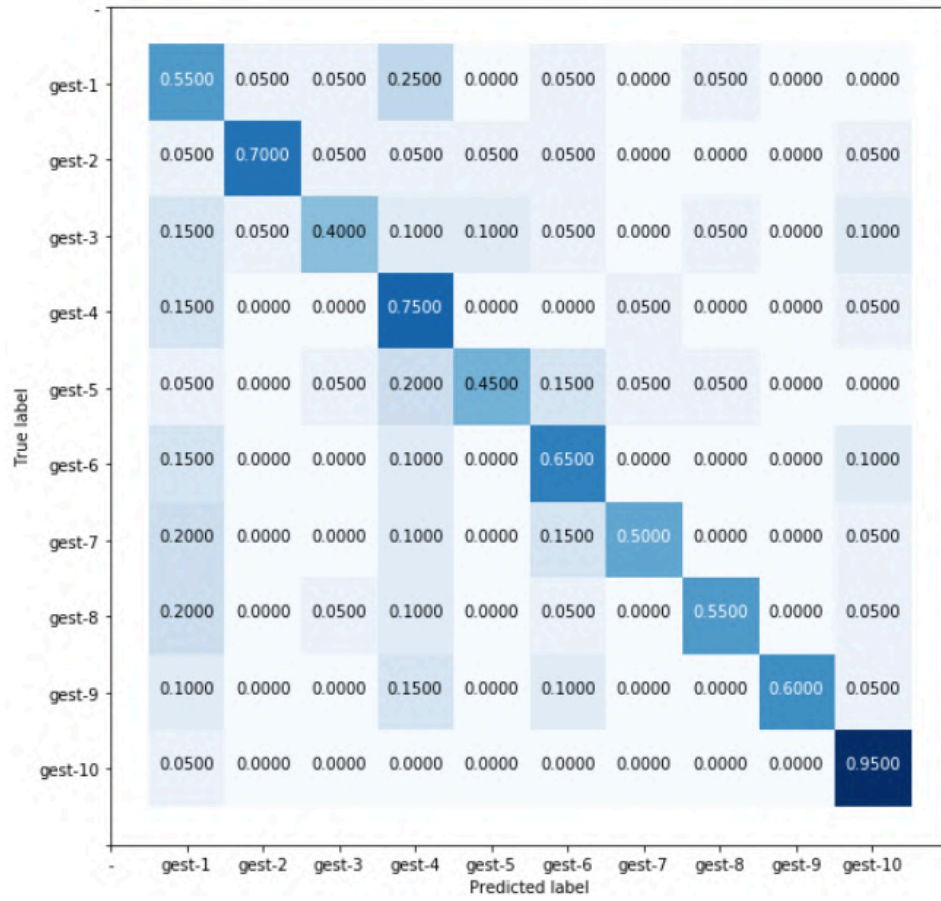


Figure A.18: Confusion matrix: UNet on NUS-HII

	precision	recall	f1-score	support
a	0.7500	0.4500	0.5625	20
b	0.8750	0.7000	0.7778	20
c	0.6667	0.4000	0.5000	20
d	0.4167	0.7500	0.5357	20
e	0.7500	0.4500	0.5625	20
f	0.5200	0.6500	0.5778	20
g	0.8333	0.5000	0.6250	20
h	0.7857	0.5500	0.6471	20
i	1.0000	0.6000	0.7500	20
j	0.6786	0.9500	0.7917	20
micro avg	0.6704	0.6000	0.6332	200
macro avg	0.7276	0.6000	0.6330	200
weighted avg	0.7276	0.6000	0.6330	200
samples avg	0.6000	0.6000	0.6000	200

Figure A.19: Performance: UNet on NUS-HII

	precision	recall	f1-score	support
a	0.9474	0.9000	0.9231	20
b	0.8696	1.0000	0.9302	20
c	1.0000	0.9500	0.9744	20
d	0.9474	0.9000	0.9231	20
e	1.0000	1.0000	1.0000	20
f	1.0000	0.9000	0.9474	20
g	0.9500	0.9500	0.9500	20
h	1.0000	1.0000	1.0000	20
i	1.0000	1.0000	1.0000	20
j	0.9524	1.0000	0.9756	20
micro avg	0.9648	0.9600	0.9624	200
macro avg	0.9667	0.9600	0.9624	200
weighted avg	0.9667	0.9600	0.9624	200
samples avg	0.9600	0.9600	0.9600	200

Figure A.20: Performance measures: FCN16 on NUS-HII

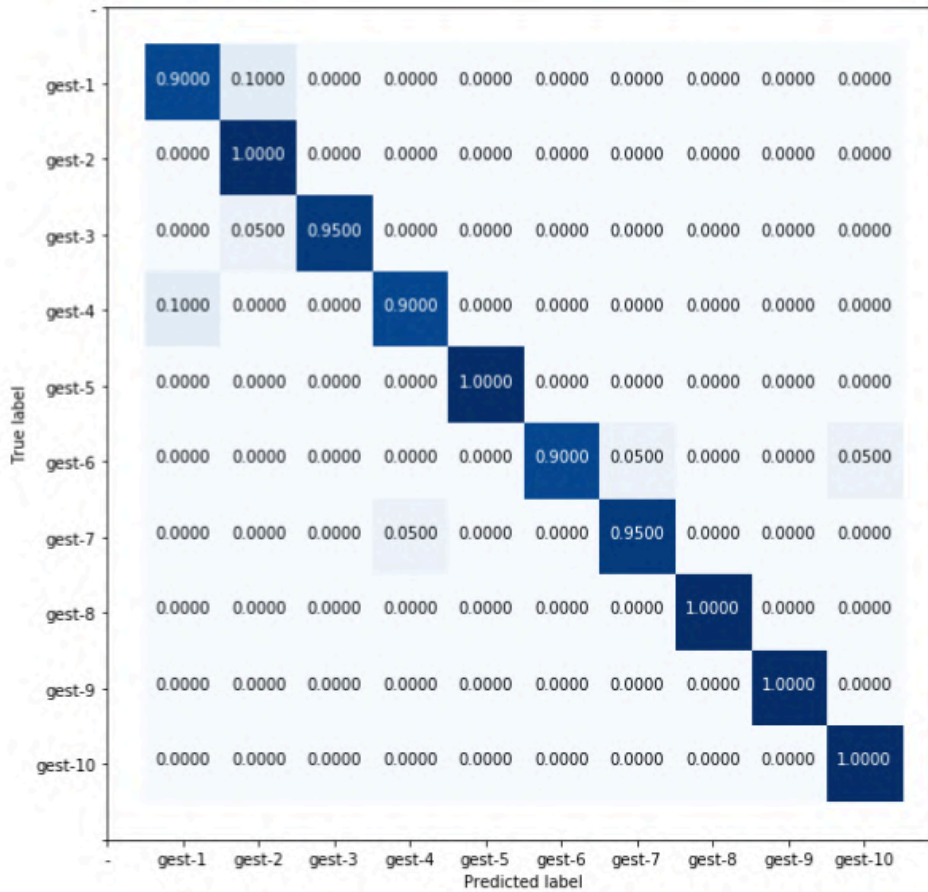


Figure A.21: Confusion matrix: FCN16 on NUS-HII

	precision	recall	f1-score	support
a	0.5000	0.8000	0.6154	20
b	0.6897	1.0000	0.8163	20
c	0.9444	0.8500	0.8947	20
d	0.8824	0.7500	0.8108	20
e	0.8571	0.3000	0.4444	20
f	0.9000	0.9000	0.9000	20
g	0.7600	0.9500	0.8444	20
h	1.0000	0.4000	0.5714	20
i	0.9524	1.0000	0.9756	20
j	0.9474	0.9000	0.9231	20
micro avg	0.8010	0.7850	0.7929	200
macro avg	0.8433	0.7850	0.7796	200
weighted avg	0.8433	0.7850	0.7796	200
samples avg	0.7850	0.7850	0.7850	200

