Project Report

on

Deep Learning based Hand Gesture Recognition of Sattriya Dance Single-Hand Gestures

Submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Electrical Engineering of Assam Science and Technology University

Session: 2022

by

Siddhant Kumar Das (180610003079)
Rishabh Lahkar (180610003070)
Aryamaan Bora (180610003021)
Antariksha K (180610003018)
Ayanmani Das (180610003024)

Under the guidance of

Dr. Amrita Ganguly
Professor
Department of Electrical & Instrumentation Engineering
Assam Engineering College

Department of Electrical & Instrumentation Engineering

Assam Engineering College, Jalukbari, Guwahati, 781013

ABSTRACT

Sattriya Dance or Sattriya Nritriya is a type of Indian classical dance that dates back to the 15th century A.D.. Mahapurusha Sankaradev, a revered Assamese saint and reformer, founded it. The Sattras or Vaishnava monasteries nurtured and preserved the dancing form as it developed over time. Recognition of hand gestures is developing into an excellent tool for numerous applications. Thus, this project aims to use the technological prowess of Hand Gesture Recognition models in recognizing single hand gestures also known as 'Asamyukta Hastas' in Sanskrit where 'asamyukta' means single and 'hastas' mean hand gesture, of Sattriya Dance. A dataset containing 'Asamyukta Hastas' of the Sattriya Dance was created with the help of a Sattriya Dancer. Using this dataset, a pretrained recognition model was trained to recognize the gestures in real-time. A web application was developed for real-time use which implements the whole model and makes it easily accessible.

Certificate From the Supervisor

This is to certify that the project entitled "Deep Learning based Hand Gesture Recognition of Sattriya Dance Single-Hand Gestures" has been carried out and presented by

Siddhant Kumar Das (180610003079)
Rishabh Lahkar (180610003070)
Aryamaan Bora (180610003021)
Antariksha K (180610003018)
Ayanmani Das (180610003024)

students of B.Tech 8th Semester (Electrical Engineering), Assam Engineering College, under my supervision and guidance in a manner satisfactory to warrant its acceptance as a prerequisite for the award of the degree of Bachelor of Technology in Electrical Engineering of the Assam Science and Technology University.

Further, the report has not been submitted/ reproduced in any form for the award of any other degree/ diploma.

Date: 30 June, 2022
Place: Guwahati

Dr. Amrita Ganguly
Professor,
Dept. of Electrical & Instrumentation Engineering
Assam Engineering College
Guwahati - 781013

Certificate From the Head of Department

This is to certify that the project entitled "Deep Learning based Hand Gesture Recognition of Sattriya Dance Single-Hand Gestures" has been been submitted by following students of B.Tech 8th Semester students:

Siddhant Kumar Das (180610003079)
Rishabh Lahkar (180610003070)
Aryamaan Bora (180610003021)
Antariksha K (180610003018)
Ayanmani Das (180610003024)

in partial fulfillment of requirements for the award of the degree of Bachelor of Technology in Electrical Engineering of Assam Science and Technology University.

Date: 30 June, 2022
Place: Guwahati

Dr. Damodar Agarwal
Head of the Department,
Dept. of Electrical & Instrumentation Engineering
Assam Engineering College
Guwahati - 781013

ACKNOWLEDGEMENT

It gives us immense pleasure in bringing out this synopsis of the project entitled 'Deep Learning based Hand Gesture Recognition of Sattriya Dance Single-Hand Gestures'.

Firstly, we would like to express our sincere gratitude to our supervisor Dr. Amrita Ganguly, Department of Electrical Engineering, Assam Engineering College, Guwahati for her invaluable suggestions and constant support throughout the entire duration of the project work. She encouraged us to work on this project. We are also grateful to our college for giving us the opportunity to work with them and providing us the necessary resources for the project.

We would also like to thank Aparna Kalita, our batchmate who helped us with the creation of Sattriya Dataset and all those who helped us to complete this project. We are immensely grateful to all involved in this project as without their inspiration and valuable suggestions it would not have been possible to develop the project within the prescribed time.

Contents

1 Introduction **1**

2 Literature Review **3**

3 Methodology **6**

3.1 Semantic Segmentation of Hand Region 6

3.1.1 Overview 6

3.1.2 Dataset 7

3.1.2.1 Data Pre-processing 8

3.1.3 Network Architecture 8

3.1.3.1 Encoder 9

3.1.3.2 Skip Connections 10

3.1.3.3 Module for increasing the receptive field 12

3.1.3.4 Decoder 12

3.1.4 Training 13

3.2 Hand Gesture Recognition of Sattriya Dance Single Hand Gestures 16

3.2.1 Overview 16

3.2.2 Dataset 16

3.2.3 Network Architecture 18

3.2.4 Training 19

3.3 Web application deployment of Deep Learning model 19

4 Results **22**

4.1 Training of the networks 22

4.2 Results of Hand Segmentation Task 23

4.3 Results of Hand Gesture Recognition Task 27

5 Conclusion and Suggestions **31**
5.1 Limitations 32
5.2 Future Scope 32

Bibliography **33**

List of Figures

1.1 Sattriya Dance 2

3.1 Two stage hand gesture recognition model for recognition of single hand Sattriya dance hand gestures 7
3.2 OUHANDS dataset sample 8
3.3 Network Architecture 9
3.4 MobileNetV2 Architecture 10
3.5 Convolutional Block Attention Module (CBAM) 11
3.6 CAM 11
3.7 SAM 12
3.8 ASPP 13
3.9 Decoder Block 14
3.10 Dataset Creation Block Diagram 17
3.11 Sattriya dataset sample 17
3.12 Streamlit 20
3.13 WebRTC 20
3.14 Streamlit's execution model 21
3.15 Web Application 21

4.1 Training Plots 23
4.2 Example Segmentations on OUHANDS Test Set (a) Input Images (b) Ground Truth Segmentation Mask (c)Mobilenetv2 Linknet (d)Deeplabv3+ (e)Mobilenetv2 Linknet + CBAM (f) Mobilenetv2 Linknet + ASPP 26
4.3 Confusion matrix on Ouhands testset 28
4.4 Confusion matrix on Sattriya testset 29

List of Tables

3.1 Structure of recognition stage CNNs 18

4.1 Convergence of Training IoU Score and Loss of the proposed methods . . . 23
4.2 Performance of models with or without ImageNet pretrained encoders . . . 24
4.3 Performance of different proposed methods 24
4.4 Performance comparison with other segmentation methods 25
4.5 Parameters and GFLOPS comparison of the methods 27
4.6 Recognition performance comparison of the methods 28
4.7 Classification Report 29
4.8 Classification Report 30

List of Abbreviations

HGR Hand Gesture Recognition

HCI Human Computer Interaction

RGB-D Red Green Blue Depth

TPU Tensor Processing Unit

GPU Graphics Processing Unit

CPU Central Processing Unit

ANN Artificial Neural Network

CNN Convolutional Neural Network

RNN Recurrent Neural Network

ResNet Residual Network

ASPP Atrous Spatial Pyramid Pooling

AG Attention Gate

CBAM Convolutional Block Attention Module

IEEE Institute of Electrical and Electronics Engineers

ISBI International Symposium on Biomedical Imaging

LSTM Long Short Term Memory

CAM Channel Attention Model

SAM Spatial Attention Model

GAP Global Average Pooling

GMP Global Max Pooling

RELU Rectified Linear Unit

IoU Intersection Over Union

mIoU Mean Intersection Over Union

WebRTC Web Real Time Communication

JS Javascript

FLOPS Floating Point Ops Per Second

Chapter 1

Introduction

Sattriya Dance or Sattriya Nritya is a major Indian Classical Dance form introduced in the 15th century A.D. by the great Vaishnava Saint of Assam, Mahapurusha Shankardeva. It was used as a medium of propagation of the Vaishnava faith in the region. The Sattras or Vaishnava monasteries have nourished and preserved this dance form for ages, which has given it a unique flair making it a gem of Assamese culture. In Karuna Borah's book 'Sattriya Nrittyar Rup Darshan'[1] it was stated that single-hand gestures and double-hand gestures are used to perform Sattriya dance. The single-hand gestures are known as 'Asamyukta hastas' and doublehand gestures are divided into two parts: 'samyukta hastas' and 'hastas nrittya'. Like normal hand gestures[2], Sattriya hand gestures are expressive and meaningful. Interaction between the dancers is achieved by the hand gestures.

With the increasing popularity of the internet and smartphones, it would be easy to identify hand gestures of the Sattriya dance using computers and smartphones. As no significant work was done in this field , it was necessary to apply contemporary technologies to identify this dance form. This would help popularise Sattriya Dance internationally and raise awareness of the rich and vibrant culture of Assam while also preserving it. A suitable method to recognise hand gesture is by using hand gesture recognition models which are based on Deep Learning Models as hand gestures are an integral part of the dance. An RGB camera input is captured, processed by the model, and then the hand action is recognized. This was implemented by developing a hand gesture recognition model which could segment hand region from any image after which recognition was done. The model was initially trained on the Ouhands dataset[3] which is a standard dataset for HGR.

Due to the lack of a Sattriya single hand dataset, we developed our own by taking numerous pictures of different hand gestures made by a skilled Sattriya dancer. Using this dataset, we trained our pretrained HGR model to recognise the different gestures. The

Figure 1.1: Sattriya Dance

model could recognise with a good degree of accuracy. To make this project easily accessible, a cloud-based web application was developed in which the HGR model is stored on a cloud server. The web application uses the camera available on the device and thus recognises the hand gestures. The project work has been documented in the upcoming sections.

