**A REPORT**

**ON**

**ACTION DETECTION BASED MODEL FOR DETECTING AND CREATING REAL TIME SIGN LANGUAGE DETECTIION MODEL.**

**BY**

**ROHIT KURUVILLA SANJAY 2018A7PS0001 COMPUTER SCIENCE**

**PREPARED IN FULFILMENT OF THE PROJECT COURSE**

**CS F376**

**A picture containing text

Description automatically generated**

**BITS, PILANI – DUBAI CAMPUS.**

**DUBAI INTERNATION ACADEMIC CITY (DIAC)**

**DUBAI,**

**UAE.**

**FIRST SEMESTER, 2021-2022**

**BITS PILANI, DUBAI CAMPUS.**

**DUBIA INTERNATIONAL ACADEMIC CITY**

**DUBAI, U.A.E.**

**FIRST SEMESTER, 2021-2022.**

**COURSE TITLE:** DESIGN PROJECT.

**COURSE NUMBER:** CS F376.

**DURATION:** SEPTEMBER 5TH, 2021 – DECEMBER 18TH, 2021.

**DATE OF START:**  SEPEMBER 5TH, 2021.

**DATE OF SUBMISSION:** DECEMBER 18TH, 2021.

**TITLE OF THE PROJECT:** ACTION DETECTION BASED MODEL FOR DETECTING AND CREATING REAL TIME SIGN LANGUAGE DETECTIION MODEL.

**ID NO.:** 2018A7PS0010U.

**NAME:** ROHIT KURUVILLA SANJAY.

**NAME AND DESIGNATION OF THE SUPERVISIOR:** DR. RAJA MUTHALAGU, HEAD OF THE DEPARTMENT.

**KEYWORDS:** LONG-SHORT TERM MEMORY MODELS, DEEP LEARNING, OPENCV, MEDIAPIPE, PYTHON, ACTION DETECTION, POSE ESTIMATION AND DETECTION

**AREAS OF FOCUS:** SIGN LANGUAGE, ACTION DETECITON, POSE ESTIMATION.

SIGNATURE OF STUDENT SIGNATURE OF SUPERVISOR

DATE:

**ACKNOLEDGMENTS.**

Firstly, I would like to extend my gratitude towards the Director of BITS Pilani, Dubai Campus, Dr. R.N Saha for all his efforts towards creating an environment of curiosity and learning at our college

I would also like to thank my supervisor, Dr. Raja Muthalagu, H.O.D of Computer Science for giving me guidance and supervision throughout the development of this project by necessary means of ideas, encouragement, and resources extensively.

I would finally like to extend my gratitude to my family and friends for being supportive and helping me with finishing this project in a nice collective environment.

**TABLE OF CONTENTS.**

**ACKNOLEGDMENTS ………………………………………………………………………..3**

**ABSTACT………………………………………………………………………………………….5**

**INTRODUCTION……………………………………………………………………………….6**

**LITERATURE SURVERY.…………………………………………………………………….8**

**PROPOSED APPROACH……………………………………………………………………9**

DATASET.

POSE ESTIMATION.

MODEL.

DIAGRAM.

**RESULTS………….…………………………………………………………………………….14**

**POSSIBLE IMPROVEMENTS…………………………………………………………….15**

**CONCLUSION………………………………………………………………………………….15**

**REFERENCE…………………………………………………………………………………….16**

**ABSTRACT.**

In today’s world, over 5% of the world population are considered physically challenged by “disabling” hearing loss, this equals to almost 400+ million people across the globe. There is almost 6000+ unique languages with their own different styles and gestures to help communicate with such challenged people. Due to the recent explosion and progress of computation and algorithms, we can tackle this new issue with different approaches and methods to find the most efficient way to the social problem showcased in this paper. From simple object detection to understand hand recognition to creating a real time hand pose and gesture recognizer, we are now able understand different ways to help create a platform where certified professionals who can communicate in sign language can help such challenged people. We have gathered 15 different papers ranging from simple journals and conferences like ICCV and CVPR. Each paper was selected that gives a different approach to the given problem statement that has been chosen. Some of the paper chosen do not exactly the exact problem but still comes under the same domain that we wish to operate. After that we showcase the simple approach, we used to train our model for real time detection of sign languages using media-pipe and LTSM models. Our model did not perform the best but due to complexity of real time detection our model does significantly detect the gesture properly but with glitches.

