

Faculty of Informatics and Computer Science

Software Engineering

Dynamic Hand Gestures Recognition

By: Ahmed Mohamed Abohegazy

Id:158324

Supervised By

Dr. Walid Hussein

**August 2020**

Abstract

Computer become a necessity and critical part of our daily life. Human and computer interaction is carried out through traditional input tools such as the mouse, keyboard and other conventional , Hand gestures is going to be a valuable tool for HCI to enhance usability by establishing new methods for the interaction process.

Recent increase in Smart TV technologies means the need for special and advantageous Applications influence the study of the intelligence smart devices like environments gesture-based systems. To encourage enhancing the usability and natural interaction, it is proposed to merge film suggestions, social media sites, calling friends, weather reports, chat app into a central device controlled by a human gesture controller to leverage HCI properties

Also, in the field of automotive user interfaces, touchless hand gesture recognition systems are becoming relevant, as they improve safety and comfort.

Recognizing dynamic hand gestures in real–time Video feeds is a difficult task due to:

- No indication is given when a movement begins and finishes in live video feed.

- presented gestures must be detected (recognized) once and only once at a time

- The whole architecture must be planned with consideration towards memory and power consumption.

In addition to the human factor, there is a huge variety in the way people execute gestures, making it challenging to detect and recognize.

The objective is to develop a real time system that takes an input from a camera then predicts the gesture class(static, dynamic), By using the power of CNN to develop a model and train it on a dataset of human gestures and increasing the performance to reach a high accuracy without neglecting the performance.

Instead of analysing the hand shape separately in each image, we can analyse the hand movement in time, which means analysing multiple images at once. If you lift two fingers, it will be identified to the machine but the 3D CNN + LSTM gives us more than that, it would be able to sense whether we push the two fingers left or right or some kind of, which also lets us observe the motion and the hand gesture.

Table of Contents

Abstract ii

Table of Contents iii

1 Introduction 1

1.1 Overview 1

1.2 Problem Statement 2

1.3 Scope and Objectives 3

1.4 Developing stages (workflow) 3

2 Related Work 4

3 Proposed solution (Methodology) 6

3.1 Solution Methodology (why 3D-CNN +LSTM?) 6

3.2 Dataset (20BN-jester) 7

3.3 Pre-processing 8

3.4 3D-CNN architecture 9

3.5 LSTM architecture 11

3.6 Model tuning 12

3.6.1 Regularization (Dropout,l2 ) 12

3.6.2 Batch normalization 12

3.6.3 Optimization algorithms (Adam vs Adelite) 13

3.6.4 Activation function (Relu , SoftMax ) 14

14

3.7 Functional/ Non-functional Requirements 15

4 Implementation 16

4.1 System implementation 16

4.1.1 Hardware used 16

4.1.2 Libraries and environment 16

4.2 Architectures tested 17

4.2.1 3D-CNN 17

4.2.2 3D-CNN +LSTM (v1,v2) 18

4.2.3 3D CNN + 3 LSTM 20

4.2.4 3D ResNet 101 +2 LSTM : 22

5 Results and Discussions 23

5.1 Architectures results 23

5.1.1 3D-CNN 23

5.1.1.1 V1 23

5.1.1.2 V2 24

5.1.2 3D-CNN + LSTM 25

5.1.2.1 V1 25

5.1.2.2 V2 26

5.1.3 3D-CNN + 3 LSTM 27

5.1.4 3D ResNet 101 +2 LSTM 28

5.2 Analysis of tested architectures 29

6 Conclusions and Future Work 31

6.1 Summary 31

6.2 Future Work 31

References 32

Appendix 3 – Installation guide 33

# Introduction

## Overview

Computers and mobile systems are now an important part of our everyday lives. The demand for these computing tools increased the need for easy and convenient user interfaces for computers. For that purpose, smart systems that use vision-based interaction and control are becoming increasingly frequent which resulted in, gesture recognition becoming increasingly admired in the research community due to different applications possibilities in HCI.

vision-based user interface is more comfortable, functional and realistic when compared to mouse and keyboard because of the intuitive gestures.

Gestures are one of the most common means of human body interaction and may require arms, hands, eyes, neck, or body to communicate with and express specific knowledge about the world.

Bobick and Wilson offered a gestural definition. According to them, gesture can be described as the motion (gesture) using the body with intention to communicate with other humans [1]. For a good gesture the sender and recipient must have the same kind of knowledge.

There are three method to detect (recognize) hand gesture:

1. Glove based system in form of wearable devices
2. To localize the dimensionality of the hand shape though key points
3. By using (analysing) the data in its raw format

The first approach applied with the requirement to wear an extra device which normally has lots of cables needed and hardware , despite provides decent results in both respects Precision and speed,.

The second approach needs an extra process in the extraction of hand-key points, which will add additional time and computational costs.

the third option Only an image capture sensor (camera is needed), hence the user does not need to wear a complicated tool to obtain appropriate recognition Precision and adequate computing speed, this method shines among other methods as the most practical solution .

briefly any hand gesture recognition system needs to be practical cause users going to use it at daily basis.

## Problem Statement

Computer vision is the field of computer science, which works on replicating aspects of the complexities of the human vision system and allowing machines to recognise and interpret objects in images and videos in the same way that humans do, computer vision has only recently worked in a limited capacity.

In favour of developments in AI and advancements in deep learning (neural networks), the technology took major steps recently in certain activities relevant to classification and labelling of objects has been able to surpass humans.

One of the driving factors behind computer vision growth is the quantity of data get generated every day, which will be used to train and improve computer vision applications.

Gestures may get categorize into two sectors static or dynamic, static gesture remains nearly unchanged over time, but dynamic gesture seems to vary over time and

The key problem is how to make the hand gestures understandable by a computer. Hand gestures differ according to finger orientation and hand structure (shape).

Nonlinearity is, therefore, is important aspect of hand gesture must be resolved with using the images conveyed content information and metadata. The extracted data from images with hand gestures can be used to recognize the gestures, The process is carried out by integrating two tasks: extraction and classification of the features.

The features of an images(frames) must be extracted before recognition of any gesture. Any method of classification will be applied after certain features have been extracted. The challenge is to define the mechanism to extract the needed features and apply those classification features.

Classification and recognition of large features are mandatory. Conventional pattern recognition models cannot process naturally produced data in raw form. Therefore, tremendous efforts are required to extract features from the raw data and are not automated, CNN has the ability to extract features easily in addition to fully connected layer added to complete the task of classification . CNN uses the above methodology to lower the memory utilization and computational power and gives a better performance. In addition to grasp the complex and non-linear relationships between data (images).