Figure A.22: Performance measures: FCN32 on NUS-HII

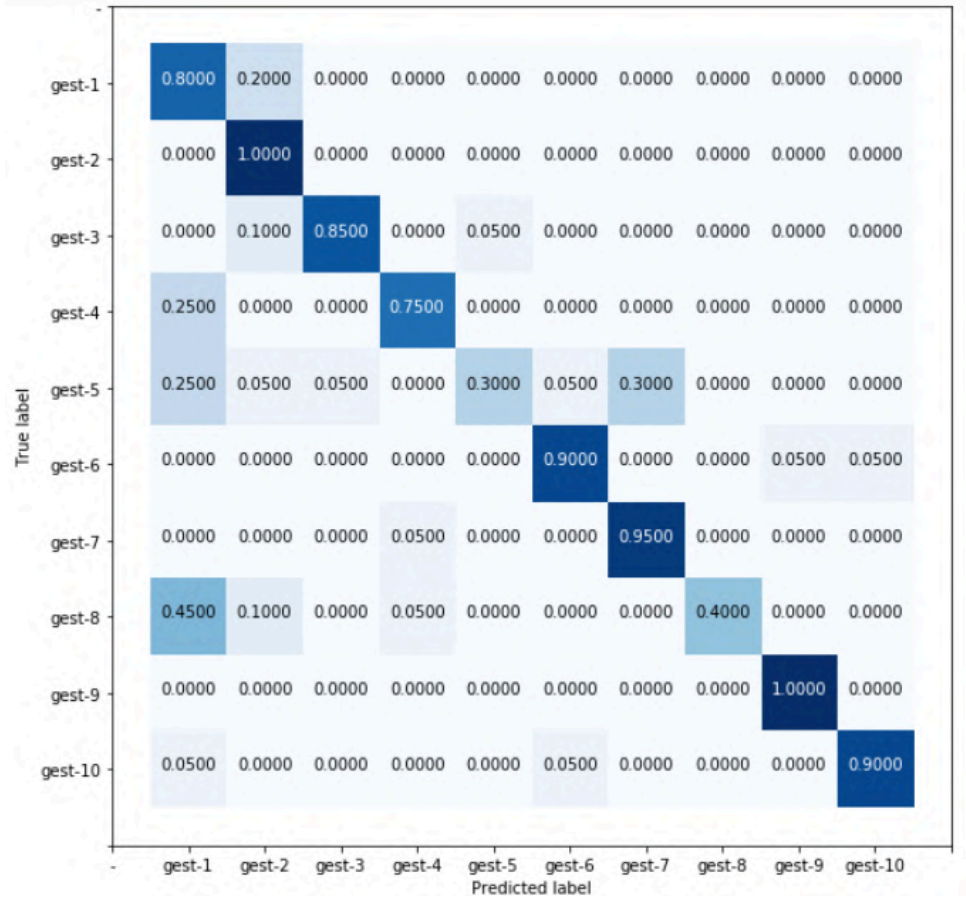


Figure A.23: Confusion matrix: FCN32 on NUS-HII

	precision	recall	f1-score	support
a	0.7764	0.9246	0.8440	199
b	0.8956	0.8191	0.8556	199
c	0.9942	0.8643	0.9247	199
d	0.9133	0.8995	0.9063	199
e	0.9119	0.8844	0.8980	199
f	0.9330	0.9095	0.9211	199
g	0.9824	0.8392	0.9051	199
h	0.9487	0.7437	0.8338	199
i	0.9343	0.9296	0.9320	199
j	0.9846	0.9648	0.9746	199
micro avg	0.9224	0.8779	0.8996	1990
macro avg	0.9274	0.8779	0.8995	1990
weighted avg	0.9274	0.8779	0.8995	1990
samples avg	0.8779	0.8779	0.8779	1990

Figure A.24: Performance measures: MobileNet on NUS-HII

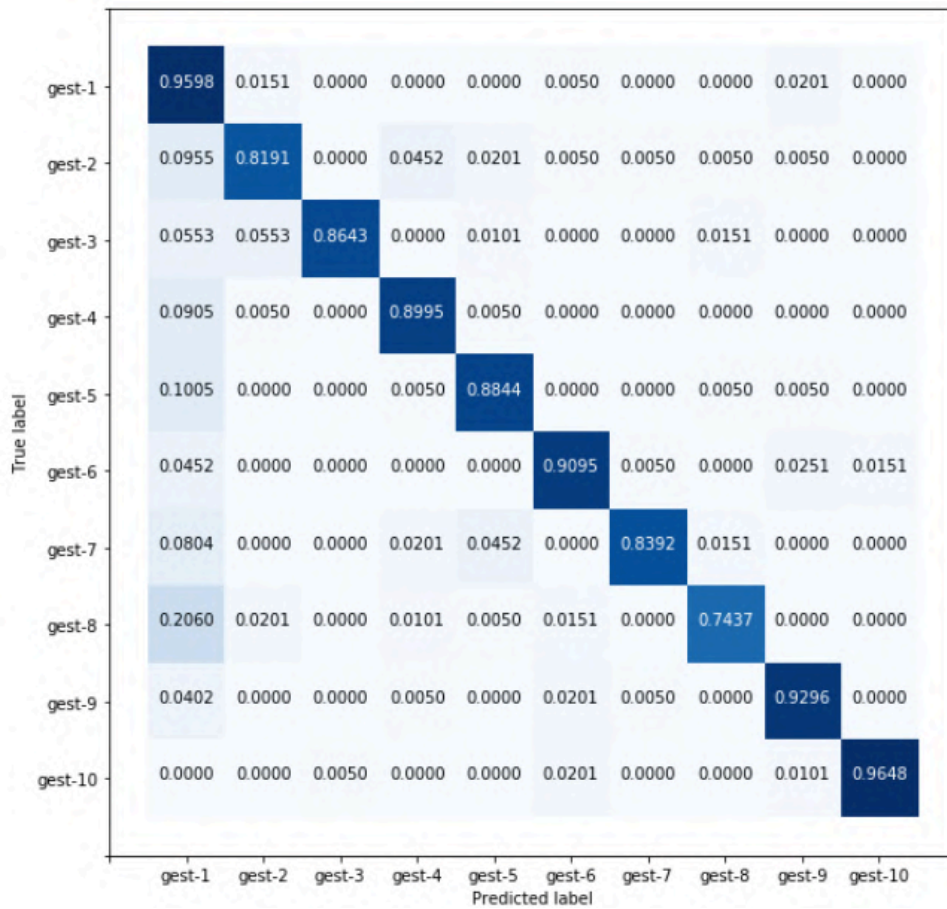


Figure A.25: Confusion matrix: MobileNet on NUS-HII

	precision	recall	f1-score	support
a	0.8706	0.8794	0.8750	199
b	0.9821	0.8291	0.8992	199
c	0.9689	0.9397	0.9541	199
d	0.9254	0.9347	0.9300	199
e	0.9679	0.9095	0.9378	199
f	0.9545	0.9497	0.9521	199
g	0.9439	0.9296	0.9367	199
h	0.9492	0.8442	0.8936	199
i	0.9330	0.9799	0.9559	199
j	0.9948	0.9548	0.9744	199
micro avg	0.9479	0.9151	0.9312	1990
macro avg	0.9490	0.9151	0.9309	1990
weighted avg	0.9490	0.9151	0.9309	1990
samples avg	0.9151	0.9151	0.9151	1990

Figure A.26: Performance measures: DLSR on augmented NUS-HII



Figure A.27: Confusion matrix: DLSR on augmented NUS-HII

	precision	recall	f1-score	support
a	1.0000	0.9500	0.9744	20
b	0.9091	1.0000	0.9524	20
c	1.0000	0.9500	0.9744	20
d	1.0000	1.0000	1.0000	20
e	1.0000	1.0000	1.0000	20
f	1.0000	0.9500	0.9744	20
g	1.0000	1.0000	1.0000	20
h	1.0000	1.0000	1.0000	20
i	1.0000	1.0000	1.0000	20
j	0.9524	1.0000	0.9756	20
micro avg	0.9850	0.9850	0.9850	200
macro avg	0.9861	0.9850	0.9851	200
weighted avg	0.9861	0.9850	0.9851	200
samples avg	0.9850	0.9850	0.9850	200