**INTRODUCTION.**

The first ever sign-language detection system was created by a company called MotionSavy, which used simple image classification techniques as a method to comprehend the challenge. After the years of small computation, after 1970s, there was a huge increase of computational abilities that was possible on computers thanks to Apple Inc. and so, different newer and more powerful algorithms started to come up. With the development of Artificial intelligence during the years of 1950s, and now combined with the computation explosion, we can see a huge jump in Artificial intelligence related approaches from symbolic reasons to sentimental analysis, computer scientists started figuring out new ways to create solutions to modern society. Around the years of 2000s, there was a new player formed in the Artificial intelligence community called “Deep Learning Approaches”. Deep Learning was a new method where we use neural networks and representation learning to help machines understand the dataset or problem given at hand. Some of the top deep learning approaches that are used prominently are **convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Deep Belief Networks (DBNs) etc., [1,2,10]** In this paper we come across many deep learning-based approaches which shall be explained further down blow.

We focus on Artificial Intelligence/Deep learning-based approaches on sign language recognition systems. A common issue that has been found across all the papers that have been reference is the lack of reliable datasets need. Many of the Sign recognition systems either train using just a few gestures or actions which hence limit the ability of the system itself to comprehend or even learn newer or other sign language inputs [3]. Since hand estimation is a difficult task due to so many parameters that come into play, from color variation, hand size, texture, length and even clarity of each visible gesture, orientation etc. the American Sign Language dataset is the most popular dataset that can be found that has collected over 100,000 different images for recognition. Each country has their own type of local sign language and due to this we cannot create a single solution that fits all. Many approaches include training with an existing dataset and then creating a gesture recognition system than can learn new signs so that it can stay true to other signs that the system was not trained before. Most initial system of sign recognition starts off with trying to capture accurate details of the hand itself as that is the keyframe and then make deductions of further analysis. As we move on towards more complex ideas, one of the most prominent and researched approaches is using the Xbox Kinect device as a reader for hand gestures and motion [10,11,12]. The Kinect offered a fully equipped hand tracking motion sensor that uses 3 different cameras to capture movement in a 3D space. Since they are equipped with depth sensing cameras and using the C++ dragonfly framework, it is seen as one of the most famous implementations for sign-language recognition systems [13].

In this paper, we try to focus mainly on how to use a single image from a typical RGB based camera found on our day-to-day laptops as the man input source for training. Even though cameras found on today’s phones are much capable of many more applications including depth, we stick to the simplest camera to achieve uttermost perfect so as it can be converted into more complicated focus later. For more clarity purposes, we can define RGB images as the images that can be represented using a 3x3 array that has representation of red, blue, and green within each cell. We prefer RGB images as they give us more better training and prediction accuracy as well as, in the real world it gives our application more robustness due to the different variation that can be found don hands ranging from size to shape and even skin color. One of the few characteristic we look for in our algorithm are mainly edge detection as our main focus is too be able to use this program where images might have many different elements or noises as backgrounds, we need our algorithm to be able to detect each and every gesture at a proper consistent speed without missing out gestures as well as we need it to be real-time detection to be able to show live demonstration and practicality. And finally, our algorithm should be able to hand different variations found to make sure the program is robust enough for different light situations especially.

So, a small introduction to our project is that we firstly had to decide between whether we wanted to go with already readymade dataset or create our own on command. Truth be told, most sign languages changes with respect to geography and culture. So, we decided to create our own dataset using the built-in camera, and then train them according to our need of action. We had to decide how to detect hand poses without and issues. And later use the key points that are detected being trained into a model which we settled for (reasons given in methodology). The biggest constraint we need to keep in mind was the fact this model needs to be lightweight enough to give back real-time feed outputs as well as have high accuracy.