CNN's specialty is not enough because the image content may be distorted by camera distance and lower pixel rescaling to improve the recognition process. Therefore, it is only when using the shape of the hand that the gesture cannot be recognized properly. The spatial-temporal Features Combination were suggested to be the best solution, particularly regarding problems of dynamic gesture recognition

## [Scope](http://www.cs.stir.ac.uk/~kjt/research/conformed.html) and Objectives

**Scope:** real time system that recognize and classify dynamic/static hand gestures.

**Objective:** using the power of 3D-CNN to develop a model that get trained on a dataset of human gestures and increasing the performance to reach a high accuracy without neglecting the performance. .in addition to using LSTM layers which result in extracting a combination of spatial-temporal features.

## Developing stages (workflow)

# Related Work

In the domain of static hand gesture recognition according to Recent analysis [2] has shown the superiority of the convolution neural Network (CNN) in representing and classifying images. , in this paper, CNN has the ability to learn complicated and non-linearity relations between data(images) Method for hand gesture recognition was proposed using CNN. Data augmentation such as rescaling, zooming, shear, flipping, width And height distortion were applied to the data.

The architecture training process was on eight thousands pictures and validated on 1600 images Divided into 10 classes. The model with augmented data got 97.12% accuracy which is almost 4% better than the standard model Without augmentation (92.87%).

In case of developing recognition of dynamic hand gestures from video feed as it’s proposed [3] These hand gesture challenges (note listed in the overview section) should be tackled by using a hierarchical architecture that enables Convolution neural network (CNN) models working offline Using sliding window to operate efficiently online.

The architecture suggested is composed of two methods:

(1) lightweight ConvNets system to recognize gestures (detector)

(2) deep ConvNets to classify the detected gestures.

Assess the architecture on available public datasets such as (Ego Gesture , NVIDIA Dynamic Hand Gesture Datasets) requiring temporary recognition And classification of hand gestures carried by . ResNeXt-101 model, in a form of classifier, got as state-of-the-art

offline classification precision Between 94.04% to 83.82% for the range modality on Ego Gesture and NVIDIA ranking.

regarding the problem of localization was discussed in the next paper [4] Human-computer interaction interfaces are increasingly needed for visual hand-gesture recognition. In several applications, hands inhabits only about 10% of the image, and the remainder of them contain history, human face, and the human body, most have a background, A human face, then a skeleton. Spatial hand location may be a difficult activity in such cases, and ground-level reality bounding boxes need to be provided for training which is not usually possible. Nevertheless, the hand location is not a necessity if the criteria are merely to recognize a gesture .

The developed hand-gesture recognition architecture will classify seven types of hand gestures in a real-time manner with user independent , achieving 97.1 percent accuracy in the dataset with basic contexts and 85.3 percent accuracy in the dataset with complicated history.

Furthermore on Real-time gesture recognition and HCI System[5] The entire system includes three units: hand detection, gesture (HCI) Recognition Based Recognition , Use deep convolutional neural network (CNN) to recognize gestures, and enable relatively complex gestures to be identified by using only one basic monocular camera. Introduce the Kalman filter for estimating the hand on the basis of which the mouse cursor control is carried out in a stable and smooth manner

The model got trained to differ between 16 kinds of gestures. 19,852 sample images, in total, are was retrieved from five people. Each gesture has more than 1,200 samples. For each gesture, 200 samples are used for validation , the model got 99.8%. accuracy when used input size of (64,64)>

furthermore, regarding using 3DCNN the next result was obtained through [6] Hand gesture recognition algorithm for drivers from challenges depth And data of strength using neural networks in 3D convolution. The approach incorporates information for the final forecast from different spatial scales.

It also employs increase in spatiotemporal data augmentation for more effective training and to reduce potential overfitting. Our system works out right the classification accuracy for the VIVA challenge dataset is 77.5%.

In order to increase the accuracy, the need to add LSTM was proposed [7] The 3 dimensional Coevolutionary Neural Network (3DCNN) combination proceeded by t Long Short-Term Memory (LSTM) configuration got used to extract spatial-temporal Features.

Finite State Machine (FSM), interacts at the end of the classification Model for controlling the effects of class decisions depending on the context of the application.

The result indicated that the merge of depth data and RGB kept 97.8% of the accuracy rate on eight chosen gestures, in addition to the FSM increased the recognition rate in a real-time test from 89% to 91%.

Moreover, on the architectures used [8], Recurrent three-dimensional coevolutionary neural network, which simultaneously detects and classifies complex hand gestures from Multi-modal Knowledge.

through evaluating the model on Dataset, gesture recognition system reaches 83.8 percent precision, outperforms state-of-the-art algorithms and exceeds 88.4 percent human accuracy

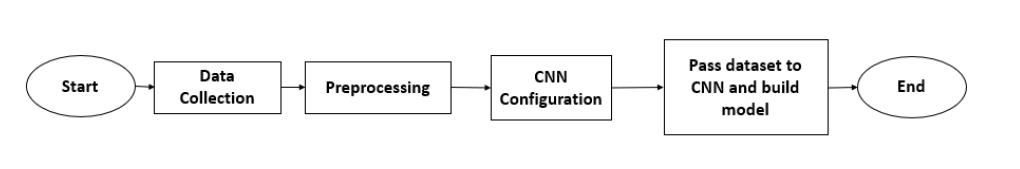
# Proposed solution (Methodology)

## Solution Methodology (why 3D-CNN +LSTM?)

Implemented CNN Enhanced Dimension as 3DCNN[7]. This approach can extract temporal features by preserving the images' spatial properties and was used in the field of behaviour recognition and video classification. One of the algorithms 's representatives is C3D. Although this algorithm can extract the temporal short-term features from the data set, it extracts the data only in a short-term manner. That inferences 3DCNN ' tends to fail when try to memorizer long sequences.

Hence gestures have around 32 to 50 frames for one gesture , this type of 3DCNN may not be able to recognise it better. Therefore, it is important to provide another network to learn temporal features on ling term functionality.

It has been suggested to merge the 3DCNN architecture with the LSTM network to aid in the process of learning the long-term temporal functions. With its sophisticated architecture, the LSTM has the advantages of learning long-term dependencies like input, output, and forget gates that control long-term sequence pattern learning.



The above figure demonstrates the phase of preparing the data to be trained

## Dataset (20BN-jester)

In recent years, recognition of gestures and their use in human-computer interfaces has become increasingly common [9], although some gestures could be recognized using a standalone image frame to create an effective and sensitive device that can recognize complex gestures

We need small variations between them in real-world video datasets on a large scale.