Chapter 2

Literature Review

The problem of Hand Gesture Recognition (HGR) is as old as Human Computer Interaction (HCI) Systems themselves. It is one of the most challenging tasks related to HCI. Due to differences in illumination, background, shadows, differences in ethnicities, etc. the task becomes even more complicated. Hand gestures have a wide use case. They are used to convey extra information during conversations, like when pointing at an object, the index finger is used, and also, they help add structure and emphasis to the words being spoken, by blind, deaf, and dumb people as a means of communication. Using gestures can help increase the amount of information being transmitted by 60%. The secret behind communicating effectively is to use more hand gestures.

From [4], Bhuyan et al showed the raw data acquisition of hand gestures as either sensor-based or vision based. Vision based gestures [5] have become more popular due to the limitations of sensor-based approaches, like requirement of gloves, which are expensive devices. The gestures have been obtained by the use of special cameras. It also has to be noted that the majority of problems related to hand gesture segmentation, like variations in illumination, background, occlusion, etc. can be solved by the use of depth cameras for capturing image RGB-D data. But, the feasibility and cost [6] of depth cameras make them a rather unpopular choice.

Before the era of deep neural networks, classical machine learning techniques like Support Vector Machines [7], k-Nearest Neighbours [8], Artificial Neural Networks [9], etc. were mainly used for image classification and recognition. Also, [10], [11], [12], use a variety of methods to capture RGB-D data, i.e., depth information.

With the advancement of powerful graphics processing units [13], and recently Tensor Processing Units (TPUs) [14], deep neural networks have been the standard for image and video related problems. [13] was also the first paper that showed the enormous advantages

of using GPUs over multicore CPUs. The GPUs parallel processing capabilities reduce the time taken to train deep neural networks significantly. In some instances, they outperform CPUs massively (almost 50-100 times faster).

In 2012, the error rate of the ImageNet Classification Challenge dropped significantly. This was mainly due to the use of deep neural networks. AlexNet [15] was the first model which incorporated deep neural networks in this challenge. Furthermore, in [16], residual networks are introduced. These differ from plain networks. These ResNets, for short have shortcut skip connections connecting different layers. The ResNets perform much better than the networks without skip connections. They are easily optimizable and have a far lower error rate. In [17], DeepLab network is introduced. The authors have used Atrous Spatial Pyramid Pooling (ASPP) to increase the receptive field of the network.

In Dadashzadeh et al [18]. proposed a hand gesture recognition model incorporating [16] and [17]. The network consists of two streams, the first called the shape stream and the second called the appearance stream. The input to the shape stream is the segmentation map, which had been extracted in the previous stage using ResNet and ASPP. The streams are subsequently fused to recognize the gesture.

Next, the encoder-decoder module is introduced in [19]. The network is called UNet. The encoder module is a contracting path to capture context, while the decoder module is a symmetric expanding path for enabling precise localization. This model outperformed the state-of-the-art in the ISBI Challenge. In [20], Attention Gates (AG) were introduced. These gates were easily integrated into the UNet architecture, creating Attention UNet. The advantage of using AGs were that it helped suppress irrelevant information while highlighting important parts of an image. It also increased the sensitivity of the model and prediction accuracy.

To tackle the problem of dynamic gesture recognition, MobileNetV2 is introduced in [21]. It uses depthwise separable convolution, residual connections with linear bottlenecks, and an inverted residual structure. These novelties make MobileNetV2 an effective option for use in scenarios where resources are limited.

For efficient semantic segmentation in real-time, LinkNet [22], was developed. In LinkNet, the spatial information from the encoder to the decoder is directly bypassed. This reduces the processing time of the network, and at the same time increases accuracy.

Furthermore, attention can also be added to these networks. The Convolution Block Attention Module (CBAM) [23], has been developed for this exact purpose. It consists of two modules. The Channel Attention Module which focuses on ‘what’ is important in an

image, while The Spatial Attention Module, focuses on 'where' the important part of an image is. These two modules combine together to determine what information to suppress and highlight. The CBAM module is a lightweight module which can be integrated into standard deep neural networks.

Another class of Neural Networks is Recurrent Neural Networks (RNN) [24]. In RNNs, there are feedforward paths using which the outputs of previous operations can be fed into future operations as input. Even though [24] is not the original RNN paper, it shows the mathematical background behind RNN using signal processing. But, RNNs suffer from a recency bias, where it gives more accurate results for recent information (short term) than older information (long term). To solve this issue another algorithm is derived called the Long Short Term Memory (LSTM) Algorithm [24]. LSTMs can work for long term memory predictions with far greater accuracy than standard RNNs. This is achieved by using special units which are used to enforce constant error flow throughout the network. The problem of vanishing and exploding gradients is also solved by this algorithm.

Chapter 3

Methodology

The structure of the hand gesture recognition model for recognition of single hand sattriya dance hand gestures is shown in (Fig. 3.1). The pipeline consists of two stages employing three deep CNN networks for two separate tasks. The first stage of the gesture recognition pipeline is semantic segmentation of the hand region from the RGB input image captured by the camera (Section 3.1). The output mask obtained from the segmentation model is passed as an input to the second stage of the gesture recognition pipeline. The second stage is composed of two streams, one for the input RGB image and one for the segmentation map from the first stage. Each of the streams consists of a deep CNN which is converged in a fully-connected layer and a softmax classifier. The output obtained is the classification label of the gesture demonstrated by the user. The training of the network is done in a stage-wise manner.

3.1 Semantic Segmentation of Hand Region

3.1.1 Overview

The task of segmenting regions of interest (hand postures) from a given RGB input frame using deep neural network methods can be categorized into three extensive steps.

- Step 1: Collection and preprocessing of dataset.
- Step 2: Design and training of semantic segmentation deep learning model.
- Step 3: Proposed deep learning model statistical evaluation and output generation.

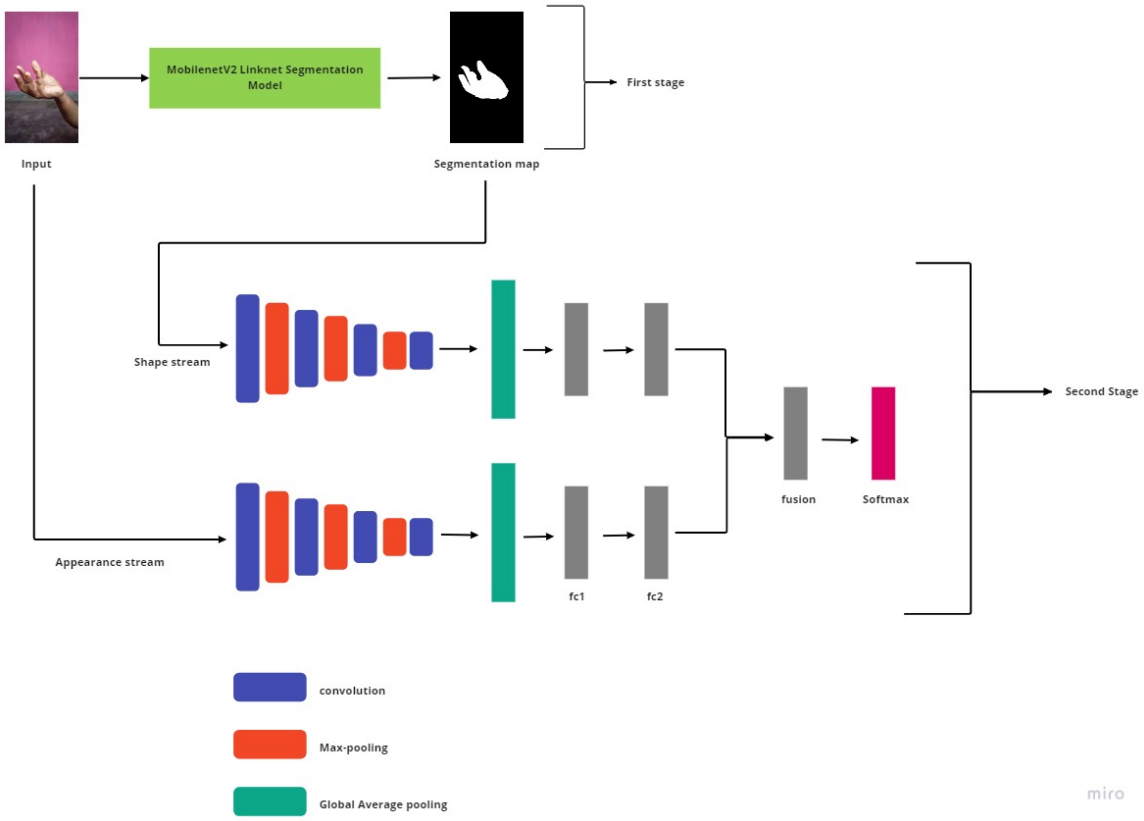

Figure 3.1: Two stage hand gesture recognition model for recognition of single hand Sattriya dance hand gestures

3.1.2 Dataset

The dataset used in the first stage of the work i.e., the segmentation task is the "OUHANDS" dataset. The dataset is a publicly available collection of static hand posture images captured while a user is demonstrating gesture commands to a hand-held device [3]. The authors used "The Intel RealSense F200" camera to collect the dataset samples. The dataset consists of 3150 hand samples (2150 training samples and 1000 test samples) along with their corresponding binary mask. The resolution for all the images is 640*480 pixels. The OUHANDS dataset comprises of 10 unique hand postures demonstrated by 23 people. The dataset is captured in Human Computer Interaction (HCI)–like settings, such as handheld camera position, uncontrolled backgrounds, variation in illumination, hand-face obstruction with different hand shapes and sizes. This makes this dataset conducive for training and testing Human Computer Interaction (HCI) methods.