**RELATED WORK**

For this project I referenced more than 35+ papers from direct sign language classification networks using HMM models to the ones using Kinect Sensors. After studying through all the papers, we have collected a total of 13 proper papers which we can reference for our project and equate it to the similar problem statement. Out of the 15 papers we selected, we took 5 papers for our referencing for code and step wise evaluation. One of the problems of using simple normal images to understand and analyze 3D hand size and hand pose. This paper uses takes a standby saying we can create a convolution graph network to reconstruct a fully hand mesh shape which gives more information than most other approaches. The author uses a weak supervision algorithm to make sure the training goes well using fine-tuning techniques. For hand pose estimation it is required to know the depth field, the author creates a depth map from a simple RGB Images, the database has both join network images and hand mesh images which were synthetic created just for the training [4]. While real time sign language can also be detected using simple lightweight algorithm that can be implemented in live video conference meetings and videocalls. The model used is an optical flow algorithm that takes in frames as it comes and then passes it into a temporal model. We use a uni-directional LSTM with one layer and 64 hidden layers. This gives us a proper 87% accuracy in real live translations and detection which is highly acceptable during conferencing. All the datasets that have been used from training/testing are from public DSG Corpus 301 video with almost 10 mins worth of recording [7]. RNN’s are also a easier way to implement a detection model like using Convolutional Neural networks as well as using object detection algorithms to get specific inputs per image. The main advantage of this implementation is the faster region-based convolutional neural network that has been kept properly into place. The author made a contrast between his method and other similar methods where he showcased a 98.2% accuracy in his model [12]. Another setup where we used depth cameras like for the implementation on how we use the input feed as a simple RGB depth image that is taken into a Hand segmentation algorithm called HandSegNet that understand the hand feature of the image and removes all other noises. This is then taken into a CNN based package called PoseNet that can recreate and understand different poses of hands given a particular image and depth. Using the output, we create a depth image of the position of the hand in a 3-dimensional space to give clarity [10]. Only one paper that we liked that used extra peripherals was the Kinect setup where, The Kinect senor is equipped with 3 cameras that can get more information than the typical consumer camera. The author taken in input from the Kinect sensor then passes it through a Hidden Markov model (HMM).

Diagram, schematic

Description automatically generated

*Fig.1. Abstract approach to sign language detection.*

The Dragonfly framework is then used to make sense of the received data to compile and classify each of the gestures and movements [13]. All these methods are highly accredited for their efforts in this problem, but most of these are just either approaches or just testing lab projects, this code is a simple approach that is lightweight and simple. These papers gave us inspiration and helped us walk through this process of creating our model. “3D Hand Shape and Pose Estimation from a Single RGB Image” showcased that we can create our own pose detection directly using images that combined with what we learnt from the paper “Real Time Sing Language Detection using Human Pose Estimation”, where LSTM models were used to train simple images from the dataset. The other 3 papers we used for reference and comparison to see if we need to switch up our style of analysis and approach as they presented the simplest and most effective models for sign language detection with high accuracy.

**PROPOSED APPROCH**

**Dataset**

The most crucial step for creating a sign language detection system is to find the right dataset to train the model with. While considering training the model itself there are a certain aspect that we need to take account for while deciding on which dataset to choose from for example, ASL dataset gives us a variety of different gestures and pose for sign languages, but the ASL dataset does not contain any information of each pose made in these pictures itself or even movements are not recorded on the sign language dataset itself. While we could impose algorithms to capture pose estimates on the images that they offer this task is tedious and makes the creation pose for new sign gestures difficult. Due to the reasons above, we approached our problem by taking it on us to create our own dataset. We had certain criteria in mind while trying to create a simple dataset. They are: -

* The images/video frames should be able to be captured at any given time.
* Extract key points such as estimation coordinates, structure and even orientation.
* No extra peripherals needed to capture the dataset.