The dataset was obtained in their unconstrained environments, with the aid of over 1300 different actors.

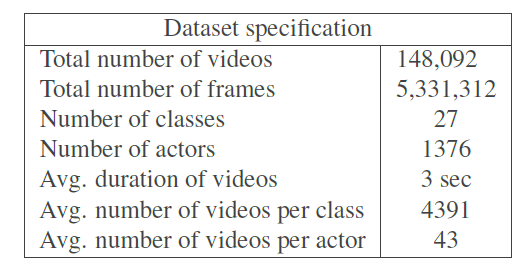
By using basic 3D convolutional neural network that achieves more than 93 percent recognition accuracy.

Various technologies have been developed for gesture recognition

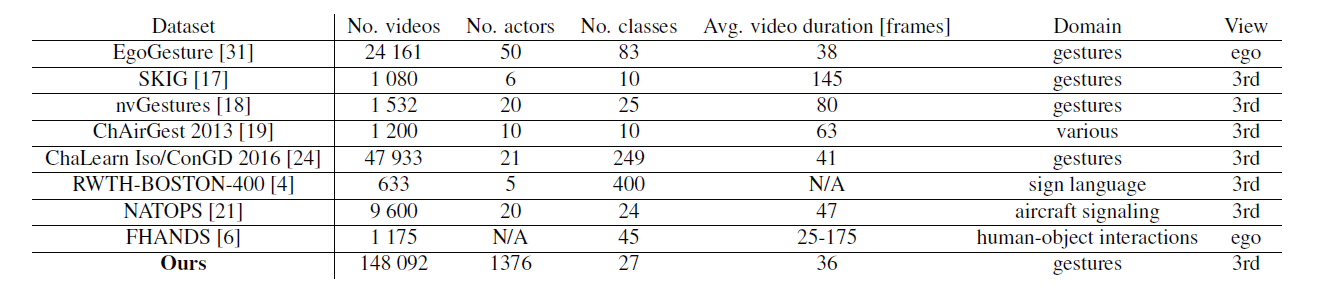
Over the years. For general recognition, wearable sensor systems is suggested and aimed at gaming, body language and sign language. Like some

Connect sensors that accurately monitor different types of information, such as motion velocity, hand orientation, acceleration, etc. The gestures are inferred from those details.

Computer vision solutions remove a device requirement (other than a camera)

Yet the train systems require vast quantities of data that can generalize to unknown situations.

It contains 148, 092 short 3-second-length video images, which total over 5 million frames. The video clips show a human gesturing in front of the camera. 1, 376 participants have reported a series of 27 acts in the data acquisition process. As such, there is a wide difference between actors in background and presentation.



Current datasets has variation in size or the number of forms of gestures and their context, number of performers, metadata given and data collection quantity . It’s suggest a modern dataset with a diverse selection of actors and video number.

The dataset focuses on a specific collection of types of behaviour covering the most frequently executed human gestures in graphical visual interface context

Categories of gesture are 25 classes of gesture , 5 which can be defined as static gestures. Those can be described by a single picture, [Drumming Fingers, Thumb Down, Stop Sign, Thumb Up, Shaking Hand]., The other types include a distinction between fine-grained graphic data such as "Zooming In With Two Fingers" and "Zooming In With Full Hand" or specifics about distance, "Rolling Hand Forward" and "Rolling Hand Backward."

Due to the computational power limitations I have selected 5 gestures to train and test them and considered to have static and dynamic gestures in my training samples

Static gestures [Thumb up , Thumb down]

Dynamic gestures [Drumming fingers, swipe two fingers right , swipe two fingers left ]

Total number of training samples equals 1800 video clips .

## Pre-processing

In order to sample the dataset into specific number of video for each gesture to train due to the limitation of computational power that not enabling to train on the whole dataset by using python script (extrecting\_training\_samples.py) the mechanism of this script is by reading csv file that contains all gestures in format of [target file ,gesture name] then by selecting the gesture wanted and the total number of videos to sample then the script returns the training sample of the selected gesture into a file with it’s name .

The next step is to overcome the problem that each gesture sequence in jester data set has slightly different duration , the solution is to normalize the temporal length of the gestures to 16 frame .

By using open CV, the frames after being normalized was resized to (100,100) and then converted into grayscale to contain only one colour channel which make it easier for the CNN to learn while the training process.

After applying the previous steps, the data get inserted into an array then converted into Numby array.

Then created label array to store the categorical classed of the gestures., then created 2D-array to store the gesture data and labels in the format of [data, label].

The next step is to normalize all our inputs to a standard scale, we're allowing the network to more quickly learn the optimal parameters for each input.

Finally by using Sklearn library (from sklearn imported train\_test\_split)

To split the data into training and validation partitions with percentage of 80% for the training and 20% for the validations

## 3D-CNN architecture

An effective approach for spatial-temporal feature learning is by using deep

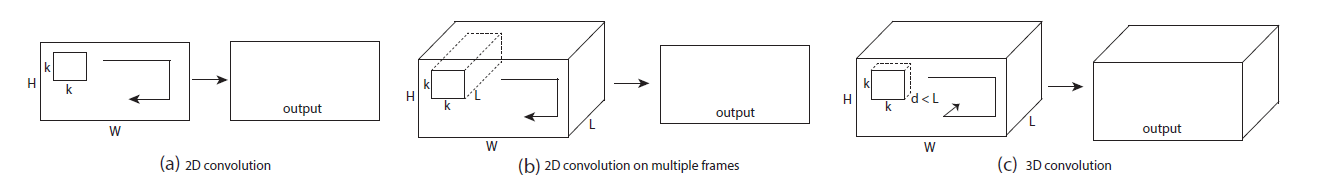
three dimensional convolutional networks (3D Conv) trained on a large scale

supervised and labelled video dataset.[10] 3D ConvNets is well designed for the learning of spatiotemporal features. 3D ConvNets has the potential to help represent the temporal details compared to 2D ConvNets Thanks to 3D convolution and 3D pooling.

Convolution and pooling operations are conducted spatial-temporally in 3D ConvNets while in 2D ConvNets they are conducted only spatially.

2D Convolution added to an data in shape of image will produce a image ,2D ConvNet applied to several images (treat them as separate channels) also produces an image.

2D ConvNets loses temporal input signal information immediately after each convolution process. Only 3D-convolution retains the input signal temporal information resulting in a volume of output not degraded. The same findings apply to 2D and 3D polls.