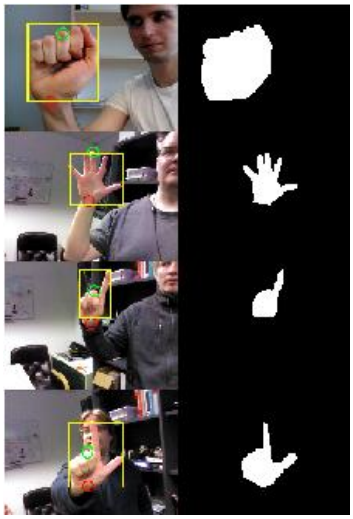

Figure 3.2: OUHANDS dataset sample

3.1.2.1 Data Pre-processing

The dataset samples were originally of the size 640*480 pixels. This was resized into 320*320 pixels for training the semantic segmentation model. The ground truth data which were the segmentation masks were binary thresholded before feeding into the models.

3.1.3 Network Architecture

The network comprises of a U-shaped decoupled encoder-decoder structure similar to the architecture of U-NET [19], The network consists of the encoding or contracting path, the decoding or expanding path and there is a section of layers that link the two paths to minimize loss of spatial information during the decoding stage. The architecture of the proposed model for segmenting hand regions is represented in Fig 2. The encoder is the narrowing contracting path of the architecture which encodes the input features from the given RGB frame. Lightweight MobileNetV2 [21] feature extraction model was employed to modify the encoder. The feature extractor was pre-trained on ImageNet in order to benefit from regularization induced by transfer learning. The decoder is the expanding path of the architecture whose goal is to project the extracted lower dimensional features by the encoder into a higher resolution spatial dimension pixel space by multiple upscaling operations. The pixel resolution of the final output map is (320 X 320), identical to the original input RGB image. The decoder architecture was adopted from LinkNet [22] network architecture which utilizes the skip connections in an efficient manner conducive for real time

application. The skip connections in the proposed model architecture links the encoder and decoder at four points.

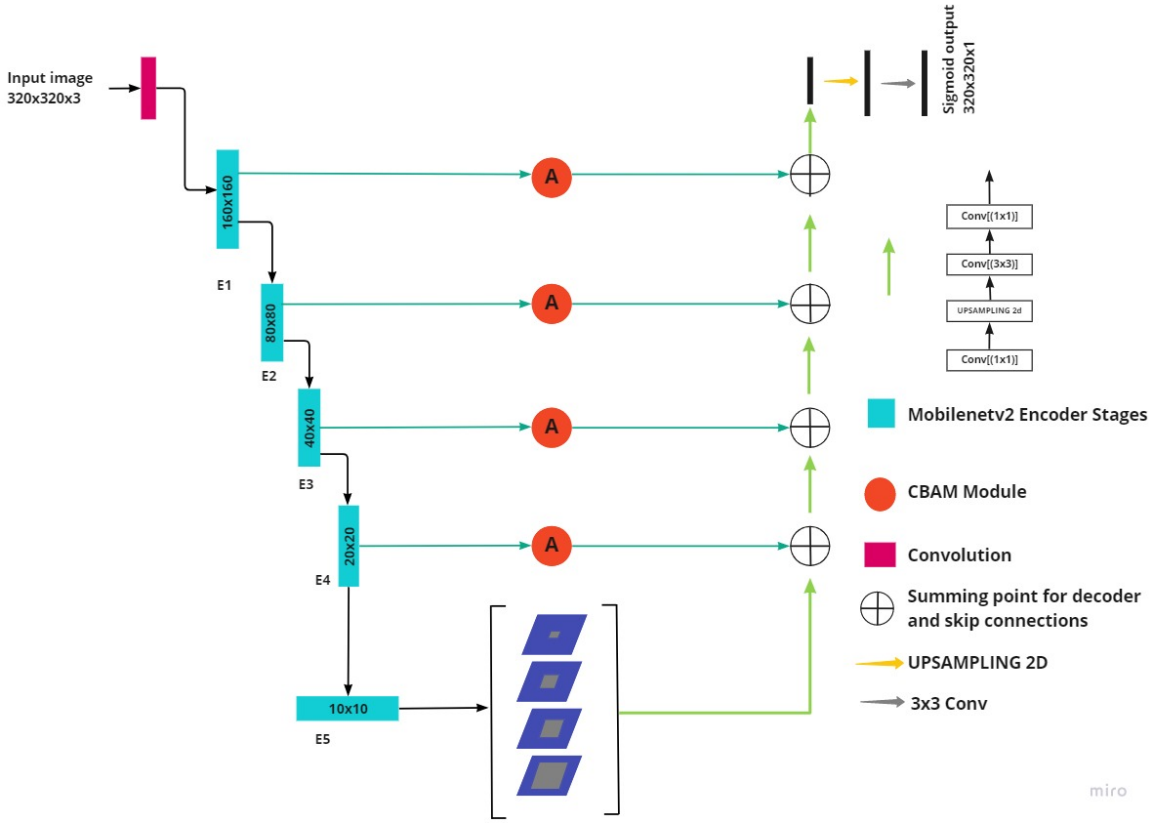

Figure 3.3: Network Architecture

3.1.3.1 Encoder

To encode the semantic information from the given hand pose image, the model employs the MobileNetV2 [21] model for contextual feature extraction. The architecture for MobileNetV2 feature extraction model is shown in Figure 3.4.

For end-to-end segmentation tasks, the fully connected layers of the architecture are omitted. MobileNetV2 uses depth-separable convolution, residual links with linear bottlenecks, and an inverted residual structure.

Depth-separable convolution is used extensively in real-time tasks for two reasons

- (i) Compared to the classic convolution, fewer parameters need to be tuned, minimizing the likelihood of overfitting the model.

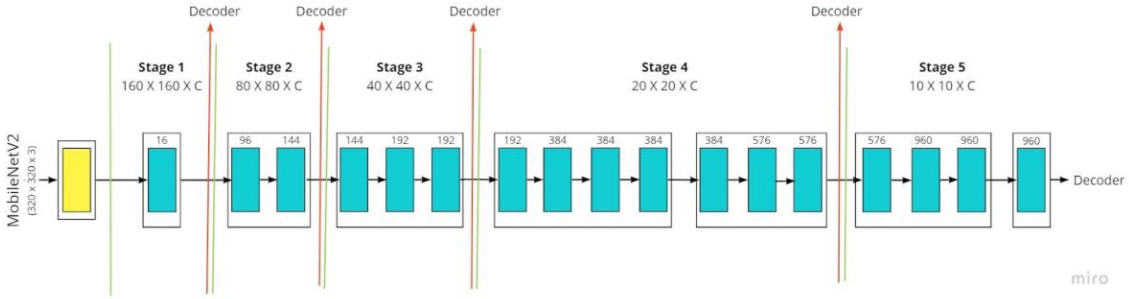

Figure 3.4: MobileNetV2 Architecture

- (ii) Fewer calculations are performed, making them computationally efficient and more conducive for real-time applications.

Depthwise convolution constitutes convolution blocks called “inverted residual blocks” in MobileNetV2. There are three layers in each block. The first layer in each block is a 1×1 convolution layer with the ReLU6 activation function, which expands the number of feature channels. The second layer is a 3x3 depthwise convolution layer. The third 1x1 convolution layer compresses the network back to the original number of channels. Inverted residual blocks squeeze the layers where skip connections are linked, degrading network performance. To overcome this, the authors introduced the linear bottleneck architecture, where the last convolution of a residual block has a linear output before adding it to the initial activations.

The input to the encoder is a RGB image with (320 X 320 X 3) dimensions. Feature maps are extracted by the encoder blocks and are repeatedly downsampled during the process. The highest level feature map obtained during the encoding stage is of the dimension (10 X 10 X 1280).

3.1.3.2 Skip Connections

Skip connections link the higher level layer from the deeper part of the architecture to the lower level layer of the decoder for fine-grained feature re-usability during the up-scaling operations. The deeper layers of the network from the encoder have rich feature information. Skip connections make the model more robust by combining these rich feature maps to the decoder blocks for effective localization. In the proposed model the skip connections are modified by incorporating Convolutional Block Attention Module (CBAM). The architecture of CBAM is shown in Fig 3.5. Applying an attention block before in the skip

connection allows for the network to put more weight on the features of the skip connection that will be relevant for adaptive feature refinement. As CBAM is a lightweight and general module it can be integrated into the network seamlessly with negligible overheads.