To all these criteria and limitations, we found a solution from Google’s own MediaPipe package. MediaPipe, is a simple package that helps us extract pose estimations and key-points as well. Instead of using images for sign languages, a real time detection system would have a real-time feed that is being passed through the model itself and hence, we create our own short video clips of size 30 videos per gesture for training. These videos are overlayed with MediaPipe package to give use all the necessary values associated with each clip.

A picture containing text, wall, indoor

Description automatically generated A picture containing text, indoor

Description automatically generated

*Fig2. A sample of how the mediapipe reads and creates mesh structure pose estimations which are saved together as a whole file.*

**Pose estimation**

A collage of people playing football

Description automatically generated with low confidenceThis was one of our biggest challenges in the project. We could not use a depth camera like some of the other papers to get a more accurate read on poses made in real time. We could have tried to even use Kinect as shown in some other papers, Kinect sensors have 3 cameras that give heat, depth, and normal feed vision for reading. Since this project focus on simple implementation for typical cameras found on laptops and phones today, we couldn’t use

Fig3. MediaPipe Function and Detection

addition of other peripherals for our project. Now pose estimation means being able to detect where hands and fingers are at in a particular feed input. These are intense number of key points to be fed into a model. [13]

Firstly, to define what pose estimation is, it is the computer vision technique where it tracks locations of a person or an object in accordance with his pose and orientation. This is done by location and identifying a lot of key points a given person or object. For example, a hand can contain more than 21 key points that can be extracted, 4 on each finger and one for the palm. Human pose estimation is what we are considering for, where our subjects showcase a set of gestures, and we try to extract important values from each frame. There are 2 types of estimations, 2D and 3D pose estimations. 2D estimations just considers the x-axis and y-axis coordinates into consideration while for 3D the z-axis is also considered for prediction. Z-axis gives us the idea about orientation and depth as well. With 3D pose estimation, certain special features like spatial arraignment, depth or even alignment are features that can be extracted from the key points itself. There are 2 main approaches used in deep learning for pose estimations, Bottom-up and Top-down model. In Bottom-up mode, it detects instances of key points in each feed input and attempts to assemble groups of key points into a skeleton for objects. Top-Down model works on the other way, where it uses object detectors to first find the object in scope and then create the instance of key points.

For our pose estimation we use a Google package called MediaPipe. It is a open-source package that can be implemented to get accurate real time pose estimations, face detection, hand estimation and etc. through simple cameras found on your computer or phone. MediaPipe overlaps a mesh structure, projecting 1000+ data points depending on what portion of the bodies are being shown, in our project we have close to 1662 key data points that are going to be extracted. As the feed stream passes through our MediaPipe function it overlays these mesh datapoints that are extracted and tagged with each pixel. These are then stored as numpy arrays for us to be later passed on as models.

**Model used**

**LSTM Model:**

Selecting the model used for training is a tiresome task, we had eliminated a lot of typical approaches from CNN’s and simple UNets for our project. We stumbled upon two very effective models that can be implemented for our approach. They were, “Long Term Short Memory” models and “Recurrent Neural Networks”. The only reasons we inclined towards the LSTM model was that since this was a real time detection videos, we needed the continuous input as the current input depends to a certain level on the pervious inputs.

LSTM models are a special kind of RNN’s that are equipped with long-term dependencies that help remember values and information that can help for the next batch of processing. This model has a simple chain structure with the basic repeating design but differs from RNN in a simple way. Below is a diagram of a simple LSTM structure.

Diagram

Description automatically generated

Diagram

Description automatically generated

***Fig4***. The inside of an LSTM function.

Now from above we shall go through all the parts of an LSTM Networks with photos and mathematical functions.

For an LSTM network to work we need a memory that is gated just like a GR, which has a similar shape as the hidden state. To control what goes in, out and retained in the memory is controlled by 3 main gates, i.e., input gate, output gate and forget gate. These gates are fully connected networks via a sigmoid activation function that can be used to compute values for each gate. The equations are as follows:

Diagram

Description automatically generated

Where, It, Ft and Ot are input, forget and output gates. X is the input data, H is hidden state, W is weights of the network and b is the Bias of the network.