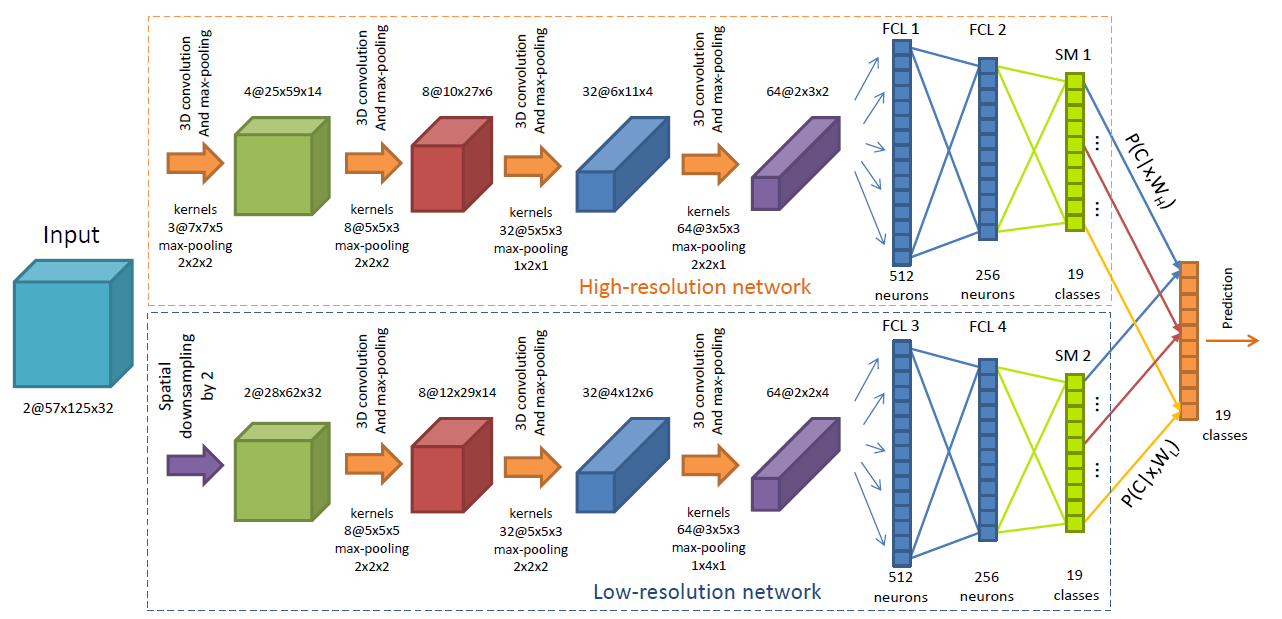


2D and 3D convolution operations[10]

1. Implementing 2D ConvNets on an image outputs in an image.
2. Implementing 2D ConvNets on a video volume (multiple frames as multiple channels) also outputs in an image.
3. Implementing 3D ConvNets results in another volume on a video stream, retaining temporal of the input signal information.

CNN Architecture consist of 3 parts:

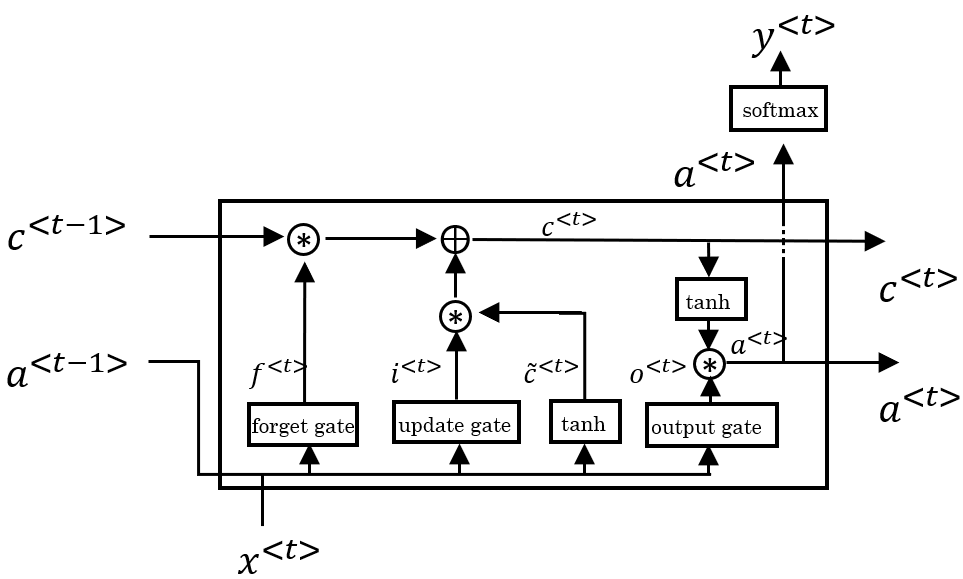
* Convolution Layer when the tensors get processed using kernels and apply activations function on the weights
* Pooling Layers used to reduce the size of representation, speed up the computations and make the features more robust
* Fully Connected (FC) Layer used to flatten the features and define the output size then use SoftMax activation function to get the probabilities of the outputs



## LSTM architecture

The LSTM units got the potential to learn from the input sequences regarding long range dependency[11], every LSTM unit has a memory cell for internal status storage And three additional gates for controlling the behaviour between the memory cell, the input(, t represent the time step ) and the output . The three gates are gate entry, gate origin, and gate forget

is a sigmoid function operator , is input non-linear translation, W and b are parameters of the model. Equation 5 Reveals that the forget gate determines how much the last internal state now adds to the internal state ,When it means 0, it forgets the last internal condition. The input gate determines how much the input influences the internal state by multiplying the non-linear data transformation by the cell , By putting those two pieces together, the cell memory is modified. Equation 6 Demonstrates that the output gate determines how much the internal state to pass to the output of the unit.Last time process outputs and cell memories are connected to the Three gates, allowing the LSTM unit catch the temporal input sequence relations, which is critical to the dynamic hand gesture recognition



## Model tuning

### Regularization (Dropout,l2 )

Firs of all what is regularization is considered any modification made to the learning algorithm that is intended to reduce the generalization error but not the learning error

Dropout:

Is going into each of the network layers and setting any likelihood of removing a node in the neural network what you are really doing is eliminating all the outgoing items from that node as well. But you end up with a much narrower network, which is very much weakened. And you do the back-propagation testing then.

Implementing drop-out consequence is that it shrinks the weights and executes some of those external laws that help avoid over-adjustment

l2 regulator:

L2 regularization formula, which defines the regularization term as the sum of the squares of all the feature weights:

### Batch normalization

The need to normalize the input is to make the cost function on average looks more symmetric , Batch normalization often enables each network layer to benefit from certain layers a bit more independently.