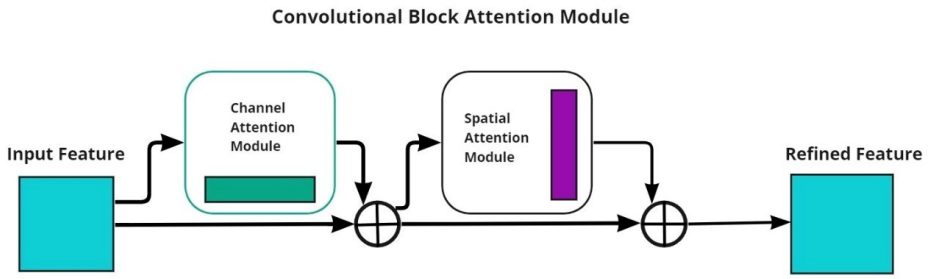

Figure 3.5: Convolutional Block Attention Module (CBAM)

CBAM contains two sequential sub-modules called the Channel Attention Module (CAM) and the Spatial Attention Module (SAM). Channel Attention Module decomposes the input tensor into 2 subsequent vectors of dimensionality (c × 1 × 1). One of these vectors is generated by Global Average Pooling (GAP) while the other vector is generated by Global Max Pooling (GMP). Average pooling is mainly used for aggregating spatial information, whereas max pooling preserves much richer contextual information in the form of edges of the object within the image which thus leads to finer channel attention. Simply put, average pooling has a smoothing effect while max pooling has a much sharper effect, but preserves natural edges of the objects more precisely.

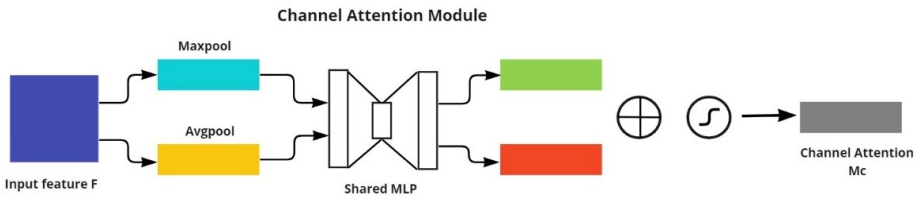

Figure 3.6: CAM

Spatial Attention Module (SAM) is a three-fold sequential operation. The first part of it is called the Channel Pool, where the Input Tensor of dimensions (c × h × w) is decomposed to 2 channels, i.e. (2 × h × w), where each of the 2 channels represent Max Pooling and Average Pooling across the channels. This serves as the input to the convolution layer which outputs a 1-channel feature map, i.e., the dimension of the output is (1 × h × w).

Thus, this convolution layer is a spatial dimension preserving convolution and uses padding to do the same. The output is then passed to a Sigmoid Activation layer. Sigmoid, being a probabilistic activation, will map all the values to a range between 0 and 1. This Spatial Attention mask is then applied to all the feature maps in the input tensor using a simple element-wise product.

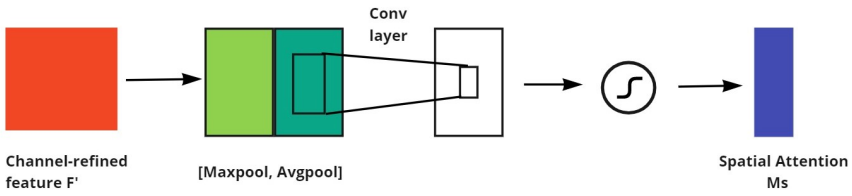

Figure 3.7: SAM

3.1.3.3 Module for increasing the receptive field

Contextual information has been shown to be extremely important for semantic segmentation tasks. The model takes advantage of a spatial pyramid pooling module to capture and aggregate multi-scale contextual information. Different scales of contextual information are combined using an Atrous Spatial Pyramid Pooling (ASPP) [17] module as shown in Fig 3.8. This kind of module has been employed successfully in the state-of-the art DeepLabv3+ for semantic segmentation. The ASPP module blends representations at different levels of convolutional features and thus enlarges the receptive field without sacrificing spatial resolution. The ASPP module used in this work has five levels, an image pooling layer, one 1 X 1 convolution and three 3 X 3 convolutions with atrous rates of 6, 12 and 18 respectively (all with 64 filters). The resulting five feature maps are concatenated together and the concatenated output is fed to 1 X 1 convolution. This module is incorporated at the bottom of the network architecture and acts as the bridge between encoder and decoder.

3.1.3.4 Decoder

In the decoder, the expanding mechanism of the Link-NET [22] architecture is employed. Link-NET is a fully convolutional network, structured in a U-shape similar to the U-NET. The Link-NET varies from the U-NET by the mechanism by which it links each encoder with its corresponding decoder block through the skip connections. Unlike U-NET, the decoder of the Link-NET does not concatenate the feature map extracted from the lower level

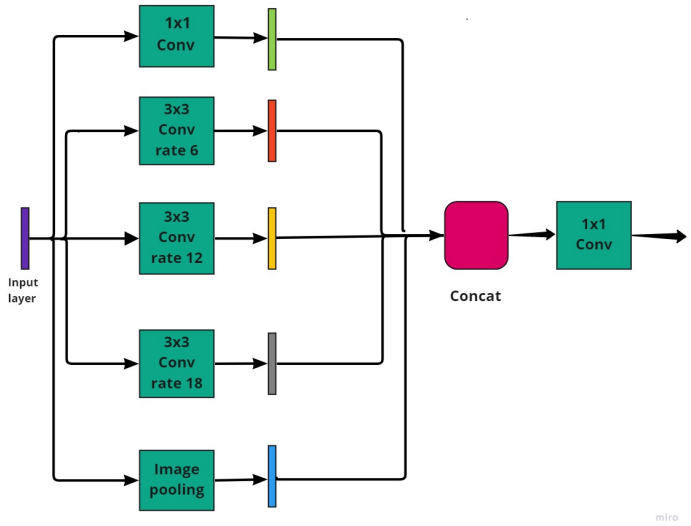

Figure 3.8: ASPP

encoder path and the higher level feature map from the deeper layers. The feature maps are added directly (through a convolutional layer) to reduce the computational parameters. The decoder produces segmentation maps at full resolution. The expanding structure constitutes four decoder blocks stacked together each linked to its corresponding encoder block through a summing point and skip connection. The decoder block structure is shown in fig 4. Each block has three layers. First layer is a (1 X 1) convolution layer with batch normalization and RELU activation. Second layer is an upsampling layer which upsamples the feature map using bilinear interpolation. It is then followed by a two convolution layers. The final decoder block of the expansion path is followed by (3 X 3) convolution layer with the sigmoid activation which outputs the required (320 X 320 X 1) segmentation mask of region of interest (hand postures).

3.1.4 Training

The training data was used to train six different models. Each of the six models was subjected to identical training. Images having a resolution of 640 x 480 pixels were included in the training data which were then scaled to a dimension of 320 X 320 pixels and fed into the network to train the model. While experimenting with various batch sizes, it was experimentally determined that batch size of 16 provided the optimal learning pattern. ”Adam” was used as the optimizer to train the deep learning model. According to [25], the method

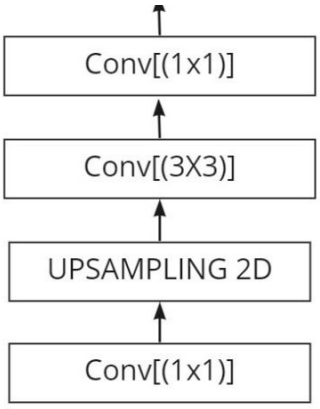

Figure 3.9: Decoder Block

is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

The set argument values for the optimizer used for training the models :

- Learning Rate: 0.001
- Exponential Decay Rate for the 1st moment estimates ($\beta 1$) : 0.9
- Exponential Decay Rate for the 2nd moment estimates ($\beta 2$): 0.999
- Epsilon value, ϵ : 1e-4

All the models were trained for 500 epochs.

- **Loss Function** : After experimenting with many loss functions it was decided to use the dice coefficient loss [26] for the task of semantic segmentation.

$$D = \frac{2\sum_i^N y_i x_i}{\sum_i^N y_i^2 + \sum_i^N x_i^2} \tag{3.1}$$

Here, y_i and x_i are a representation of pixel values of corresponding predicted mask and the ground truth mask, respectively. The values of y_i and x_i are either 0 or 1. The denominator in the dice loss equation represents the sum of total pixels from both the predicted mask and the ground truth mask. The numerator is a representation of successfully predicted pixels, since the sum only advances when y_i and x_i match (both of value 1). If two masks completely overlap or the predicted output is accurate,

the Dice Coefficient gets its maximum value to 1. Otherwise, the Dice coefficient starts to decrease, getting to its minimum value to 0 if the two masks don ‘t overlap at all or the predicted output is totally inaccurate.

- **Intersection Over Union metric (IoU)** : To quantitatively evaluate the model performance, the IoU metric commonly known as the Jaccard Index is used. It is a commonly used metric for evaluating segmentation performance. It measures the percentage overlap between the ground truth mask and the predicted mask. The value of the IoU score is in the range of 1 to 0. A good prediction will result in a higher IoU score whereas a poor prediction will result in a lower IoU score. In mathematical terms, it is defined as the number of pixels that are common between the ground truth mask and predicted mask divided by the total number of pixels present in both of the masks,

$$IoU = \frac{(GroundTruthMaskPixels) \bigcap (PredictedMaskPixels)}{(GroundTruthMaskPixels) \bigcup (PredictedMaskPixels)} \tag{3.2}$$

  Mean IoU is the average IoU score of all the individual image segmentation in the dataset. This is used to evaluate and compare the various models.