Figure A.28: Performance measures: S3RNet on augmented NUS-HII dataset

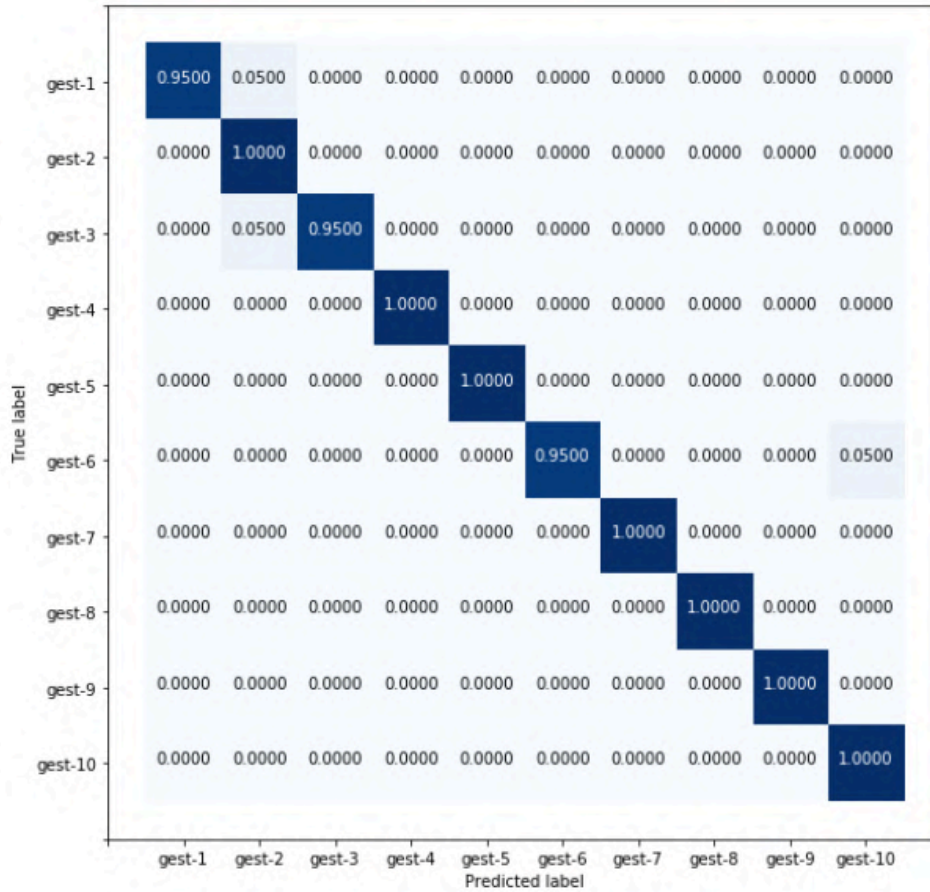


Figure A.29: Confusion matrix: S3RNet on augmented NUS-HII

	precision	recall	f1-score	support
a	0.9975	0.9950	0.9962	400
b	1.0000	0.9975	0.9987	400
c	1.0000	0.9950	0.9975	400
d	1.0000	0.9975	0.9987	400
e	0.9975	0.9975	0.9975	400
f	1.0000	0.9900	0.9950	400
g	0.9975	1.0000	0.9988	400
h	0.9950	0.9975	0.9963	400
i	0.9950	1.0000	0.9975	400
j	0.9950	1.0000	0.9975	400
micro avg	0.9977	0.9970	0.9974	4000
macro avg	0.9978	0.9970	0.9974	4000
weighted avg	0.9978	0.9970	0.9974	4000
samples avg	0.9970	0.9970	0.9970	4000

Figure A.30: Performance measures: S3RNet on NUS-HII

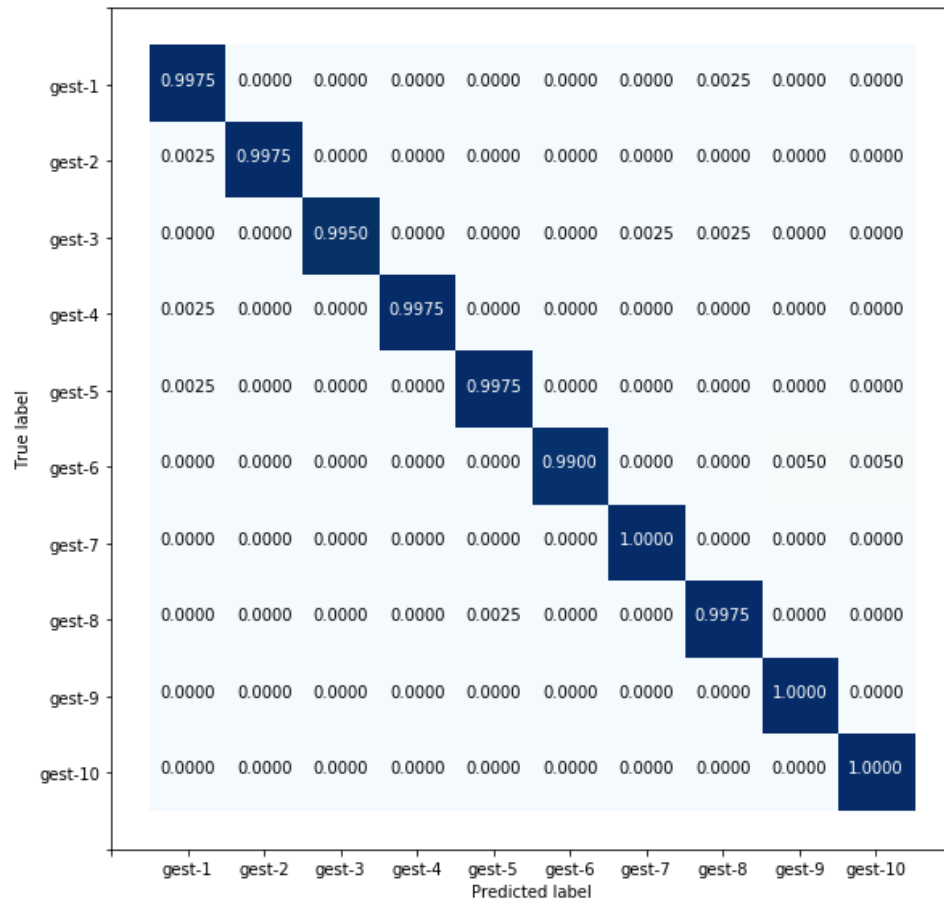


Figure A.31: Confusion matrix: S3RNet on NUS-HII

	precision	recall	f1-score	support
ain	0.9574	0.9052	0.9306	422
al	0.9868	0.8364	0.9054	269
aleff	0.9418	0.8283	0.8814	332
bb	0.9752	0.8774	0.9238	359
dal	0.9368	0.8190	0.8740	326
dha	0.8533	0.9101	0.8808	345
dhad	0.9674	0.8892	0.9267	334
fa	0.8766	0.8721	0.8744	391
gaaf	0.9141	0.7801	0.8418	341
ghain	0.9517	0.8987	0.9245	395
ha	0.8212	0.8495	0.8351	319
haa	0.9030	0.8852	0.8940	305
jeem	0.8907	0.8935	0.8921	310
kaaf	0.8915	0.8588	0.8748	354
khaa	0.9199	0.8913	0.9054	322
la	0.9583	0.9226	0.9401	349
laam	0.9517	0.9153	0.9331	366
meem	0.9153	0.9178	0.9165	353
nun	0.9626	0.9203	0.9410	364
ra	0.9301	0.9217	0.9259	332
saad	0.9155	0.8870	0.9010	354
seen	0.8839	0.9083	0.8959	327
sheen	0.9750	0.9040	0.9381	302
ta	0.9489	0.8705	0.9080	363
taa	0.9337	0.8424	0.8857	368
thaa	0.9342	0.8442	0.8869	353
thal	0.9299	0.9241	0.9270	316
toot	0.9413	0.8966	0.9185	358
waw	0.9537	0.9015	0.9268	274
ya	0.9713	0.8841	0.9256	345
yaa	0.9754	0.9225	0.9482	258
zay	0.9091	0.8727	0.8905	275
micro avg	0.9290	0.8829	0.9054	10781
macro avg	0.9305	0.8828	0.9054	10781
weighted avg	0.9304	0.8829	0.9054	10781
samples avg	0.8829	0.8829	0.8829	10781

Figure A.32: Performance measures: AlexNet on ArASL-PMU

	precision	recall	f1-score	support
A	0.9785	0.9867	0.9826	600
B	0.9704	0.9850	0.9777	600
C	0.9803	0.9950	0.9876	600
D	0.9899	0.9767	0.9832	600
E	0.9847	0.9683	0.9765	600
F	0.9983	0.9933	0.9958	600
G	0.9931	0.9617	0.9771	600
H	0.9567	0.9933	0.9747	600
I	0.9950	0.9900	0.9925	600
J	0.9983	0.9933	0.9958	600
K	0.9933	0.9850	0.9891	600
L	0.9950	0.9933	0.9942	600
M	0.9724	0.9967	0.9844	600
N	0.9949	0.9817	0.9883	600
O	0.9949	0.9700	0.9823	600
P	0.9950	0.9867	0.9908	600
Q	0.9884	0.9933	0.9909	600
R	0.9851	0.9950	0.9900	600
S	0.9630	0.9967	0.9795	600
T	0.9966	0.9733	0.9848	600
U	0.9733	0.9733	0.9733	600
V	0.9643	0.9467	0.9554	600
W	0.9610	0.9850	0.9728	600
X	0.9841	0.9300	0.9563	600
Y	0.9785	0.9867	0.9826	600
Z	0.9901	0.9967	0.9934	600
micro avg	0.9835	0.9821	0.9828	15600
macro avg	0.9837	0.9821	0.9827	15600
weighted avg	0.9837	0.9821	0.9827	15600
samples avg	0.9821	0.9821	0.9821	15600

Figure A.33: Performance measures: AlexNet on GRASSK-ASL

	precision	recall	f1-score	support
ain	0.9741	0.9810	0.9776	422
al	0.9594	0.9665	0.9630	269
aleff	0.9776	0.9132	0.9443	334
bb	0.9853	0.9304	0.9570	359
dal	0.9444	0.9387	0.9415	326
dha	0.9752	0.9101	0.9415	345
dhad	0.9666	0.9521	0.9593	334
fa	0.9082	0.9361	0.9219	391
gaaf	0.9767	0.8592	0.9142	341
ghain	0.9738	0.9418	0.9575	395
ha	0.9536	0.9028	0.9275	319
haa	0.9754	0.9115	0.9424	305
jeem	0.9105	0.9194	0.9149	310
kaaf	0.9441	0.9521	0.9481	355
khaa	0.9519	0.9224	0.9369	322
la	0.9853	0.9599	0.9724	349
laam	0.9749	0.9563	0.9655	366
meem	0.9796	0.9518	0.9655	353
nun	0.9747	0.9533	0.9639	364
ra	0.9564	0.9247	0.9403	332
saad	0.9385	0.9492	0.9438	354
seen	0.9842	0.9511	0.9673	327
sheen	0.9932	0.9603	0.9764	302
ta	0.9634	0.9421	0.9526	363
taa	0.9425	0.9348	0.9386	368
thaa	0.9486	0.9405	0.9445	353
thal	0.9674	0.9399	0.9535	316
toot	0.9579	0.9525	0.9552	358
waw	0.9778	0.9635	0.9706	274
ya	0.9849	0.9478	0.9660	345
yaa	0.9960	0.9729	0.9843	258
zay	0.9239	0.9273	0.9256	275
micro avg	0.9628	0.9396	0.9511	10784
macro avg	0.9633	0.9395	0.9511	10784
weighted avg	0.9631	0.9396	0.9510	10784
samples avg	0.9396	0.9396	0.9396	10784