The following equations are used to normalize the input :

**Normalize mean:**

**Normalize variance:**

x/=

### Optimization algorithms (Adam vs Adelite)

Adadelta:

Adadelta is an Adagrad extension which therefore seeks to minimize the aggressive Adagrad that monotonously decreases the learning rate by reducing the range of the previous cumulative gradient to a certain defined size of w. The total operating time t then depends on the value above and the present gradient, We may not need to set the default learning rate in Adadelta, because we take the ratio of the running average of the previous phases to the actual gradient. Note (g) represent gradient divertive regarding w .

Adam — Adaptive Moment Estimation:

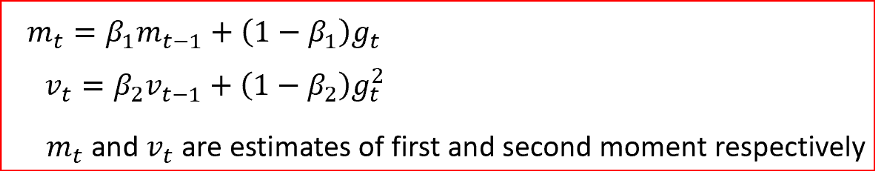
Adam can be used as an Adagrad mix that works best on sparse gradients, and RMSprop that works well in online and non-stationary environments.

Adam uses the exponential moving gradient average to measure the learning rate, rather than a flat average as in Adagrad. It preserves an exponentially decaying mean of past gradients

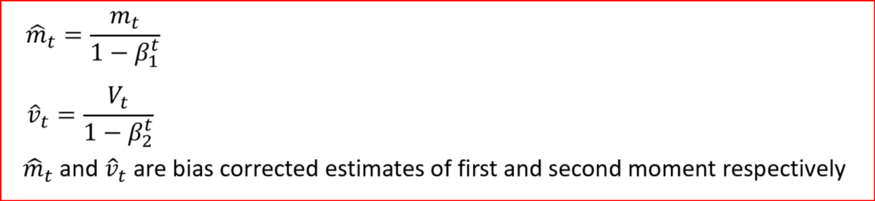
Adam is effective in computing and has very little memory need.

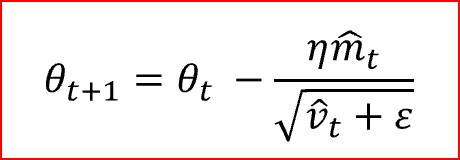
Adam algorithm first changes the linear moving averages of the gradient () and the square gradient (), which is the first and second moment figures.

Hyperparameters , , [0, 1] govern such moving averages' exponential decay levels as seen below



Moving averages are initialized as 0 which contributes to calculations of the moment that are skewed around 0 particularly during the initial time stages. Such tendency in initialization can be quickly counteracted, contributing to bias-corrected calculations.



 Finally, we update the parameter as shown below

### Activation function (Relu , SoftMax )

**ReLU (Rectified Linear Unit) Activation Function:**

the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

ReLU (Rectified Linear Unit) became very popular because it helps address some optimization problems observed with sigmoid. A ReLU is simply defined as:

function is zero for negative values and it grows linearly for positive values.

The ReLU is also very simple to implement (generally, three instructions are enough),

while the sigmoid is a few orders of magnitude more. This helped to squeeze the

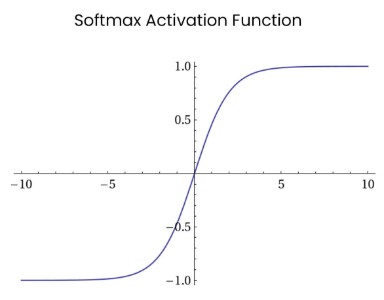
neural networks onto an early GPU

### 

**SoftMax:**

SoftMax functions are used for logistical regression and basic neural network implementation in machine learning, and they are the introductory activation units, which produce probabilities for the output .

z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z). And again, j indexes the output units, so j = 1, 2, ..., K.



## Functional/ Non-functional Requirements

There are some features of real-time gesture recognition implementations that the sys-tem must satisfy:

1. An adequate precision for the classification,
2. Quick response time,

(iii) resource utilization

1. single-time activation for any gesture conducted.

# Implementation

## System implementation

### Hardware used

The hardware specification used while the training and pre-processing the data :

|  |  |
| --- | --- |
| Hardware | Model |
| CPU | Intel core i7-9750 |
| Memory | 16gb DDR4 |
| GPU | Nvidia GTX 1650 4 GB |
| OS | Windows 10 |

### Libraries and environment

The code was developed using python 3.7 and Spyder ide was used as developing averment.

* TensorFlow GPU and keras:

Used as deep leering framework to build and train the code using GPU accelerator to enhance the training and speed up the computations.

* Matplotlib library:

Used to generate and visualize the training results (confusion matrix, error loss, accuracy)

* NumPy library:

Used to manipulate the tensor before feeding them into the neural network such as normalize the input data and adding them to an array.

* Open cv library:

Open cv played two major roles ,first of all used it to resize and convert into grayscale while pre-processing training sample , secondly used to capture the input through live feed from the camera then process it o feed it into the model to get predicted .

* Sklearn library:

Played a major role to split the data into training sample and validation sample

## Architectures tested

### 3D-CNN

Note the architecture consist of 2 version with only difference on the optimizer ,

The purpose of this architecture is to test whether optimizer Adam or Adadelta

In dealing with jester 20 bn .

Note : the architecture has 2 versions

The next table shows the parameter of the model architecture :

|  |  |
| --- | --- |
| Input size | (16,100,100,3) |
| Activation function | Relu and SoftMax for the dully connected layers |
| Batch size | 32 |
| Epoch | 200 |
| Optimizer | V1> Adadelta V2> Adam |

The next table shows the architecture of the model :

|  |  |
| --- | --- |
| Model content | Details |
| First 3D convolution layer | 16 filter of size (3,3,3),input shape (16,100,100,3),activation function Relu ,padding =’same’ |
| Batch normalization layer |  |
| Max pooling 3D | Pool size(2,2,2) |
| Second 3D convolution layer | 32 filter of size (3,3,3,) ,Relu |
| Batch normalization |  |
| Max pooling 3D | Pool size(2,2,2) |
| Third 3D convolution layer | 64 filter of size (3,3,3,) ,Relu |
| Batch normalization |  |
| Max pooling 3D | Pool size(2,2,2) |
| 4th 3D convolution layer | 128 filter of size (3,3,3,) ,Relu |
| Batch normalization |  |
| Max pooling 3D | Pool size(2,2,2) |
| Flatten layer |  |
| Dense layer | 64 neurons ,Relu |
| Batch normalization |  |
| Dropout layer | Dropout =0.20 |
| Dense layer | 64 neurons ,Relu |
| Batch normalization |  |
| Dropout layer | Dropout =0.20 |
| Output layer | 6 neurons, SoftMax |
| Metrics/loss | Acc/ categorical\_crossentropy |