- **F1-score** : F1 score also known as F score or F-measure, is the weighted average of precision and recall. It can also be interpreted as harmonic mean of the precision and recall where it can reach a best value of 1 and worst value of 0. If the value is 1 then it shows perfect precision and recall. If either of them is 0 then it shows the lowest possible value i.e., 0. It is primarily used to compare the performance of two classifiers. For example, we have two classifiers A and B. If the classifier A has higher recall and B has higher precision, then the F1 score can be used to determine which of them produces better results.

$$f1 - Score = 2(P * R)/(P + R) \tag{3.3}$$

  Where, P is for precision and R is for recall.

For deep-learning implementations, Keras and Tensorflow frameworks were used. The system was configured with the NVIDIA P100 GPU.

3.2 Hand Gesture Recognition of Sattriya Dance Single Hand Gestures

In this stage of the recognition model pipeline, the hand gesture labels are predicted using a fusion of two CNNs.

3.2.1 Overview

The task of recognising hand gestures and predicting the labels

- Step 1: Creation of Sattriya Single Hand Dataset.
- Step 2: Design and training of recognition deep CNNs.
- Step 3: Deep learning model statistical evaluation and output generation.

3.2.2 Dataset

For the second stage, recognition of Sattriya Hand Gestures there were no publicly available datasets for use, a dataset had to be created to train and test the model. A 12.0 MP OnePlus mobile device camera - OnePlus GM1901 f/1.8 1/25 4.74mm ISO2000 was used to capture the images of the gestures. A total of 1100 images of 10 gestures were captured. For the training set, each gesture has 20 images each of 5 subjects, i.e., a total of 100 images per gesture. The test set is made up of 100 random images from the same 5 subjects, each containing 10 photos of each gesture. The photos were taken with a monochromatic background. Before the labelling of the images, the images were cropped and resized to a fixed dimension of 640x480 pixels (4:3 Aspect Ratio). The dataset includes images of the particular hand gestures that are shown by the dancers to depict the various stories of folklore. The images were accordingly labelled with their respective meanings using a labelling tool known as ImageJ.

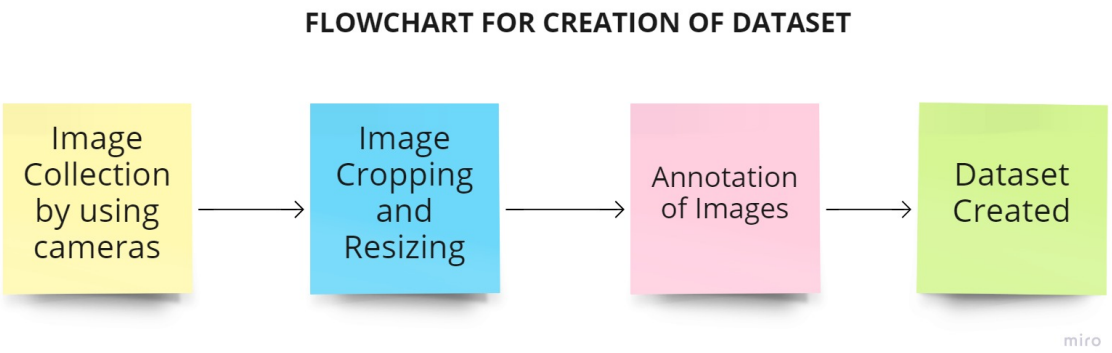

Figure 3.10: Dataset Creation Block Diagram

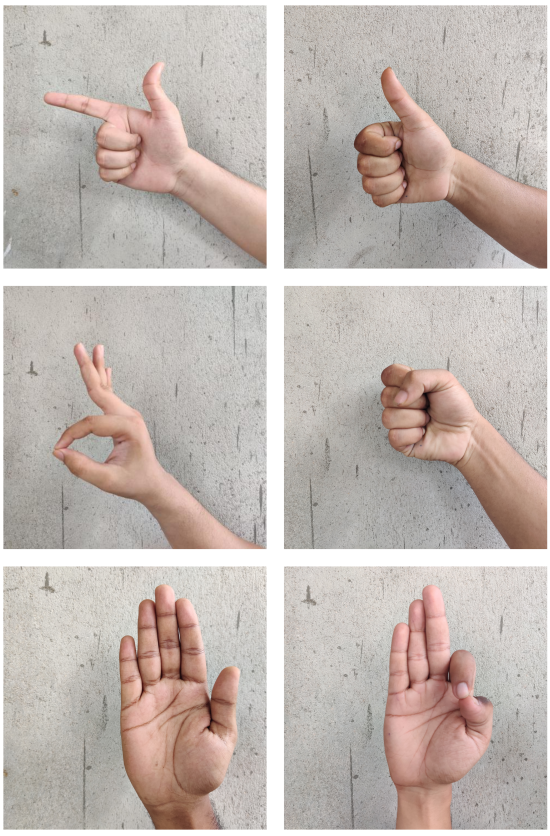

Figure 3.11: Sattriya dataset sample

3.2.3 Network Architecture

In the second stage of the hand gesture recognition model, the gesture labels are predicted using a fusion of two CNNs. As shown in 3.1, this stage consists of two separate CNNs with the same architecture where each network exploits the shape-based and appearance-based information respectively, conveyed by the hand segmentation map from the previous stage and the RGB image for robust hand gesture recognition.

The detailed structure of the CNN is represented in Table 3.1

Table 3.1: Structure of recognition stage CNNs

Layer	**Type**	**Output Shape**
input		320 × 320 × 3
conv1	convolution	318 × 318 × 16
pool1	max-pooling	106 × 106 × 16
conv2	convolution	104 × 104 × 32
pool2	max-pooling	34 × 34 × 32
conv3	convolution	32 × 32 × 64
pool3	max-pooling	10 × 10 × 64
conv4	convolution	8 × 8 × 128
pool4	max-pooling	128
dropout1	dropout	128
fc1	fully connected	64
dropout2	dropout	64
fc2	fully connected	64

Appearance stream and shape stream, both CNNs have identical architecture, except that the shape stream CNN uses a one-channel input image. This is followed by a (3X3) convolution operation. The convolution operation is followed by a max pooling operation which downscales the feature map into (106 * 106 * 16) spatial resolution. The activation function used to fire the neurons is RELU. Similar convolution max-pooling operations are carried out as depicted in Table 3.1. This is followed by a global average pooling and dropout. The neurons are finally converged in two fully-connected layers at the end of the network.

A fusion function is then deployed to fuse the outputs of the last fully-connected layer of each network in an element-wise summation manner. Let $f^S \epsilon R^d$ and $f^A \epsilon R^d$ be the

outputs of the last fully connected layers (fc2) from the first (shape stream) and second (appearance stream) CNN respectively. The output of the fusion function is

$$f_i(sum) = f_i^S + f_i^A \tag{3.4}$$

where 1 $\leq$ i $\leq$ d and $f_i(sum)$ ϵR^d , and d is the number of the neurons (64 for fc2). The feature vector $f_i(sum)$ is then fed into a softmax classifier for joint supervised learning.

3.2.4 Training

Adam was used as the optimizer to train the deep learning model. According to [25], the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

The set argument values for the optimizer used for training the models :

- Learning Rate: 0.001
- Exponential Decay Rate for the 1st moment estimates (β1) : 0.9
- Exponential Decay Rate for the 2nd moment estimates (β2): 0.999
- Epsilon value,ϵ : 1e-4

Loss Function : After experimenting with many loss functions it was decided to use the binary cross entropy loss [26] for the task of recognition of gestures and prediction of labels.

$$Loss = -(y \log(p) + (1 - y) \log(1 - p)) \tag{3.5}$$

3.3 Web application deployment of Deep Learning model

A web based application has been developed for real time implementation of the hand gesture recognition model for recognition of Single Hand Sattriya Dance Hand Gestures, which can process frames taking video input directly from readily available devices of the user such as webcams or smartphones. The application is built using Streamlit WebRTC.

WebRTC : WebRTC (Web Real-Time Communication) (fig3.12) enables web servers and clients, including web browsers, to send and receive video, audio, and arbitrary data

streams over the network with low latency.WebRTC extends Streamlit's powerful capabilities to transmit video, audio, and arbitrary data streams between frontend and backend processes, like browser JavaScript and server-side Python.

It is now supported by major browsers like Chrome, Firefox, and Safari, and its specs are open and standardized. Browser-based real-time video chat apps like Google Meet are common examples of WebRTC usage.

Figure 3.12: Streamlit

Figure 3.13: WebRTC

Streamlit : Streamlit (fig 3.13) is an open-source python framework for building web apps for Machine Learning and Data Science. Web apps can be deployed easily using Streamlit. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

- Upon each execution, the Python script is executed from top to bottom
- Each execution of the Python script renders the frontend view, sending data from Python to JS as arguments to the component.
- The frontend triggers the next execution sending data from JS to Python as a component value.