Figure A.34: Performance measures: LeNet on ArASL-PMU

	precision	recall	f1-score	support
A	0.9917	0.9967	0.9942	600
B	0.9788	1.0000	0.9893	600
C	0.9983	0.9983	0.9983	600
D	0.9917	1.0000	0.9959	600
E	1.0000	0.9550	0.9770	600
F	0.9983	0.9950	0.9967	600
G	0.9900	0.9933	0.9917	600
H	0.9933	0.9917	0.9925	600
I	0.9934	1.0000	0.9967	600
J	0.9933	0.9950	0.9942	600
K	0.9967	1.0000	0.9983	600
L	0.9983	1.0000	0.9992	600
M	0.9933	0.9933	0.9933	600
N	0.9950	1.0000	0.9975	600
O	0.9933	0.9883	0.9908	600
P	0.9803	0.9950	0.9876	600
Q	0.9983	0.9800	0.9891	600
R	0.9949	0.9817	0.9883	600
S	0.9834	0.9850	0.9842	600
T	0.9983	0.9600	0.9788	600
U	0.9753	0.9883	0.9818	600
V	0.9833	0.9833	0.9833	600
W	0.9835	0.9917	0.9876	600
X	0.9884	0.9917	0.9900	600
Y	0.9933	0.9883	0.9908	600
Z	0.9917	0.9983	0.9950	600
micro avg	0.9913	0.9904	0.9909	15600
macro avg	0.9914	0.9904	0.9908	15600
weighted avg	0.9914	0.9904	0.9908	15600
samples avg	0.9904	0.9904	0.9904	15600

Figure A.35: Confusion matrix: LeNet on GRASSK-ASL

	precision	recall	f1-score	support
0	0.9818	0.9609	0.9712	281
1	1.0000	0.9893	0.9946	281
2	0.9823	0.9858	0.9840	281
3	1.0000	1.0000	1.0000	280
4	0.9856	0.9786	0.9821	280
5	0.9823	0.9858	0.9840	281
6	0.9609	0.9609	0.9609	281
7	1.0000	0.9857	0.9928	280
8	0.9929	0.9964	0.9947	281
9	0.9965	1.0000	0.9982	281
A	0.9964	1.0000	0.9982	280
B	1.0000	1.0000	1.0000	280
C	1.0000	1.0000	1.0000	281
D	1.0000	0.9964	0.9982	280
E	1.0000	1.0000	1.0000	280
F	1.0000	1.0000	1.0000	281
G	1.0000	1.0000	1.0000	280
H	1.0000	1.0000	1.0000	281
I	0.9965	1.0000	0.9982	281
J	1.0000	0.9964	0.9982	280
K	1.0000	1.0000	1.0000	281
L	1.0000	1.0000	1.0000	281
M	0.9929	0.9964	0.9947	281
N	0.9929	0.9893	0.9911	281
O	0.9617	0.9822	0.9718	281
P	1.0000	1.0000	1.0000	281
Q	1.0000	1.0000	1.0000	281
R	0.9965	1.0000	0.9982	281
S	1.0000	1.0000	1.0000	281
T	0.9962	0.9962	0.9962	261
U	1.0000	0.9964	0.9982	280
V	0.9856	0.9786	0.9821	280
W	0.9675	0.9571	0.9623	280
X	1.0000	1.0000	1.0000	281
Y	1.0000	1.0000	1.0000	280
Z	0.9929	1.0000	0.9964	280
micro avg	0.9933	0.9926	0.9930	10082
macro avg	0.9934	0.9926	0.9930	10082
weighted avg	0.9934	0.9926	0.9929	10082
samples avg	0.9926	0.9926	0.9926	10082

Figure A.36: Performance measures: TRNetV16 on Massey-G12

	precision	recall	f1-score	support
0	0.9890	0.9607	0.9746	280
1	0.9929	1.0000	0.9964	279
2	0.9790	1.0000	0.9894	280
3	1.0000	0.9892	0.9946	277
4	0.9964	1.0000	0.9982	280
5	0.9964	0.9964	0.9964	279
6	0.9964	0.9964	0.9964	280
7	1.0000	1.0000	1.0000	280
8	1.0000	1.0000	1.0000	280
9	1.0000	0.9964	0.9982	280
a	1.0000	1.0000	1.0000	279
b	1.0000	1.0000	1.0000	280
c	1.0000	1.0000	1.0000	280
d	1.0000	1.0000	1.0000	280
e	1.0000	1.0000	1.0000	280
f	1.0000	1.0000	1.0000	280
g	0.9929	1.0000	0.9964	280
h	1.0000	1.0000	1.0000	279
i	1.0000	1.0000	1.0000	280
j	1.0000	0.9964	0.9982	278
k	1.0000	1.0000	1.0000	279
l	0.9894	1.0000	0.9947	280
m	1.0000	1.0000	1.0000	280
n	0.9689	1.0000	0.9842	280
o	0.9892	0.9857	0.9875	280
p	1.0000	0.9893	0.9946	280
q	0.9929	1.0000	0.9964	280
r	1.0000	0.9964	0.9982	280
s	1.0000	0.9964	0.9982	280
t	0.9962	1.0000	0.9981	260
u	0.9964	1.0000	0.9982	280
v	1.0000	0.9779	0.9888	271
w	0.9967	0.9967	0.9967	300
x	1.0000	0.9967	0.9983	299
y	1.0000	1.0000	1.0000	299
z	1.0000	0.9866	0.9933	299
micro avg	0.9964	0.9961	0.9963	10118
macro avg	0.9965	0.9961	0.9963	10118
weighted avg	0.9965	0.9961	0.9963	10118
samples avg	0.9961	0.9961	0.9961	10118

Figure A.37: Performance measures: B4CN on Massey-G12

	precision	recall	f1-score	support
0	0.9964	0.9964	0.9964	280
1	1.0000	1.0000	1.0000	279
2	0.9964	0.9964	0.9964	280
3	0.9964	1.0000	0.9982	277
4	1.0000	1.0000	1.0000	280
5	1.0000	1.0000	1.0000	279
6	0.9858	0.9929	0.9893	280
7	1.0000	1.0000	1.0000	280
8	1.0000	1.0000	1.0000	280
9	1.0000	1.0000	1.0000	280
a	1.0000	0.9964	0.9982	279
b	1.0000	1.0000	1.0000	280
c	1.0000	1.0000	1.0000	280
d	1.0000	1.0000	1.0000	280
e	1.0000	1.0000	1.0000	280
f	1.0000	1.0000	1.0000	280
g	1.0000	1.0000	1.0000	280
h	1.0000	1.0000	1.0000	279
i	1.0000	1.0000	1.0000	280
j	1.0000	1.0000	1.0000	278
k	1.0000	1.0000	1.0000	279
l	1.0000	1.0000	1.0000	280
m	0.9929	1.0000	0.9964	280
n	1.0000	0.9964	0.9982	280
o	0.9964	0.9964	0.9964	280
p	1.0000	1.0000	1.0000	280
q	1.0000	1.0000	1.0000	280
r	1.0000	1.0000	1.0000	280
s	1.0000	1.0000	1.0000	280
t	1.0000	1.0000	1.0000	260
u	1.0000	1.0000	1.0000	280
v	1.0000	0.9963	0.9982	271
w	0.9933	0.9867	0.9900	300
x	1.0000	1.0000	1.0000	299
y	1.0000	1.0000	1.0000	299
z	1.0000	1.0000	1.0000	299
micro avg	0.9988	0.9988	0.9988	10118
macro avg	0.9988	0.9988	0.9988	10118
weighted avg	0.9988	0.9988	0.9988	10118
samples avg	0.9988	0.9988	0.9988	10118