### 3D-CNN +LSTM (v1,v2)

The second architecture ,the combination of 3D-CNN and LSTM help to extract spatial-temporal features which comes handy when dealing with problem such as Dynamic hand gesture recognition

The next table shows the parameter of the model architecture:

note :padding =’same’ , stride =(1,1,1)

|  |  |
| --- | --- |
| Input size | (16,100,100,3) |
| Activation function | Relu and SoftMax for the dully connected layers |
| Batch size | 32 |
| Epoch | 200 |
| Optimizer | V1,V2 > Adadelta |

The next table shows the architecture of the model version 1:

|  |  |
| --- | --- |
| Model content v1 | Details |
| First 3D convolution layer | 16 filter of size (3,3,3),input shape (16,100,100,3),activation function Relu ,padding =’same’ |
| Second 3D convolution layer | 16 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(2,2,2) |
| Max pooling 3D | Pool size(1,2,2) |
| Third 3D convolution layer | 64 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(1,2,2) |
| 4th 3D convolution layer | 128 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(1,2,2) |
| LSTM 2D layer | 64 filter of size (3,3) |
| Global Average Pooling 3D |  |
| Dropout layer | Dropout =0.5 |
| Output layer | 6 neurons, SoftMax |

The next table shows the architecture of the model version 2:

|  |  |
| --- | --- |
| Model content v2 | Details |
| First 3D convolution layer | 16 filter of size (3,3,3),input shape (16,100,100,3),activation function Relu ,padding =’same’ |
| Batch normalization |  |
| Second 3D convolution layer | 16 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(2,2,2) |
| Max pooling 3D | Pool size(1,2,2) |
| Third 3D convolution layer | 64 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(1,2,2) |
| 4th 3D convolution layer | 128 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(1,2,2) |
| Batch normalization |  |
| LSTM 2D layer | 64 filter of size (3,3) |
| Global Average Pooling 3D |  |
| Dropout layer | Dropout =0.5 |
| Output layer | 6 neurons, SoftMax |

### 3D CNN + 3 LSTM

This architecture is modified to have deeper neural network and 3 LSTM layers instead of 1.

The next table shows the parameter of the model architecture:

note: padding =’same’ , stride =(1,1,1)

|  |  |
| --- | --- |
| Input size | (16,100,100,3) |
| Activation function | Relu and SoftMax for the dully connected layers |
| Batch size | 32 |
| Epoch | 200 |
| Optimizer | V1,V2 > Adadelta |

The next table shows the architecture of the model :

|  |  |
| --- | --- |
| Model content v1 | Details |
| First 3D convolution layer | 16 filter of size (3,3,3),input shape (16,100,100,3),activation function Relu ,padding =’same’ |
| Second 3D convolution layer | 16 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(2,2,2) |
| Third 3D convolution layer | 32 filter of size (3,3,3) , Relu |
| 4th 3D convolution layer | 32 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(1,2,2) |
| 5th 3D convolution layer | 64 filter of size (3,3,3) , Relu |
| 5th 3D convolution layer | 64 filter of size (3,3,3) , Relu |
| 6th 3D convolution layer | 3D convolution layer |
| Max pooling 3D | Pool size(1,2,2) |
| 6th 3D convolution layer | 128 filter of size (3,3,3) , Relu |
| 7th 3D convolution layer | 128 filter of size (3,3,3) , Relu |
| Max pooling 3D | Pool size(1,2,2) |
| LSTM 2D layer | 64 filter of size (3,3) |
| LSTM 2D layer | 64 filter of size (3,3) |
| LSTM 2D layer | 64 filter of size (3,3) |
| Global Average Pooling 3D |  |
| Dropout layer | Dropout =0.5 |
| Output layer | 6 neurons, SoftMax |

### 3D ResNet 101 +2 LSTM :

The third architecture was implemented use unofficial version of ResNet ,the modification changed the convolutional layers from 2D ConvNets to 3D ConvNets in addition to my modification by inserting 2 Layers of LSTM before the output layer.

The next table shows the parameter of the model architecture:

|  |  |
| --- | --- |
| Input size | (16,100,100,3) |
| Activation function | Relu and SoftMax for the dully connected layers |
| Batch size | 8 |
| Epoch | 100 |
| Optimizer | Adadelta |

The next table shows the architecture of the model:

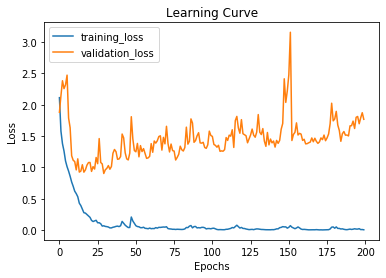
|  |  |
| --- | --- |
| Model content | Details |
| 3D ResNet | Input shape=(16,100,100,3) |
| LSTM 2D layer | 64 filter of size (3,3) |
| LSTM 2D layer | 64 filter of size (3,3) |
| LSTM 2D layer | 64 filter of size (3,3) |
| Output layer | 6 neurons, SoftMax |