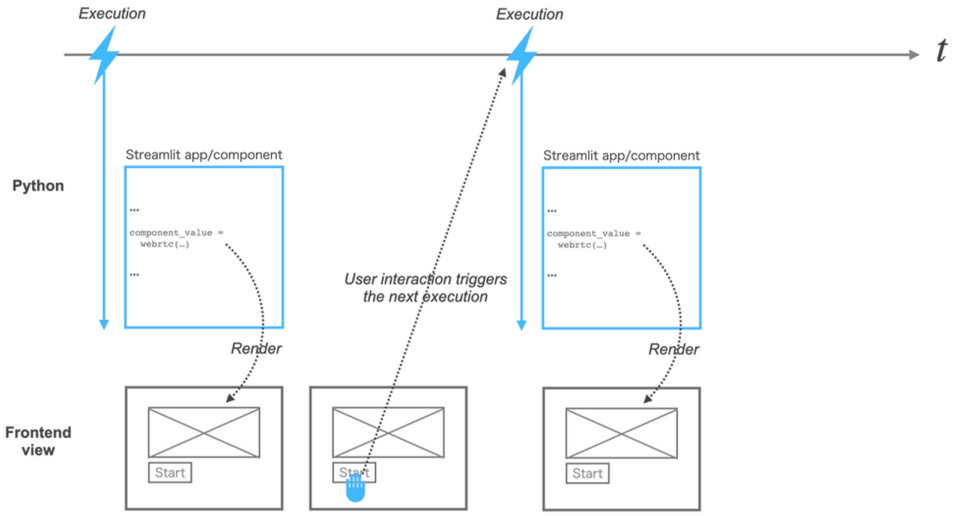

Figure 3.14: Streamlit's execution model

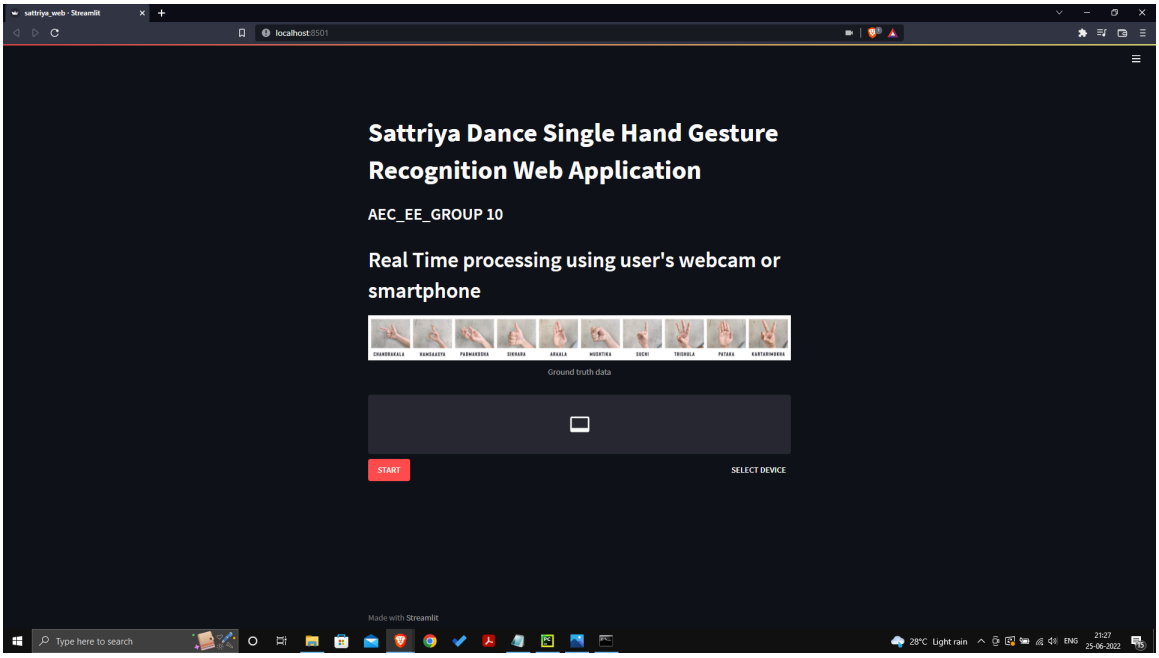

Figure 3.15: Web Application

Chapter 4

Results

4.1 Training of the networks

Evaluation of the models during the training phase was based on the convergence of the Intersection Over Union score. Table 4.1 summarizes the convergence of the proposed methods during the training. It is observed that the three proposed methods had a similar rate of convergence of the IoU score. With a low IoU score at the first epoch, the IoU score increased rapidly in all the three methods with the base method reaching the highest score of 0.6819 after 10 epochs, followed by Proposed MobileNetv2 LinkNet+ASPP and Proposed MobileNetv2 LinkNet+CBAM with IoU score of 0.6750 and 0.5043 respectively. The rate of convergence of all the methods decreases significantly after 100 epochs. After 500 epochs of training, Proposed MobileNetv2 LinkNet+ASPP achieved the highest IoU score of 0.9841 among the three. The dice coefficient loss function decreases at a high rate initially but the rate slows down at higher epochs. After 300 epochs, the rate of convergence of the loss function reduces greatly and by the end of the 500 epochs, the Proposed MobileNetv2 LinkNet+CBAM recorded the lowest loss function value (0.0078) followed by Proposed MobileNetv2 LinkNet+ASPP (0.0080) and Proposed MobileNetv2 LinkNet (0.0081). The Training and Validation IoU score and Loss function plot is shown in fig 4.1 along with the training accuracy plot.

Table 4.1: Convergence of Training IoU Score and Loss of the proposed methods

Epoch	MobileNetv2 LinkNet		MobileNetv2 LinkNet+ASPP		MobileNetv2 LinkNet+CBAM	
	Training IoU	Training Loss	Training IoU	Training Loss	Training IoU	Training Loss
1	0.2045	0.6600	0.1947	0.6548	0.2127	0.6772
10	0.6819	0.1897	0.6750	0.1946	0.5043	0.3307
30	0.9200	0.0417	0.9192	0.0421	0.8681	0.0707
50	0.9566	0.0222	0.9560	0.0225	0.9421	0.0298
100	0.9710	0.0147	0.9707	0.0149	0.9692	0.0156
200	0.9759	0.0122	0.9759	0.0122	0.9756	0.0123
300	0.9787	0.0108	0.9795	0.0104	0.9792	0.0105
400	0.9813	0.0094	0.9823	0.0090	0.9823	0.0089
500	0.9839	0.0081	0.9841	0.0080	0.9832	0.0078

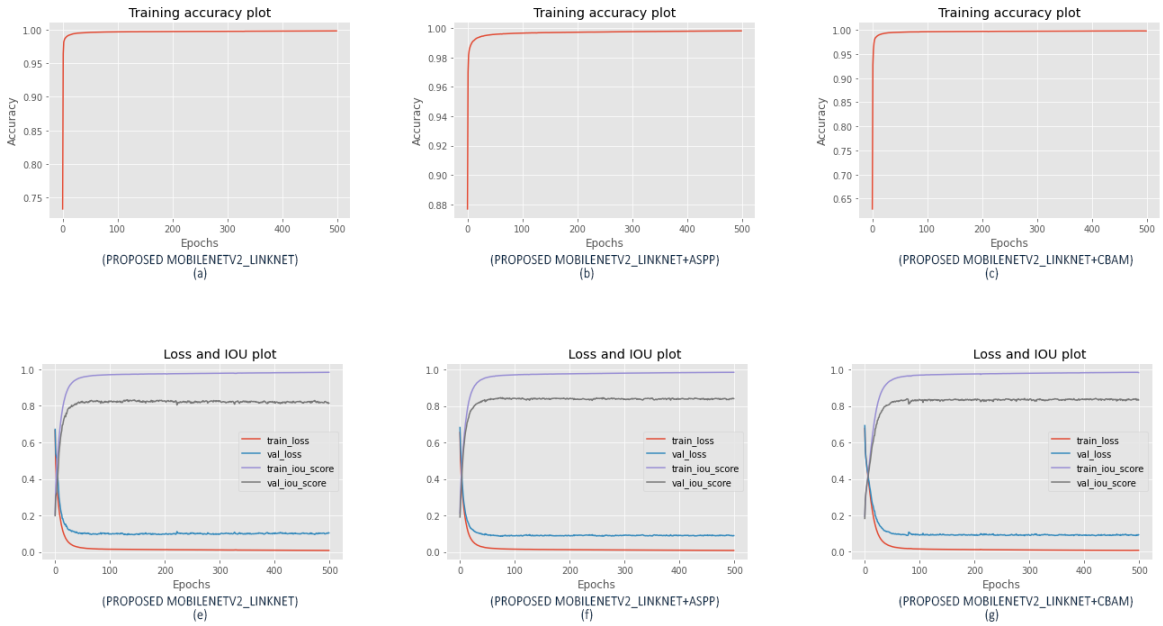

Figure 4.1: Training Plots

4.2 Results of Hand Segmentation Task

For quantitative comparison of the accuracy of the proposed methods, mIoU (Mean Intersection Over Union), commonly known as the Jaccard Index was measured on the Ouhands Test dataset. In the beginning, the efficacy of using a pretrained mobiletNetV2[21] encoder to a non-pretrained mobiletNetV2[21] encoder is compared. Two of the proposed methods were trained with ImageNet[15] pretrained weights and were compared with their non-pretrained versions in Table 4.2. It can be seen that the mIoU percentage of the two pretrained methods has a significant difference (14.42% and 13.36%) from the non-pretrained

methods. Thus, the ImageNet pretrained mobiletnetv2 encoder was proceeded to be used in the proposed model.

Table 4.2: Performance of models with or without ImageNet pretrained encoders

Model	**mIOU**
Non-pretrained Mobilenetv2 Linknet + ASPP	70.57%.
Non-pretrained Mobilenetv2 Linknet + CBAM	71.33%.
ImageNet Pretrained Mobilenetv2 Linknet + ASPP	84.89%.
ImageNet Pretrained Mobilenetv2 Linknet + CBAM	84.49%.