Figure A.38: Performance measures: WB3CN on Massey-G12

	precision	recall	f1-score	support
ain	0.9811	0.9858	0.9835	422
al	0.9811	0.9628	0.9719	269
aleff	0.9726	0.9551	0.9637	334
bb	0.9749	0.9721	0.9735	359
dal	0.9810	0.9509	0.9657	326
dha	0.9819	0.9449	0.9631	345
dhad	0.9817	0.9611	0.9713	334
fa	0.9815	0.9488	0.9649	391
gaaf	0.9726	0.9355	0.9537	341
ghain	0.9922	0.9646	0.9782	395
ha	0.9436	0.9436	0.9436	319
haa	0.9797	0.9508	0.9651	305
jeem	0.9670	0.9452	0.9560	310
kaaf	0.9826	0.9549	0.9686	355
khaa	0.9708	0.9286	0.9492	322
la	0.9942	0.9885	0.9914	349
laam	0.9944	0.9781	0.9862	366
meem	0.9885	0.9717	0.9800	353
nun	0.9942	0.9451	0.9690	364
ra	0.9842	0.9367	0.9599	332
saad	0.9828	0.9024	0.9409	379
seen	0.9969	0.9755	0.9861	327
sheen	0.9966	0.9768	0.9866	302
ta	0.9613	0.9587	0.9600	363
taa	0.9915	0.9457	0.9680	368
thaa	0.9692	0.9802	0.9746	353
thal	0.9649	0.9557	0.9603	316
toot	0.9857	0.9637	0.9746	358
waw	0.9925	0.9599	0.9759	274
ya	0.9882	0.9710	0.9795	345
yaa	0.9961	0.9961	0.9961	258
zay	0.9699	0.9382	0.9538	275
micro avg	0.9811	0.9577	0.9693	10809
macro avg	0.9811	0.9578	0.9692	10809
weighted avg	0.9812	0.9577	0.9692	10809
samples avg	0.9577	0.9577	0.9577	10809

Figure A.39: Performance measures: WB3CN on ArASL-PMU

	precision	recall	f1-score	support
ain	0.9904	0.9787	0.9845	422
al	0.9778	0.9814	0.9796	269
aleff	0.9969	0.9551	0.9755	334
bb	0.9914	0.9666	0.9788	359
dal	0.9936	0.9571	0.9750	326
dha	0.9939	0.9478	0.9703	345
dhad	0.9704	0.9820	0.9762	334
fa	0.9683	0.9361	0.9519	391
gaaf	0.8847	0.9677	0.9244	341
ghain	0.9922	0.9696	0.9808	395
ha	0.9096	0.9781	0.9426	319
haa	0.9464	0.9836	0.9646	305
jeem	0.9578	0.9516	0.9547	310
kaaf	0.9358	0.9859	0.9602	355
khaa	0.9657	0.9627	0.9642	322
la	0.9971	0.9828	0.9899	349
laam	1.0000	0.9590	0.9791	366
meem	0.9972	0.9915	0.9943	353
nun	0.9971	0.9478	0.9718	364
ra	0.9727	0.9669	0.9698	332
saad	0.9911	0.8813	0.9330	379
seen	0.9758	0.9878	0.9818	327
sheen	0.9899	0.9735	0.9816	302
ta	0.9750	0.9669	0.9710	363
taa	0.9755	0.9728	0.9741	368
thaa	0.9200	0.9773	0.9478	353
thal	0.9933	0.9335	0.9625	316
toot	0.9885	0.9581	0.9730	358
waw	1.0000	0.9708	0.9852	274
ya	1.0000	0.9768	0.9883	345
yaa	0.9961	1.0000	0.9981	258
zay	0.9700	0.9418	0.9557	275
micro avg	0.9747	0.9648	0.9697	10809
macro avg	0.9754	0.9654	0.9700	10809
weighted avg	0.9756	0.9648	0.9698	10809
samples avg	0.9648	0.9648	0.9648	10809

Figure A.40: Performance measures: B4CN on ArASL-PMU

	precision	recall	f1-score	support
ain	0.9904	0.9748	0.9826	318
al	0.9949	0.9606	0.9774	203
aleff	1.0000	0.9802	0.9900	252
bb	0.9848	0.9628	0.9737	269
dal	0.9508	0.9431	0.9469	246
dha	0.9879	0.9423	0.9646	260
dhad	0.9959	0.9681	0.9818	251
fa	0.9822	0.9388	0.9600	294
gaaf	0.9786	0.8945	0.9347	256
ghain	0.9795	0.9630	0.9711	297
ha	0.9333	0.9333	0.9333	240
haa	0.9865	0.9522	0.9690	230
jeem	0.9655	0.9573	0.9614	234
kaaf	0.9736	0.9663	0.9699	267
khaa	0.9713	0.9793	0.9753	242
la	0.9812	0.9924	0.9868	263
laam	0.9963	0.9674	0.9816	276
meem	0.9772	0.9698	0.9735	265
nun	0.9850	0.9599	0.9723	274
ra	0.9756	0.9600	0.9677	250
saad	0.9819	0.9544	0.9680	285
seen	0.9915	0.9433	0.9668	247
sheen	0.9956	0.9868	0.9912	227
ta	0.9739	0.9560	0.9649	273
taa	0.9565	0.9531	0.9548	277
thaa	0.9810	0.9699	0.9754	266
thal	0.9784	0.9496	0.9638	238
toot	0.9707	0.9851	0.9779	269
waw	0.9949	0.9563	0.9752	206
ya	0.9922	0.9768	0.9844	259
yaa	0.9948	0.9795	0.9871	195
zay	0.9749	0.9372	0.9557	207
micro avg	0.9803	0.9599	0.9700	8136
macro avg	0.9805	0.9598	0.9700	8136
weighted avg	0.9804	0.9599	0.9700	8136
samples avg	0.9599	0.9599	0.9599	8136

Figure A.41: Performance measures: TRNetV16 on ArASL-PMU

	precision	recall	f1-score	support
A	1.0000	1.0000	1.0000	600
B	1.0000	1.0000	1.0000	600
C	1.0000	1.0000	1.0000	600
D	1.0000	1.0000	1.0000	600
E	1.0000	1.0000	1.0000	600
F	1.0000	1.0000	1.0000	600
G	1.0000	1.0000	1.0000	600
H	1.0000	1.0000	1.0000	600
I	1.0000	1.0000	1.0000	600
J	1.0000	1.0000	1.0000	600
K	1.0000	1.0000	1.0000	600
L	1.0000	1.0000	1.0000	600
M	1.0000	0.9983	0.9992	600
N	0.9983	1.0000	0.9992	600
O	1.0000	1.0000	1.0000	600
P	1.0000	1.0000	1.0000	600
Q	1.0000	1.0000	1.0000	600
R	1.0000	1.0000	1.0000	600
S	1.0000	1.0000	1.0000	600
T	1.0000	1.0000	1.0000	600
U	1.0000	1.0000	1.0000	600
V	1.0000	0.9983	0.9992	600
W	1.0000	1.0000	1.0000	600
X	1.0000	0.9983	0.9992	600
Y	1.0000	1.0000	1.0000	600
Z	0.9983	1.0000	0.9992	600
micro avg	0.9999	0.9998	0.9998	15600
macro avg	0.9999	0.9998	0.9998	15600
weighted avg	0.9999	0.9998	0.9998	15600
samples avg	0.9998	0.9998	0.9998	15600

Figure A.42: Performance measures: WB3CN on GRASSK-ASL

	precision	recall	f1-score	support
A	1.0000	1.0000	1.0000	600
B	1.0000	1.0000	1.0000	600
C	1.0000	1.0000	1.0000	600
D	1.0000	1.0000	1.0000	600
E	1.0000	1.0000	1.0000	600
F	1.0000	1.0000	1.0000	600
G	1.0000	1.0000	1.0000	600
H	1.0000	1.0000	1.0000	600
I	1.0000	1.0000	1.0000	600
J	1.0000	1.0000	1.0000	600
K	1.0000	1.0000	1.0000	600
L	1.0000	1.0000	1.0000	600
M	1.0000	0.9983	0.9992	600
N	0.9983	1.0000	0.9992	600
O	1.0000	1.0000	1.0000	600
P	1.0000	1.0000	1.0000	600
Q	1.0000	1.0000	1.0000	600
R	1.0000	1.0000	1.0000	600
S	1.0000	1.0000	1.0000	600
T	1.0000	1.0000	1.0000	600
U	1.0000	1.0000	1.0000	600
V	1.0000	0.9983	0.9992	600
W	1.0000	1.0000	1.0000	600
X	1.0000	0.9983	0.9992	600
Y	1.0000	1.0000	1.0000	600
Z	0.9983	1.0000	0.9992	600
micro avg	0.9999	0.9998	0.9998	15600
macro avg	0.9999	0.9998	0.9998	15600
weighted avg	0.9999	0.9998	0.9998	15600
samples avg	0.9998	0.9998	0.9998	15600

Figure A.43: Performance measures: B4CN on GRASSK-ASL

	precision	recall	f1-score	support
A	0.9979	0.9958	0.9969	480
B	1.0000	0.9938	0.9969	480
C	1.0000	1.0000	1.0000	480
D	1.0000	0.9979	0.9990	480
E	0.9876	0.9958	0.9917	480
F	1.0000	0.9958	0.9979	480
G	0.9877	1.0000	0.9938	480
H	1.0000	0.9875	0.9937	480
I	0.9979	0.9938	0.9958	480
J	1.0000	1.0000	1.0000	480
K	1.0000	0.9875	0.9937	480
L	1.0000	0.9979	0.9990	480
M	0.9937	0.9812	0.9874	480
N	0.9835	0.9938	0.9886	480
O	1.0000	0.9854	0.9927	480
P	1.0000	0.9979	0.9990	480
Q	0.9979	1.0000	0.9990	480
R	0.9957	0.9604	0.9777	480
S	0.9855	0.9938	0.9896	480
T	1.0000	0.9896	0.9948	480
U	0.9670	0.9771	0.9720	480
V	0.9750	0.9750	0.9750	480
W	0.9958	0.9812	0.9885	480
X	0.9978	0.9625	0.9799	480
Y	1.0000	0.9979	0.9990	480
Z	1.0000	1.0000	1.0000	480
micro avg	0.9947	0.9901	0.9924	12480
macro avg	0.9947	0.9901	0.9924	12480
weighted avg	0.9947	0.9901	0.9924	12480
samples avg	0.9901	0.9901	0.9901	12480