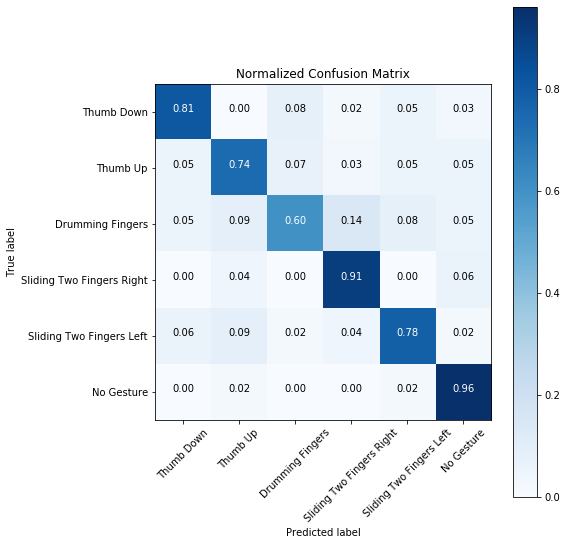
# Results and Discussions

## Architectures results

### 3D-CNN

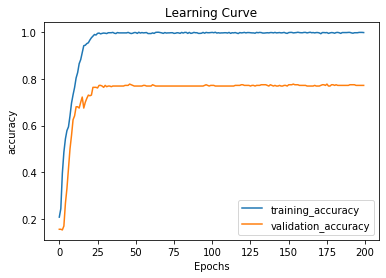
#### V1

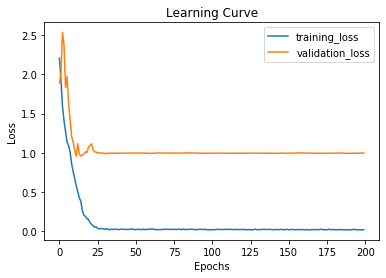


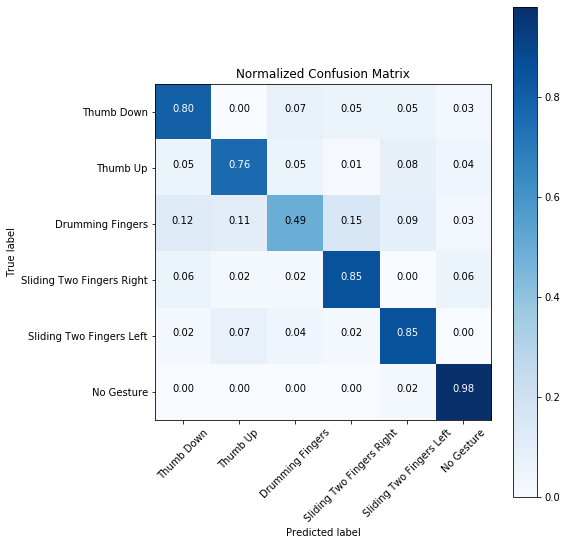


|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Val ACCURACY | LOSS | Val loss |
| 0.9986110925674438 | 0.7861111164093018 | 0.0053652412225751 | 1.28344591061274 |

#### V2



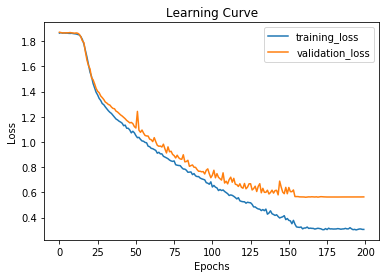
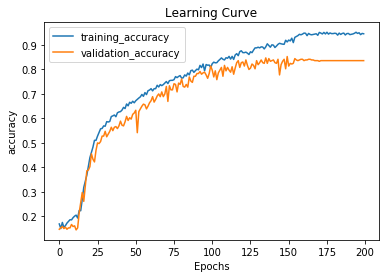




|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Val ACCURACY | LOSS | Val loss |
| 0.9965277910232544 | 0777777791023254 | 0.0260232564682761 | 0.99841263824039 |

### 3D-CNN + LSTM

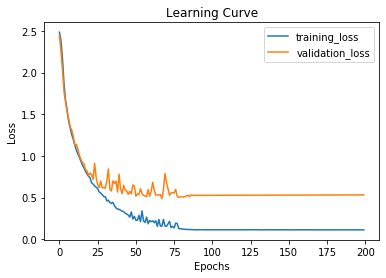
#### V1

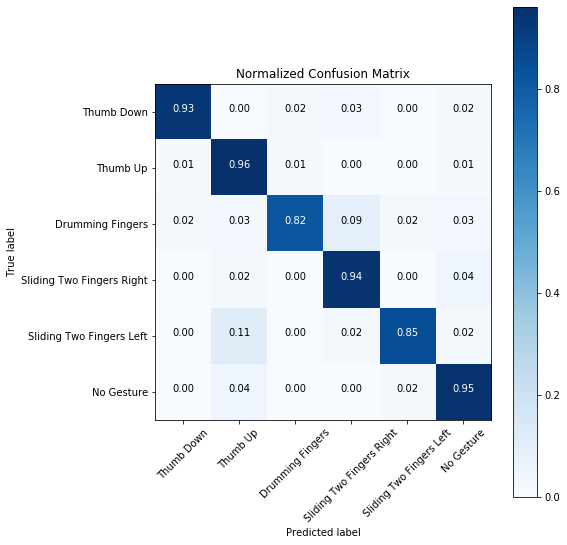




|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Val ACCURACY | LOSS | Val loss |
| 0.9458 | 0.8361 | 0.3058 | 0.5638 |

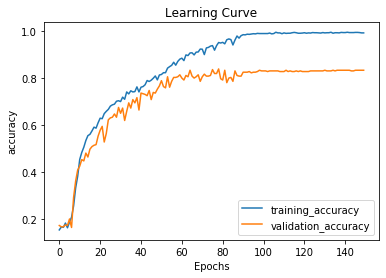
#### V2

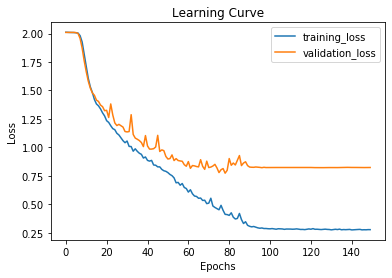




|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Val ACCURACY | LOSS | Val loss |
| 0.9854166507720947 | 0.9083333611488342 | 0.2008116000228458 | 0.51964872015847 |

### 3D-CNN + 3 LSTM

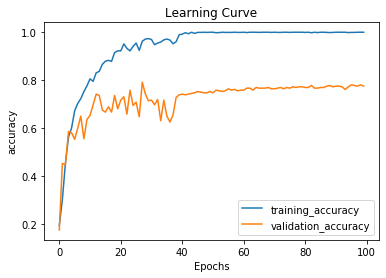


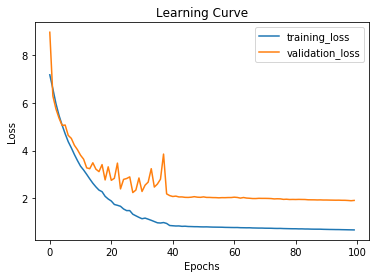




|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Val ACCURACY | LOSS | Val loss |
| 0.9924 | 0.8333 | 0.2766 | 0.823 |

### 3D ResNet 101 +2 LSTM



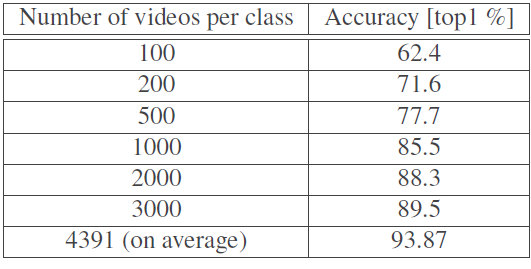




|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Val ACCURACY | LOSS | Val loss |
| 1.0 | 0.6769013226032257 | 0.2766 | 1.9121499631139967 |

## Analysis of tested architectures

Due to the nature of large-scale datasets there is an effect of the size of the dataset on the training accuracy as shown in the below figure.