The segmentation performance of the proposed methods were then evaluated, with and without the two modules, ASPP [17] and CBAM [23], and also a combination of both the modules in Table 4.3. It is observed that using either the CBAM or ASPP modules along with the base proposed method has resulted in better performance compared to the base proposed method, with the ASPP module being slightly better than the CBAM module along with an increase in the total parameters by 2 million. The combination of both the ASPP and CBAM module on the other hand has lesser mIOU than using each of the modules individually with the base method. In conclusion, Mobilenetv2 Linknet + CBAM is the best method with a balance between segmentation accuracy and model size.

Table 4.3: Performance of different proposed methods

Model	**Total Parameters**	**mIOU**
Mobilenetv2 Linknet	4,146,617	83.57%.
Mobilenetv2 Linknet + CBAM	4,247,471	84.49%.
Mobilenetv2 Linknet + ASPP	6,155,193	84.89%.
Mobilenetv2 Linknet + ASPP + CBAM	6,256,047	84.33%.

For the demonstration of the performance of the proposed model, the difference between the proposed models and other segmentation models are compared and analysed. The original architecture of three state of the art segmentation models, HGR-Net [18], U-Net [19] and Google's Deeplabv3+ [17] (with ResNet-50 feature extractor) have been implemented. In Table 4.4, it is observed that the proposed methods are much superior compared to HGR-Net and U-Net. When compared with Deeplabv3+, it can be seen that

Table 4.4: Performance comparison with other segmentation methods

Model	**Total Parameters**	**mIOU**
HGR-Net[18]	279,633	72.06%.
U-Net[19]	31,055,297	77.82%.
Deeplabv3+[17]	11,852,353	83.87%.
Mobilenetv2 Linknet *	4,146,617	83.57%.
Mobilenetv2 Linknet + CBAM *	4,247,471	84.49%.
Mobilenetv2 Linknet + ASPP *	6,155,193	84.89%.

* : Proposed models

the proposed methods with significantly lesser parameters have a slightly better mIOU percentage. A qualitative comparison of the segmentation result on the OUHANDS dataset are illustrated in figure 4.2. It shows that the segmentation methods perform efficiently in different environments i.e., in different backgrounds and different lighting conditions.

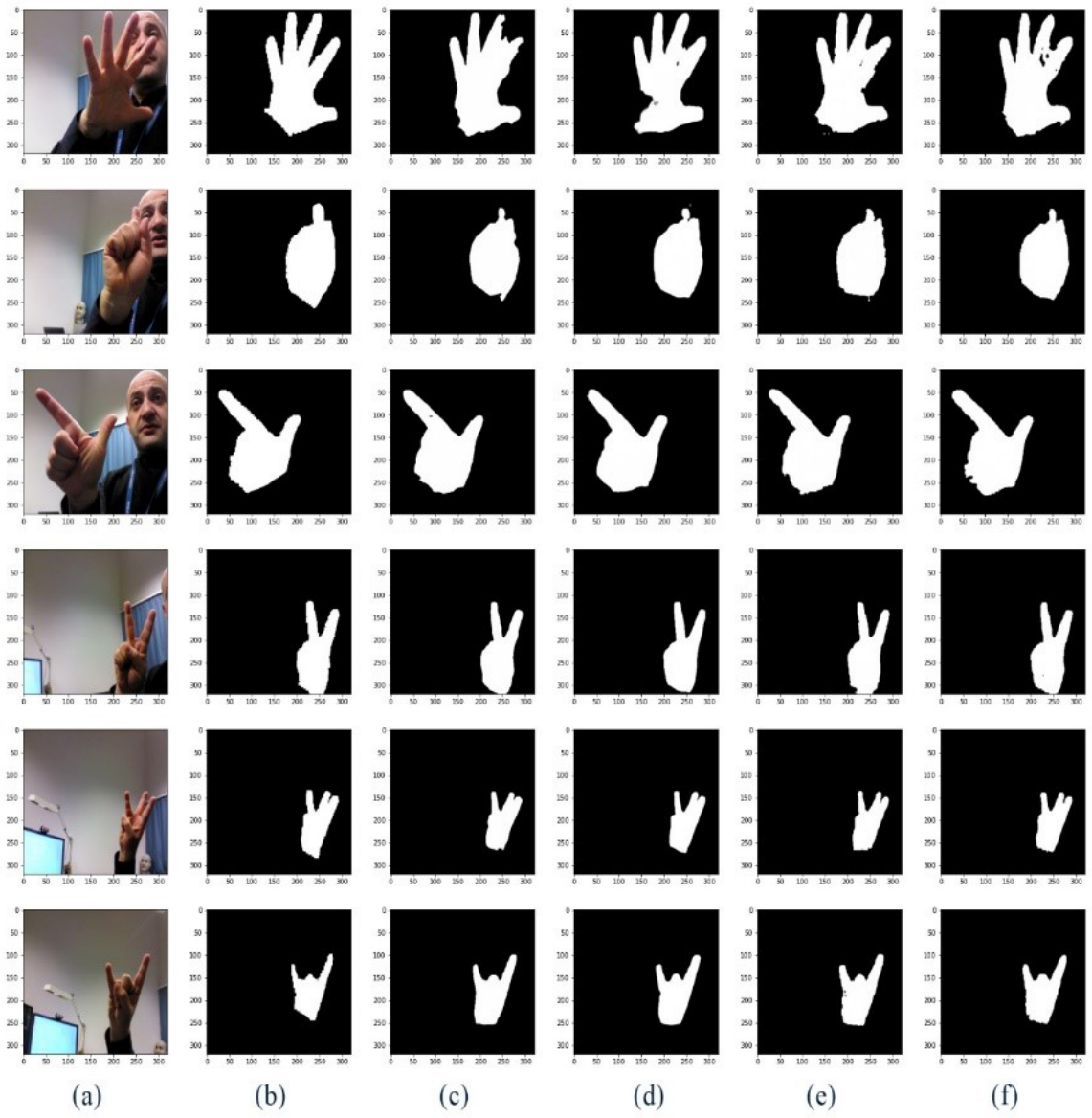

Figure 4.2: Example Segmentations on OUHANDS Test Set (a) Input Images (b) Ground Truth Segmentation Mask (c)Mobilenetv2 Linknet (d)Deeplabv3+ (e)Mobilenetv2 Linknet + CBAM (f) Mobilenetv2 Linknet + ASPP

The parameters and FLOPs of the proposed methods are further compared with those of other state-of-the-art methods in Table 4.5. HGR-Net has the lowest number of model parameters, but at the same time, experimentation has shown it has the lowest segmentation accuracy compared to all other experimented methods. The number of parameters of the proposed methods is significantly less than that of UNet and Deeplabv3+. According to the experimentation results, the segmentation performance of the proposed methods is comparatively better than the other three. Among them, the Mobilenetv2 Linknet + CBAM gives better segmentation at a lower computation cost.

Table 4.5: Parameters and GFLOPS comparison of the methods

Model	**Total Parameters**	**GFLOPS**
HGR-Net	279,633	30
U-Net	31,055,297	2410
Deeplabv3+	11,852,353	519
Mobilenetv2 Linknet*	4,146,617	87.8
Mobilenetv2 Linknet + CBAM*	4,247,471	101
Mobilenetv2 Linknet + ASPP*	6,155,193	88.2

* : Proposed models

4.3 Results of Hand Gesture Recognition Task

The Hand Gesture Recognition task is achieved by the 2-stream network model as used in the HGR-Net Model. After stage 1, which is the segmentation task, the image is passed to the classification stage.

The proposed models are now evaluated for the task of hand gesture recognition. The f1-score metric is used for evaluating the networks. The base model is first evaluated on the Ouhands Test Dataset. As desired optimum results were obtained from the evaluation, it was decided to proceed to test the models now on the Sattriya Test Dataset. Table 4.6 compares the performance of different models on the Sattriya Test Dataset. As observed when channel attention modules are attached to the base model, the f1-score significantly increases to above 0.9 with MobileNetv2 Linknet Aspp having the highest f1-score of 0.95. Mobilenetv2 Linknet record the lowest f1-score but it is also has the lowest number of parameters. Performance of Mobilnetv2 Linknet + ASPP and MobileNetV2 + CBAM is

better than state-of-the-art DeeplabV3+ model, also the parameters of each of them account for only 36.89% and 52.72% respectively of the parameters of DeeplabV3+.

Table 4.6: Recognition performance comparison of the methods

Model	**Total Parameters**	**f1-score**
Deeplabv3+	12,049,515	0.9081
Mobilenetv2 Linknet*	4,343,779	0.8717
Mobilenetv2 Linknet + CBAM*	4,444,633	0.9172
Mobilenetv2 Linknet + ASPP*	6,352,427	0.9502

* : Proposed models

Figure 4.3 and 4.4 shows the confusion matrix of the MobileNetV2 Linknet + CBAM network in terms of recognition accuracy on Ouhands and Sattriya testset respectively. The column and row indices denote the predicted results and the target classes,respectively.