Figure A.44: Performance measures: TRNetV16 on GRASSK-ASL

	precision	recall	f1-score	support
a	0.8750	0.7000	0.7778	20
b	1.0000	0.9000	0.9474	20
c	0.8333	1.0000	0.9091	20
d	0.8636	0.9500	0.9048	20
e	1.0000	0.7500	0.8571	20
f	1.0000	0.7500	0.8571	20
g	0.8636	0.9500	0.9048	20
h	1.0000	0.8000	0.8889	20
i	0.8696	1.0000	0.9302	20
j	1.0000	0.9000	0.9474	20
micro avg	0.9206	0.8700	0.8946	200
macro avg	0.9305	0.8700	0.8925	200
weighted avg	0.9305	0.8700	0.8925	200
samples avg	0.8700	0.8700	0.8700	200

Figure A.45: Performance measures: B4CN on NUS-HII

	precision	recall	f1-score	support
a	0.9048	0.9500	0.9268	20
b	1.0000	0.8500	0.9189	20
c	0.9412	0.8000	0.8649	20
d	1.0000	0.9000	0.9474	20
e	1.0000	0.9500	0.9744	20
f	0.9444	0.8500	0.8947	20
g	1.0000	0.9500	0.9744	20
h	1.0000	0.9000	0.9474	20
i	0.9474	0.9000	0.9231	20
j	0.9444	0.8500	0.8947	20
micro avg	0.9674	0.8900	0.9271	200
macro avg	0.9682	0.8900	0.9267	200
weighted avg	0.9682	0.8900	0.9267	200
samples avg	0.8900	0.8900	0.8900	200

Figure A.46: Performance measures: B4CN on NUS-HII

	precision	recall	f1-score	support
a	0.8929	0.6250	0.7353	40
b	0.7576	0.6250	0.6849	40
c	0.8400	0.5250	0.6462	40
d	0.9231	0.6000	0.7273	40
e	0.8276	0.6000	0.6957	40
f	0.9333	0.7000	0.8000	40
g	0.8889	0.8000	0.8421	40
h	0.8276	0.6000	0.6957	40
i	0.8947	0.8500	0.8718	40
j	1.0000	0.9250	0.9610	40
micro avg	0.8810	0.6850	0.7707	400
macro avg	0.8786	0.6850	0.7660	400
weighted avg	0.8786	0.6850	0.7660	400
samples avg	0.6850	0.6850	0.6850	400

Figure A.47: Performance measures: TRNetV16 on NUS-HII

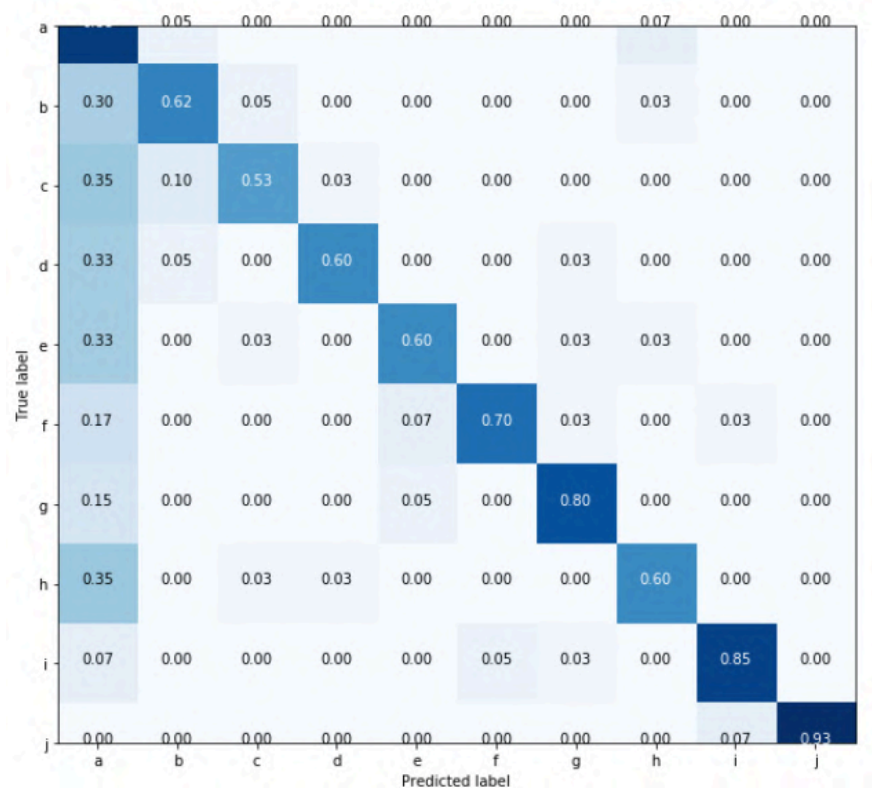


Figure A.48: Confusion matrix: TRNetV16 on NUS-HII



Figure A.49: Confusion matrix: B4CN on NUS-HII

	precision	recall	f1-score	support
a	0.9750	0.9750	0.9750	40
b	0.9286	0.9750	0.9512	40
c	0.9487	0.9250	0.9367	40
d	1.0000	0.9500	0.9744	40
e	1.0000	0.9750	0.9873	40
f	0.9500	0.9500	0.9500	40
g	0.9524	1.0000	0.9756	40
h	1.0000	0.9500	0.9744	40
i	1.0000	1.0000	1.0000	40
j	0.9512	0.9750	0.9630	40
micro avg	0.9699	0.9675	0.9687	400
macro avg	0.9706	0.9675	0.9688	400
weighted avg	0.9706	0.9675	0.9688	400
samples avg	0.9675	0.9675	0.9675	400

Figure A.50: Performance measures: TRNetV16 on NUS-HII (masked)

	precision	recall	f1-score	support
a	1.0000	0.4500	0.6207	40
b	0.8667	0.6500	0.7429	40
c	0.9474	0.4500	0.6102	40
d	0.8000	0.8000	0.8000	40
e	0.8056	0.7250	0.7632	40
f	0.9000	0.6750	0.7714	40
g	0.7917	0.9500	0.8636	40
h	0.9655	0.7000	0.8116	40
i	0.9355	0.7250	0.8169	40
j	0.9444	0.8500	0.8947	40
micro avg	0.8801	0.6975	0.7782	400
macro avg	0.8957	0.6975	0.7695	400
weighted avg	0.8957	0.6975	0.7695	400
samples avg	0.6975	0.6975	0.6975	400

Figure A.51: Performance measures: B4CN on NUS-HII (masked)

	precision	recall	f1-score	support
a	0.9744	0.9500	0.9620	40
b	1.0000	1.0000	1.0000	40
c	1.0000	0.9500	0.9744	40
d	1.0000	0.9500	0.9744	40
e	1.0000	0.9750	0.9873	40
f	0.9756	1.0000	0.9877	40
g	0.9756	1.0000	0.9877	40
h	0.9756	1.0000	0.9877	40
i	1.0000	0.9750	0.9873	40
j	1.0000	1.0000	1.0000	40
micro avg	0.9899	0.9800	0.9849	400
macro avg	0.9901	0.9800	0.9848	400
weighted avg	0.9901	0.9800	0.9848	400
samples avg	0.9800	0.9800	0.9800	400

Figure A.52: Performance measures: WB3CN on NUS-HII (masked)



Figure A.53: Confusion matrix: B4CN on NUS-HII

REFERENCES

- [1] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [2] M. Rossi, S. Benatti, E. Farella, and L. Benini, “Hybrid emg classifier based on hmm and svm for hand gesture recognition in prosthetics,” in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT)*, 2015.
- [3] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018.
- [4] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.
- [5] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Proceeding of the Springer International Conference on Medical Image Computing and Computer-assisted Intervention*, 2015, pp. 234–241.

- [6] S. Jégou, M. Drozdal, D. Vazquez, A. Romero, and Y. Bengio, “The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 11–19.
- [7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818.
- [8] L. Wang, W. Hu, and T. Tan, “Recent developments in human motion analysis,” *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, 2003.
- [9] M. J. Cheok, Z. Omar, and M. H. Jaward, “A review of hand gesture and sign language recognition techniques,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131–153, 2019.
- [10] Y. Wu and T. S. Huang, “Human hand modeling, analysis and animation in the context of hci,” in *Proceedings of the IEEE International Conference on Image Processing (Cat. 99CH36348)*, 1999.
- [11] Y. Nam and K. Wohn, “Recognition of space-time hand-gestures using hidden markov model,” in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 1996.