Which was addressed when the problem of OOM(out of memory ) appeared in the early development stages when building the model architecture so the solution available was:

1. To upgrade the hardware to be at the recommended specs which are as follows ( 128 Gb of ram , 2 1080 NVidia GTX with 8 Gb of memory each and Finlay intel core i7 or i9 processor )
2. The more feasible solution is to build multiple architectures and evaluate them while pushing the training limits of the current hardware to the limits.

That’s why I have putting multiple architectures evaluation in the results part .

Regrading the training sample size as mentioned above is about 1800 video clips for 6 different gesture.

When coming to the hyperparameter settings I have used random values to set them but the challenging problem is the choice of the optimizer mean while other parts such as activation function , kernel size ,batch size etc was already recommend the best practice for them in the jester data set research paper [9].

Most people who entered the companions for classifying gestures using jester data set recommend the use of Adadelta optimizer instead of Adam, why?

The reasonable answer for me is Adadelta removes the use of the learning rate parameter completely by replacing it with the exponential moving average of squared deltas.

Which is more useful when dealing with jester dataset and the prove to that is the comparison between 3D-CNN V1 and V2, V1 with Adadelta gets better Val accuracy.

The need to add call-back while training is critical to know when to stop training and save best weights before the models starts to over fit.

call-back is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc).

the next properties was used :

-Early Stopping :

It’s used to stop training after specific number of epochs if there is no change for such as Val accuracy , and you set parameter called “patience” to specify the number of epochs after last change to stop training , in the case of the above trails it was set to 25

* Model Checkpoint :

Is used to save checkpoints of the model weights while training to use them after that in the predication process.

The problem of overfitting which some times referred as high variance and the problem of underfitting which some times refereed as high bias can be detect by monitoring the training values throw text or plots but plots are more useful to visualize if the model is overfitting or underfitting .

Other hints to detect the above issue:

* Train set high error most likely bias problem
* Validation set high error compared to train set error most likely variance problem

One more obstacle appeared how to prevent overfitting or underfitting with consideration of the relatively small training sample the next techniques was used across or with specific architectures to evaluate the results to get as best as possible model :

* Batch normalization
* L2 regularization
* Drop out

The trails shows that we can recognize hand gesture and obtain good results from fairly simple model only 3D convnet and LSTM, the model that shows promising results are :

* 3D-CNN + 3 LSTM
* 3D-CNN + LSTM V2

Considering the prediction process, I used a mechanism to get prediction from the two models and average the output which helps to improve the real time detection and recognition

Demo was created to use the predicted hand gestures to control Spotify basic features.

# Conclusions and Future Work

## Summary

The need to interact with computers and smart devices rapidly increased and becoming critical to most of our daily life routine which rise the question how to enhance the interaction methods between human and machine to ease the use and accomplish more tasks easily and quickly, one of the solution for that question is to use one of natural ways that humans use to interact with each other which represented in the form of hand gestures , even people from different cultures and different languages can interact and understand each other easily through gestures .

To over come the problem is how to make hand gesture recognition more easier without the need to wear devices to detect the gesture ,computer vision provided us with the solution to use the combination of two deep neural network [3D CNN , LSTM] to extract spatial-temporal features which enables the advantages of detect dynamic hand gestures and static hand gestures .

The model is trained on training sample contains 1800 video clips for 5 different gestures,

The gestures are combination of 2 static gestures and 3 dynamic gesture.

The model achieved 98% on the training accuracy and 90% on the validation accuracy.

In order to prove the idea of wide range usages of hand gestures application ,the model can recognize gestures and then use the predicted gesture to control Spotify basic features such as volume up ,volume down , previous , next , play/stop .

## Future Work

My future intentions regrading the project could be summarized into 3 points :

1. Enhance the accuracy by feeding the network more data and optimizing the architecture further more(note to do that the need to increase the darkwave capabilities to train the model ) .
2. Increase the model knowledge to detect more gesture classes to make it more robust to implement touchless user interface for complicated systems usage.
3. Add TensorFlow lite extension to enable Deployment of such models on mobile and IoT devices

References

1. A. Wilson and A. Bobick, “Learning visual behavior for gesture analysis,” Proceedings of International Symposium on Computer Vision – ISCV
2. M. Z. Islam, M. S. Hossain, R. U. Islam, and K. Andersson, “Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation,” 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2019
3. O. Kopuklu, A. Gunduz, N. Kose, and G. Rigoll, “Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks,” 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), 2019.
4. P. Bao, A. I. Maqueda, C. R. Del-Blanco, and N. García, “Tiny hand gesture recognition without localization via a deep convolutional network,” IEEE Transactions on Consumer Electronics, vol. 63, no. 3, pp. 251–257, 2017
5. A Real-time Hand Gesture Recognition and Human-Computer Interaction System. arXiv 2017
6. P. Molchanov, S. Gupta, K. Kim, and J. Kautz, “Hand gesture recognition with 3D convolutional neural networks,” 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015.
7. N. L. Hakim, T. K. Shih, S. P. K. Arachchi, W. Aditya, Y.-C. Chen, and C.-Y. Lin, “Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model,” Sensors, vol. 19, no. 24, p. 5429, 2019.
8. P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, “Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016
9. J. Materzynska, G. Berger, I. Bax, and R. Memisevic, “The Jester Dataset: A Large-Scale Video Dataset of Human Gestures,” 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019
10. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning Spatiotemporal Features with 3D Convolutional Networks,” 2015 IEEE International Conference on Computer Vision (ICCV), 2015.
11. Z. Xu, S. Li, and W. Deng, “Learning temporal features using LSTM-CNN architecture for face anti-spoofing,” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015.

Appendix 3 – Installation guide

To run the code without facing any issues and following libraries are needed :

* Python3.7
* Anaconda
* NumPy 1.18.1
* pandas 1.0.1
* matplotlib 3.1.3
* OpenCV-python 4.2.0.34
* Pynput 1.6.8
* Scikit-learn 0.22.1
* Keras 2.3.1
* TensorFlow 1.14.0
* TensorFlow-base 1.14.0
* TensorFlow-estimator 2.0.1
* TensorFlow -GPU 1.14.0
* TensorFlow -GPU-estimator 2.1.0