We now have a Candidate memory gate whose purpose is to control the flow of data into the memory cell (explained later). We use a tanh function which enables a larger range from (-1, 1). This gives us the equation of



LSTM memory cell is controlled by the input and output gates. The input gate facilitates how much data is passed into candidate cell and then the forget gate controls how much of the old memory is kept as well. We can condense this process into a simple mathematical formula



The hidden state is compiled with the output gate, ,it is controlled by a Tanh function and can be expressed as



Now, in our project the reason why LSTM fit properly is due to the fact these models needed less data to train and can give high accuracy and close prediction with less epochs. Due to our computational resource disability LSTM models can train much faster than most models out there with high number of parameters in consideration. Because we needed real time predictions LSTM models are faster in giving out predictions which is a key ingredient in creating real time detections

**Diagram**

Now below we have a representation in block form of the structed process through which our whole project is based on. It is divided into 3 parts, the data collection process, the training and model creation process and finally real time detection process.

**Data Collection Process:**

***A picture containing timeline

Description automatically generated***

***Fig5****. We use OpenCV to access the device camera and then impose MediaPipe libraries to get pose estimation key values and then stores them in devices.*

**The Training and Model creation Process:**

***Chart, waterfall chart

Description automatically generated***

***Fig6.*** *We access the clip and divide it into training and testing which then pass-through out LSTM model and then we validate our model using a testing set and later save the final model.*

**Real time Detection Model:**

***Timeline

Description automatically generated with medium confidence***

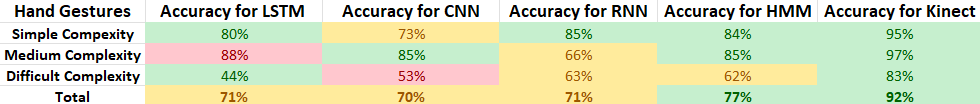
***Fig7.*** *We load our model into the camera stream using OpenCV and then use the model to detect real time gestures.*

From the above block diagram, it shows our novel approach to simply detection in real time. We tried to keep our approach as simple as possible due the speed worthiness of our model as well kept our accuracy as high as possible. Further improvements can be noted but this was the final approach we have made.

**RESULTS**

As we suspected we created a model that can now detect hand gestures in real time. Our model overlayed perfectly with OpenCV and each pose estimation that is being sent from the MediaPipe package into the LSTM model to accurately figure out which coordination and poses mean what word.

While we measure accuracy of our detection model, we achieve a 100% accuracy in predicting in static images, but this cannot be considered while describing real time accuracy. We personally tried testing for sign languages sign detections and the table below highlights our findings.



From the above we can concluse the following results.

* Our model suffers when the system is faced with more complex poses.
* We have a total real time testing accuracy of 78.04%.

***A picture containing text, person

Description automatically generated*Application

Description automatically generated**

***Hello Gesture***

***A picture containing text

Description automatically generated*A picture containing text, indoor

Description automatically generated**

***Thank you Gesture***

***A picture containing graphical user interface

Description automatically generated*A picture containing application

Description automatically generated**

***I love you gesture.***

When we contrast our approach with other well-known papers, The Kinect based approach had a accuracy of close 100% in most dataset it has tried with, but this is due to its extract Kinect sensor used to detect gestures. And all our typical CNN networks were able to achieve 60-80% depending on which dataset we are talking about.

**POSSIBLE IMPROVEMENTS.**

So now that we have seen the function of our model work perfectly with real time detection model inbuilt into the camera itself. The question arises that what more can we do to make this even better? Many approaches showcase that a single inbuilt model is not enough to capture the whole sing language system directly. We recommend a fully cross model platform where LSTM models are combined with NLP because then since sign language are derived from the same aspects of typical everyday speech, we can combine the power of NLP with action detection to accurate detect even more versatile and accurate.