- [12] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [13] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2017.
- [14] A. Wexelblat, “Research challenges in gesture: Open issues and unsolved problems,” in *Springer International Gesture Workshop*, 1997, pp. 1–11.
- [15] A. Fourney, “Design and evaluation of a presentation maestro: Controlling electronic presentations through gesture,” Master’s thesis, University of Waterloo, 2009.
- [16] S. Vigneshwaran, M. ShifaFathima, V. VijaySagar *et al.*, “Hand gesture recognition and voice conversion system for dumb people,” in *Proceedings of the 5th IEEE International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019.
- [17] J. H. Seong and Y. Choi, “Design and implementation of user interface through hand movement tracking and gesture recognition,” in *Proceedings of the IEEE International Conference on Information and Communication Technology Convergence (ICTC)*, 2018.
- [18] S. Pancholi and A. M. Joshi, “Electromyography-based hand gesture recognition system for upper limb amputees,” *Sensors Letters*, vol. 3, no. 3, pp. 1–4, 2019.

- [19] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014.
- [20] H. Zhang, Y. Wang, and C. Deng, “Application of gesture recognition based on simulated annealing bp neural network,” in *Proceedings of the IEEE International Conference on Electronic & Mechanical Engineering and Information Technology*, 2011.
- [21] H. Lahiani, M. Elleuch, and M. Kherallah, “Real time hand gesture recognition system for android devices,” in *Proceedings of the 15th IEEE International Conference on Intelligent Systems Design and Applications (ISDA)*, 2015.
- [22] R. Rastgoo, K. Kiani, and S. Escalera, “Multi-modal deep hand sign language recognition in still images using restricted boltzmann machine,” *Entropy*, vol. 20, no. 11, p. 809, 2018.
- [23] J. R. Pansare, S. H. Gawande, and M. Ingle, “Real-time static hand gesture recognition for american sign language (asl) in complex background,” *Journal of Signal and Information Processing*, vol. 3, no. 03, p. 364, 2012.
- [24] R. Rokade, D. Doye, and M. Kokare, “Hand gesture recognition by thinning method,” in *Proceedings of the IEEE International Conference on Digital Image Processing*, 2009.
- [25] R. Lionnie, I. K. Timotius, and I. Setyawan, “Performance comparison of several pre-processing methods in a hand gesture recognition system based on nearest

- neighbor for different background conditions,” *Journal of ICT Research and Applications*, vol. 6, no. 3, pp. 183–194, 2012.
- [26] N. Dardas, Q. Chen, N. D. Georganas, and E. M. Petriu, “Hand gesture recognition using bag-of-features and multi-class support vector machine,” in *Proceedings of the IEEE International Symposium on Haptic Audio Visual Environments and Games*, 2010.
- [27] N. B. Ibrahim, M. M. Selim, and H. H. Zayed, “A dynamic skin detector based on face skin tone color,” in *Proceedings of the 8th IEEE International Conference on Informatics and Systems (INFOS)*, 2012.
- [28] N. Ibrahim, M. Selim, and H. Zayed, “An automatic arabic sign language recognition system (arslrs),” *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 4, pp. 470–477, 2018.
- [29] X. Wang, M. Xia, H. Cai, Y. Gao, and C. Cattani, “Hidden-markov-models-based dynamic hand gesture recognition,” *Mathematical Problems in Engineering*, vol. 2012, 2012.
- [30] C. Shan, Y. Wei, T. Tan, and F. Ojardias, “Real time hand tracking by combining particle filtering and mean shift,” in *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.
- [31] G. Kumar and P. K. Bhatia, “A detailed review of feature extraction in image processing systems,” in *Proceedings of the 4th IEEE International Conference on Advanced Computing & Communication Technologies*, 2014.

- [32] A. Tharwat, T. Gaber, A. E. Hassanien, M. K. Shahin, and B. Refaat, “Sift-based arabic sign language recognition system,” in *Proceedings of the Springer Afro-European Conference For Industrial Advancement*, 2015.
- [33] F. Yasir, P. C. Prasad, A. Alsadoon, and A. Elchouemi, “Sift based approach on bangla sign language recognition,” in *Proceedings of the 8th IEEE International Workshop on Computational Intelligence and Applications (IWCIA)*, 2015.
- [34] Z. Xia, Q. Lei, Y. Yang, H. Zhang, Y. He, W. Wang, and M. Huang, “Vision-based hand gesture recognition for human-robot collaboration: A survey,” in *Proceedings of the 5th IEEE International Conference on Control, Automation and Robotics (ICCAR)*, 2019.
- [35] H.-G. Yeom, I.-H. Jang, and K.-B. Sim, “Variance considered machines: Modification of optimal hyperplanes in support vector machines,” in *Proceedings of the IEEE International Symposium on Industrial Electronics*, 2009, pp. 1144–1147.
- [36] J. Belliveau, D. Kennedy, R. McKinstry, B. Buchbinder, R. Weisskoff, M. Cohen, J. Vevea, T. Brady, and B. Rosen, “Functional mapping of the human visual cortex by magnetic resonance imaging,” *Science*, vol. 254, no. 5032, pp. 716–719, 1991.
- [37] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of the Springer COMPSTAT*, 2010, pp. 177–186.
- [38] E. Dogo, O. Afolabi, N. Nwulu, B. Twala, and C. Aigbavboa, “A comparative analysis of gradient descent-based optimization algorithms on convolutional neu-

- ral networks,” in *Proceedings of the IEEE International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, 2018, pp. 92–99.
- [39] P. Zhou and J. Feng, “Understanding generalization and optimization performance of deep cnns,” in *Proceedings of the PMLR International Conference on Machine Learning*, 2018, pp. 5960–5969.
- [40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.
- [42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [43] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

- [45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.
- [46] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [47] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [48] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.
- [49] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.
- [50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

- [51] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.
- [52] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [53] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.
- [54] S. Ghanem, A. Imran, and V. Athitsos, “Analysis of hand segmentation on challenging hand over face scenario.” in *Proceedings of the 12th ACM International Conference on Pervasive Technologies Related to Assistive (PETRA) Environments*, 2019, pp. 236–242.
- [55] S. Zhao, W. Yang, and Y. Wang, “A new hand segmentation method based on fully convolutional network,” in *Proceedings of the IEEE Chinese Control and Decision Conference (CCDC)*, 2018, pp. 5966–5970.
- [56] A. Urooj and A. Borji, “Analysis of hand segmentation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4710–4719.

- [57] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, “Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1949–1957.
- [58] S. Sudhakaran and O. Lanz, “Attention is all we need: Nailing down object-centric attention for egocentric activity recognition,” *arXiv preprint arXiv:1807.11794*, 2018.
- [59] O. K. Oyedotun and A. Khashman, “Deep learning in vision-based static hand gesture recognition,” *Neural Computing and Applications*, vol. 28, no. 12, pp. 3941–3951, 2017.
- [60] R. A. Elsayed, M. S. Sayed, and M. I. Abdalla, “Skin-based adaptive background subtraction for hand gesture segmentation,” in *Proceedings of the International Conference on Electronics, Circuits, and Systems (ICECS)*, 2015.
- [61] G. Poon, K. C. Kwan, and W.-M. Pang, “Occlusion-robust bimanual gesture recognition by fusing multi-views,” *Multimedia Tools and Applications*, pp. 1–20, 2019.
- [62] K. Roy, A. Mohanty, and R. R. Sahay, “Deep learning based hand detection in cluttered environment using skin segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

- [63] M. Arsalan, D. S. Kim, M. Owais, and K. R. Park, “Or-skip-net: Outer residual skip network for skin segmentation in non-ideal situations,” *Expert Systems with Applications*, vol. 141, p. 112922, 2020.
- [64] M. Al-Rousan, K. Assaleh, and A. Tala’a, “Video-based signer-independent arabic sign language recognition using hidden markov models,” *Applied Soft Computing*, vol. 9, no. 3, pp. 990–999, 2009.
- [65] M. Rivera-Acosta, J. M. Ruiz-Varela, S. Ortega-Cisneros, J. Rivera, R. Parra-Michel, and P. Mejia-Alvarez, “Spelling correction real-time american sign language alphabet translation system based on yolo network and lstm,” *Electronics*, vol. 10, no. 9, p. 1035, 2021.
- [66] A. Nagaraj. Image data set for alphabets in the american sign language. [Online]. Available: <https://www.kaggle.com/grassknotted/asl-alphabet>
- [67] S. Sharma, K. Kumar, and N. Singh, “Deep eigen space based asl recognition system,” *IETE Journal of Research*, pp. 1–11, 2020.
- [68] S. Ameen and S. Vadera, “A convolutional neural network to classify american sign language fingerspelling from depth and colour images,” *Expert Systems*, vol. 34, no. 3, p. e12197, 2017.
- [69] N. Song, H. Yang, and P. Wu, “A gesture-to-emotional speech conversion by combining gesture recognition and facial expression recognition,” in *Proceedings of the IEEE First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, 2018.