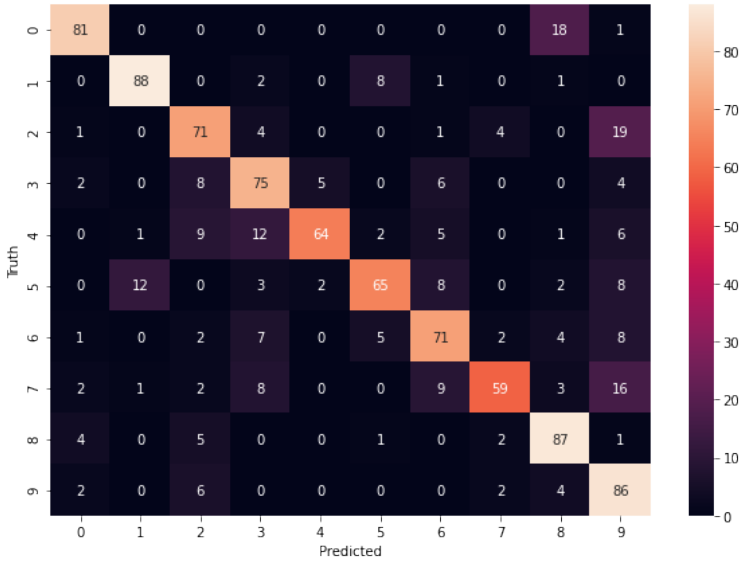

Figure 4.3: Confusion matrix on Ouhands testset

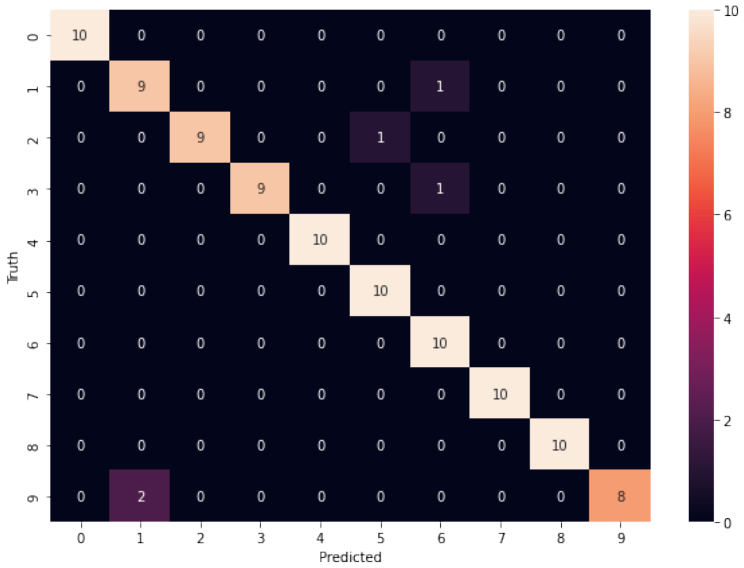

Figure 4.4: Confusion matrix on Sattriya testset

1. Recognition Model Evaluation with MobileNetV2 Linknet + CBAM Network (On Ouhands Test Set)

Table 4.7: Classification Report

Class Number	Precision	Recall	f1-score
0	0.87	0.81	0.84
1	0.86	0.88	0.87
2	0.69	0.71	0.70
3	0.68	0.75	0.71
4	0.90	0.64	0.75
5	0.80	0.65	0.72
6	0.70	0.71	0.71
7	0.86	0.59	0.70
8	0.72	0.87	0.79
9	0.58	0.86	0.69
Accuracy			0.75
Macro Avg	0.77	0.75	0.75
Weighted Avg	0.77	0.75	0.75

2. Recognition Model Evaluation with MobileNetV2 Linknet + CBAM Network (On Sattriya Test Set)

Table 4.8: Classification Report

Class Number	Precision	Recall	f1-score
0	1.00	0.70	0.82
1	1.00	0.70	0.82
2	0.83	1.00	0.91
3	0.77	1.00	0.87
4	0.91	1.00	0.95
5	1.00	1.00	1.00
6	0.91	1.00	0.95
7	0.91	1.00	0.95
8	1.00	1.00	1.00
9	1.00	0.80	0.89
Accuracy			0.92
Macro Avg	0.93	0.92	0.92
Weighted Avg	0.93	0.92	0.92

Observing table 4.8 be concluded that the model can be efficiently used to segment and recognize the hand gestures of the dancers to interpret the various meanings of the ancient dance form.

Chapter 5

Conclusion and Suggestions

Real-Time Performance of semantic segmentation in mobile devices is greatly restricted by the limited processing power of the hardware installed in such devices, used extensively in the field of hand gesture recognition systems. Existing work in this field involves a trade-off between segmentation accuracy and speed, the latter being more important for real-time application. Thus, there is a requirement for a model that can maintain a good balance between speed and accuracy for optimum performance. The model proposed in this project adopts a Mobilenetv2 backbone pre-trained on the ImageNet dataset and the decoder architecture of LinkNet using lateral skip-connections to reduce model parameters and computation cost. Additionally, experimentation was done by implementing two different modules with the base proposed method (Mobilenetv2 Linknet), an ASPP module and a CBAM module to improve segmentation accuracy without any significant increase in the number of model parameters. To verify the overall efficacy of the methods, they were compared with various state-of-the-art models on the OUHANDS benchmark dataset and it was confirmed that the proposed model, Mobilenetv2 Linknet + CBAM with 4.25 million parameters and 88.2 G floating-point operations, strikes the best balance between accuracy and speed, achieving a mean IOU of 84.49%.

Cultural art forms are a way to inform other people about one's culture. It is also the way that other people can have respect, gain knowledge, and give importance to a culture's traditions and norms. To help people learn about the beautiful dance form, Sattriya Nritya, the proposed models has been implemented to recognise and interpret the different gestures that the dancers make to depict a story. The models were trained on a self-made dataset of 1100 images. As of now, the proposed model, Mobilenetv2 Linknet + CBAM, can successfully recognise 10 hand gestures of the Sattriya dance form with an accuracy of

95%. Further collection of data will help enable us to make a complete database of the entire dance form.

5.1 Limitations

The main limitations of the study involve the ability to segment video data and the amount and variety of data available on different types of hand gestures. Temporal information needs to be considered for the model to perform efficiently on video data. The model is not able to properly segment two hand gestures at the same time. It was also observed that the skin colour of the hand affects the segmentation performance of the model. The collection of more data will help enhance the performance of the model, thereby addressing these limitations.

5.2 Future Scope

The proposed model focuses on semantic segmentation of hand postures which is a pre-requisite for real-time recognition of hand gestures in complex environment using RGB cameras. The project can be extended to build a classification model with low computational complexity conducive for real time application which can classify and recognise the hand gestures given by the user. Also future research can be carried out to reduce the model parameters and achieve a trade off between accuracy and model size which will make it more feasible to employ in real time application which can run on embedded devices.

Bibliography

[1] K Borah. Sattriya nrityar rup darshan. *Grantha-Sanskriti. Tarazan, Jorhat*, 2009.

[2] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[3] Matti Matilainen, Pekka Sangi, Jukka Holappa, and Olli Silvén. Ouhands database for hand detection and pose recognition. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–5. IEEE, 2016.

[4] Debajit Sarma and MK Bhuyan. Methods, databases and recent advancement of vision-based hand gesture recognition for hci systems: A review. *SN Computer Science*, 2(6):1–40, 2021.

[5] Chung-Ju Liao, Shun-Feng Su, and Ming-Chang Chen. Vision-based hand gesture recognition system for a dynamic and complicated environment. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2891–2895. IEEE, 2015.

[6] Abhishek B Jani, Nishith A Kotak, and Anil K Roy. Sensor based hand gesture recognition system for english alphabets used in sign language of deaf-mute people. In *2018 IEEE SENSORS*, pages 1–4. IEEE, 2018.

[7] Phat Nguyen Huu and Tan Phung Ngoc. Hand gesture recognition algorithm using svm and hog model for control of robotic system. *Journal of Robotics*, 2021, 2021.

[8] Malek Z Alksasbeh, Ahmad H Al-Omari, BA Alqaralleh, Tamer Abukhalil, Anas Abukarki, Ibrahim Alkore Alshalabi, and Amal Alkaseasbeh. Smart hand gestures recognition using k-nn based algorithm for video annotation purposes. *Indonesian Journal of Electrical Engineering and Computer Science*, 21(1):242–252, 2021.

[9] Phat Nguyen Huu, Quang Tran Minh, et al. An ann-based gesture recognition algorithm for smart-home applications. *KSII Transactions on Internet and Information Systems (TIIS)*, 14(5):1967–1983, 2020.

[10] Yi Li. Hand gesture recognition using kinect. In *2012 IEEE International Conference on Computer Science and Automation Engineering*, pages 196–199. IEEE, 2012.

[11] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International conference on image processing (ICIP)*, pages 1565–1569. IEEE, 2014.

[12] Di Wu, Fan Zhu, and Ling Shao. One shot learning gesture recognition from rgbd images. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–12. IEEE, 2012.

[13] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880, 2009.

[14] Norman Jouppi, Cliff Young, Nishant Patil, and David Patterson. Motivation for and evaluation of the first tensor processing unit. *ieee Micro*, 38(3):10–19, 2018.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[18] Amirhossein Dadashzadeh, Alireza Tavakoli Targhi, Maryam Tahmasbi, and Majid Mirmehdi. Hgr-net: a fusion network for hand gesture segmentation and recognition. *IET Comput. Vis.*, 13(8):700–707, 2019.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[20] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[21] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[22] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

[23] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[24] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[26] Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, oct 2020.