Our database collection is comparatively small than what we would typically expect and that is because we needed a versatile system where we can create sign languages on the go depending on which ethnicity needs it, but our model also lacks clarity. Clarity is the concept of how clear a feed input is with respect to lighting and composition. Our model might suffer due the stagnant environment and condition which makes the model suffer as well, we recommend using different types of environments and different angles of gesture to help our model perform better.

Finally, we also recommend implementing a fully graphical interface that’s a little more user friendly that can give users the change to create new gestures when needed, maybe even a proper designed slot to showcase where the sentences are formed.

**CONCLUSION**

In this report we tried to tackle a real-world problem, where implementing a design system with light weight models and high accuracy is considered a programming feat. To summarize, we firstly created our own dataset for training so that we can implement our own specific sign-language sequence. We used MediaPipe package with OpenCV to capture real time pose estimations and coordinated for our object. The values are saved in the respective files and then we take these datapoints and pass it through our LSTM model. This LSTM model trains on this coordination and pose estimations which are almost 1662 key feature points are trainer so that our model knows what to look for in each gesture. The model is then loaded onto the OpenCV stream itself to give us back real time detection for which gesture is being showcased. There is a couple way to improve our whole model with combination of other open-sourced platforms as well try to get more sign languages trained as well. We could also have paired our LSTM model with an Fast-RNN network to take care of background lighting changes so that they don’t affect our overall performance.

In the future we could try creating a full stacked application to host the model perfect and deploy it when needed with designated attributes to simply train new gestures on command.

**REFERENCES:**

1. *Saleh Ahemed Khan, S.M. Asaduzzaman, Amit Debnath Joy, Morsalin Hossain, “An Efficient Sign Language Translator Device Using CNN and Customized ROI segmentation”, 2019, ICCET.*
2. *Meteen Ahmed, Mujtaba Idrees, Zain ul Abideen, Rafia Mumtaz, Sana Khalique,”Deaf Talk using 3D animated Sign Language”, SAI Computing Conference 2016.*
3. *Bowen Shi, Diane Brentari, Greg Shakhnarovich, Karen Livescu,”Finger Spelling Detection In American Sign Language”,2021*
4. *Christian Zimmermann, Thomas Brox, “Learning To Estimate 3D Hand-Pose From Single RGB Images”,CVPR, 2019*
5. *Thad Starner, Alex Pentland,”real time American sign language recognition from video using Hidden Markov models”,1997*
6. *Shadman Shahriar, Ashraf Siddiqueue, Tanverrul Islam, Abesh Ghosh, Rajat Chakraborty, Asir Intisar Khan, “Real Time American Sign Language Recognition Using Skin Segmentation and Image Category Classification and CNN and DL”,IEEE, 2018.*
7. *Amit Moryassef, Ioannis Tsochantardis, Roee Aharoni, “real time sign language detection using human pose estimation”,CVPR,2020*
8. *Priyanka Mekala, Ying Gao, Jeffery Fan, Asad Davan,”real time sign language recognition based on neural network architecture”,2018*
9. *Alina Kuznetsova, Laura Leal-Taize, Bodo Rosenhahn,”real time sign language recognition using consumer depth camera”, IEEE, 2013*
10. *Zulfiquar A Memom, M.Uzair Ahmed, S. Talha Hussain, Zahid Abbas Baig, Umer Aziz, “Real Time Translator for Sign Languages”, ICFIT, 2017*
11. *Brandon Garcia, Sigberto Alarcon Viesca, “Real-time American sign language recognition with convolutional neural networks”*
12. *Oishee Bintey Hoque, Mohammad Imrul Jubair, Md. Saiful Islam, Al-Farabi Akash, Alvin Sachie Paulson, “Real Time Bangladeshi Sign Language Detection Using Faster RCNN”,CVPR 2018.*
13. *Simon Lang, Marco Block, Raul Rojas,”Sign Language Recognition using* Kinect”.