- [70] A. Dadashzadeh, A. T. Targhi, M. Tahmasbi, and M. Mirmehdi, “Hgr-net: A fusion network for hand gesture segmentation and recognition,” *arXiv preprint arXiv:1806.05653*, 2018.
- [71] B. Xie, X. He, and Y. Li, “Rgb-d static gesture recognition based on convolutional neural network,” *IET Journal of Engineering*, vol. 2018, no. 16, pp. 1515–1520, 2018.
- [72] S. F. Chevtchenko, R. F. Vale, and V. Macario, “Multi-objective optimization for hand posture recognition,” *Expert Systems with Applications*, vol. 92, pp. 170–181, 2018.
- [73] S. Shahriar, A. Siddiquee, T. Islam, A. Ghosh, R. Chakraborty, A. I. Khan, C. Shahnaz, and S. A. Fattah, “Real-time american sign language recognition using skin segmentation and image category classification with convolutional neural network and deep learning,” in *Proceedings of the IEEE TENCON Region 10 Conference*, 2018.
- [74] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th ACM International Conference on Machine Learning*, 2008.
- [75] S. Liu and Q. Xiao, “A signer-independent sign language recognition system based on the weighted knn/hmm,” in *Proceedings of the 7th IEEE International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2015.

- [76] M. Khari, A. K. Garg, R. G. Crespo, and E. Verdú, “Gesture recognition of rgb and rgb-d static images using convolutional neural networks,” *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 5, no. 7, 2019.
- [77] N. Pugeault and R. Bowden, “Spelling it out: Real-time asl fingerspelling recognition,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV)*, 2011, pp. 1114–1119.
- [78] A. A. Barbhuiya, R. K. Karsh, and R. Jain, “Cnn based feature extraction and classification for sign language,” *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 3051–3069, 2021.
- [79] S. Suresh, M. H. TP, and M. Supriya, “Sign language recognition system using deep neural network,” in *Proceedings of the 5th IEEE International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019.
- [80] T. Ozcan and A. Basturk, “Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition,” *Neural Computing and Applications*, vol. 31, no. 12, pp. 8955–8970, 2019.
- [81] H. Birk, T. B. Moeslund, and C. B. Madsen, “Real-time recognition of hand alphabet gestures using principal component analysis,” in *Proceedings of the Scandinavian Conference on Image Analysis*, vol. 1. Proceedings published by various publishers, 1997, pp. 261–268.

- [82] Y. Ye, Y. Tian, M. Huenerfauth, and J. Liu, “Recognizing american sign language gestures from within continuous videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.
- [83] M. R. Abid, E. M. Petriu, and E. Amjadian, “Dynamic sign language recognition for smart home interactive application using stochastic linear formal grammar,” *Transactions on Instrumentation and Measurement*, vol. 64, no. 3, pp. 596–605, 2014.
- [84] M. ElBadawy, A. Elons, H. A. Shedeed, and M. Tolba, “Arabic sign language recognition with 3d convolutional neural networks,” in *Proceedings of the 8th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2017.
- [85] M. A. Rahim, M. R. Islam, and J. Shin, “Non-touch sign word recognition based on dynamic hand gesture using hybrid segmentation and cnn feature fusion,” *Applied Sciences*, vol. 9, no. 18, p. 3790, 2019.
- [86] P. Kumar, P. P. Roy, and D. P. Dogra, “Independent bayesian classifier combination based sign language recognition using facial expression,” *Information Sciences*, vol. 428, pp. 30–48, 2018.
- [87] S. C. Satapathy, N. S. M. Raja, V. Rajinikanth, A. S. Ashour, and N. Dey, “Multi-level image thresholding using otsu and chaotic bat algorithm,” *Neural Computing and Applications*, vol. 29, no. 12, pp. 1285–1307, 2018.

- [88] G. Latif, N. Mohammad, R. AlKhalaf, R. AlKhalaf, J. Alghazo, and M. Khan, “An automatic arabic sign language recognition system based on deep cnn: An assistive system for the deaf and hard of hearing,” *International Journal of Computing and Digital Systems*, vol. 9, no. 4, pp. 715–724, 2020.
- [89] G. Latif, N. Mohammad, J. Alghazo, R. AlKhalaf, and R. AlKhalaf, “Arasl: Arabic alphabets sign language dataset,” *Data in Brief*, vol. 23, p. 103777, 2019.
- [90] E. K. Elsayed and D. R. Fathy, “Sign language semantic translation system using ontology and deep learning,” *International Journal of Advanced Computer Science and Applications*, vol. 11, 2020.
- [91] Y. Saleh and G. Issa, *Arabic Sign Language Recognition Through Deep Neural Networks Fine-tuning*. International Association of Online Engineering, 2020.
- [92] M. A. Bencherif, M. Algabri, M. A. Mekhtiche, M. Faisal, M. Alsulaiman, H. Mathkour, M. Al-Hammadi, and H. Ghaleb, “Arabic sign language recognition system using 2d hands and body skeleton data,” *IEEE Access*, vol. 9, pp. 59 612–59 627, 2021.
- [93] A. Barczak, N. Reyes, M. Abastillas, A. Piccio, and T. Susnjak, *A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures*. Massey University, 2011.

- [94] B. Garcia and S. A. Viesca, “Real-time american sign language recognition with convolutional neural networks,” *Convolutional Neural Networks for Visual Recognition*, vol. 2, pp. 225–232, 2016.
- [95] D. A. Reddy, J. P. Sahoo, and S. Ari, “Hand gesture recognition using local histogram feature descriptor,” in *Proceedings of the 2nd IEEE International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 199–203.
- [96] A. Deza and D. Hasan, *MIE324 Final Report: Sign Language Recognition*, 2018.
- [97] C. Xiao, N. Madapana, and J. Wachs, “One-shot image recognition using prototypical encoders with reduced hubness,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2252–2261.
- [98] A. Ashiquzzaman, H. Lee, K. Kim, H.-Y. Kim, J. Park, and J. Kim, “Compact spatial pyramid pooling deep convolutional neural network based hand gestures decoder,” *Applied Sciences*, vol. 10, no. 21, p. 7898, 2020.
- [99] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.
- [100] Y. S. Tan, K. M. Lim, C. Tee, C. P. Lee, and C. Y. Low, “Convolutional neural network with spatial pyramid pooling for hand gesture recognition,” *Neural Computing and Applications*, vol. 33, no. 10, pp. 5339–5351, 2021.

- [101] S. N. MOHAMMED and H. M. RADA, “English numbers recognition based on sign language using line-slope features and pso-dbn optimization method,” *Journal of Engineering Science and Technology*, vol. 15, no. 3, pp. 1855–1867, 2020.
- [102] H. M. Bui, M. Lech, E. Cheng, K. Neville, and I. S. Burnett, “Using grayscale images for object recognition with convolutional-recursive neural network,” in *Proceedings of the 6th IEEE International Conference on Communications and Electronics (ICCE)*, 2016, pp. 321–325.
- [103] D. Cai, K. Chen, Y. Qian, and J.-K. Kämäräinen, “Convolutional low-resolution fine-grained classification,” *Pattern Recognition Letters*, vol. 119, pp. 166–171, 2019.
- [104] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

VITAE

Personal Details

- **Name:** M Faisal Nurnoby
- **Nationality:** Bangladeshi
- **Date of Birth:** 15/ 09/ 1984
- **Email:** *faisal.noby@gmail.com*
- **Permanent Address:** House: 125, New College Road, Rawshan Mahol, Bad-dah Para, Zip: 8200, Bangladesh.
- **Research Interests:** Artificial Intelligence, Computer Vision, Human Com-puter Interaction, Web Security

Education

- **MSc.** Degree in Computer Science, ICS, KFUPM, Dhahran, KSA, 2021.
- **MBA** Degree in Human Resource Management & Finance, School of Business, North South University, Dhaka, Bangladesh, 2013.

- **BSc.** Degree in Computer Engineering, School of Engineering, North South University, Dhaka, Bangladesh 2007.

Working Experience

- Grader of ICS410, ICS353 and ICS 104, KFUPM, from Feb. 2020 to Apr. 2021
- Instructor at ECE Lab, NSU, from Feb. 2009 to Dec. 2017

Publications

- M. Faisal Nurnoby, El-Sayeed M. El-Alfy, and Hamzah Luqman, "Evaluation of cnn models with transfer learning for recognition of sign language alphabets with complex background", in *Proceedings of the IEEE International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 2020, pp. 1–6.
- M. Faisal Nurnoby and El-Sayeed M. El-Alfy, "Overview and Case Study for Ransomware Classification Using Deep Neural Network," in *Proceedings of the 2nd IEEE Middle East and North Africa COMMunications Conference (MEN-ACOMM)*, 2019, pp. 1–6.
- M. Faisal Nurnoby and El-Sayeed M. El-Alfy, "S3RNet: Hand Gesture and Sign Language Recognition Through Semantic Segmentation" (Under preparation)
- M. Faisal Nurnoby and El-Sayeed M. El-Alfy, "Multi-culture Sign Language Detection and Recognition using Fine-tuned Convolutional Neural Network" (Under preparation)