3rd National Conference on Green Technology and Science for Sustainable Development
© 2020 IJSRST | Volume 5 | Issue 6 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
International Journal of Scientific Research in Science and Technology

Sign Language Recognition using Convolutional Neural Network

Prof. Pradyumna P. Kulkarni[1], Suraj S. Bhute[2], Akash P. Wagh[3]
[1,2,3]Computer Science and Engineering, Anuradha Engineering College, Sant Gadgebaba Amravati, Chikhli, India
Corresponding Author: kul.gokul@gmail.com

ABSTRACT

The inability to speak is considered to be a true disability. People with this disability use different modes to communicate with others, the hand gesture is one of the methods used in sign language for non-verbal communication. Developing sign language application for deaf people are often vital, as they'll be ready to communicate easily with even those that don't understand language. This paper presents a unique sign language recognition system that comprises of two-stage: Tracking and Representation. In the hand tracking phase, a hand dataset is used to extract the hand tract to pre-train Convolutional Neural Network hand models. The hand tracking is performed by the particle filter that associate with hand motion and CNN pre-trained hand models into a joint circumstance observation model. The predicted hand position corresponds to the situation of the particle with the best joint circumstance. Based on the predicted hand position is segmented and is the input to the hand representation phase.

Keywords : Sign Language Recognition, Convolutional Neural Network, Recurrent Neural Network, ISL

I. INTRODUCTION

The motion of any body part like a face, a hand is a form of gesture. Here for gesture recognition, we are using image processing and computer vision. Gesture recognition enables a computer to know human actions and also acts as an interpreter between computers and humans. This could provide the potential for humans to interact naturally with the computers without any physical contact of the mechanical devices.[1]

Sign language is additionally serving a similar meaning as speech communication does. This is used by the deaf and dumb community all over the world but in their regional form like ISL, ASL. Sign language are often performed by using Hand gesture either by one hand or two hands.[3] It is of two types Isolated sign language and continuous sign language. Isolated sign language consists of a single gesture having a single word while continuous ISL or Continuous Sign language is a sequence of gestures that generate a meaningful sentence.

TABLE I : Major Components of SLR

Fingerspelling	Word level sign vocabulary	No manual features
Used to spell words letter by letter .	Used for the majority of communicatio n.	Facial expressions and tongue, mouth and body position.

Deaf people around the world communicate using sign language as distinct from speech communication in their everyday a visible language that uses a system of manual, facial and body movements as the means of

communication.[2] Sign language isn't a universal language, and different sign languages are utilized in different countries, just like the many spoken languages everywhere the planet.

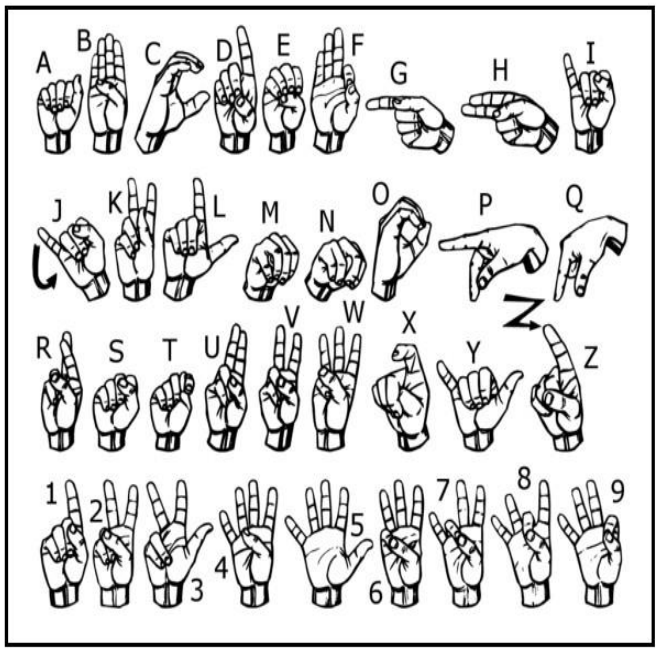

Fig 1: Finger Spelling Indian Sign Language [4]

II. RELATED WORK

A lot of research has been done in the field of sign language recognition using various approaches. Many methods are used for the conversion of Indian sign language into text. Various methods are deployed, each having its advantages and disadvantages. Many such employed are for the conversion of ASL to text. The same methods could be applied to interpret the ISL as well.

- **KarAradhana and Pinaki Sankar Chatterje** published the paper on "A Video-based Approach for Translating Sign Language to Simple Sentence in English" in 2013. In this paper, the first method uses a video-based approach for translating the sign to the equivalent sentence in English. It consists of 3 modules. The first module is the video processing module.[5]
- **Ashish S. Nikam Aarti G. Ambekar** published the paper on "Sign Language Recognition Using Image-Based Hand Gesture Recognition Techniques" in 2016. In this paper, the second method uses an Artificial Neural Network to interpret the ISL. It consists of getting the image and Preprocessing it to refine it, segmentation of the hand area, extraction of features, and final classification. [6]
- **Paulraj M P, Sazali Yaacob, Mohd Shuhanaz Zanar Azalan, and Rajkumar Palaniappan**, published the paper on "A Phoneme Based Sign Language Recognition System using 2D Moment Invariant Interleaving feature and Neural Network" on 2011. In this paper, a phoneme based method is also used to recognize the sign language. There are 44 phonemes in the English language; therefore 44 gestures can be formed. [7]
- **Rakesh.B. S, Tamilarasan.S, Avinash N** published the paper on "Hand Gesture Recognition based on Real-time Indian Sign Language". In this paper, we are introducing Hand Gesture Recognition, which will display messages based on input gestures. All input images are captured by a web camera. The proposed system doesn't require to be trained every time. [8]

III. METHODOLOGY

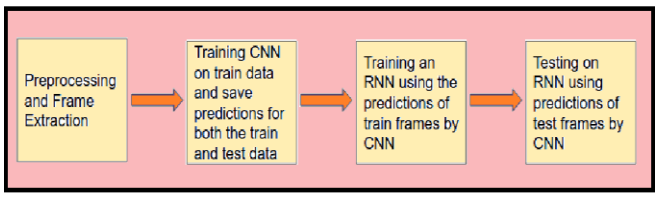

Fig 2 : Common Flow for training images using faster r-cnn

- First, we will extract the frames from the multiple video sequences of each gesture.
- After the first step, noise from the frames i.e background, body parts other than hand are removed to extract more relevant features from the frame.
- training on the spatial features. We have used inception model for this purpose which is a deep neural net.
- Store the train and test frame predictions. We'll use the model obtained in the above step for the prediction of frames.
- The predictions of the train data are now given to the RNN model for training on the temporal features. We have used LSTM model for this purpose.[9]

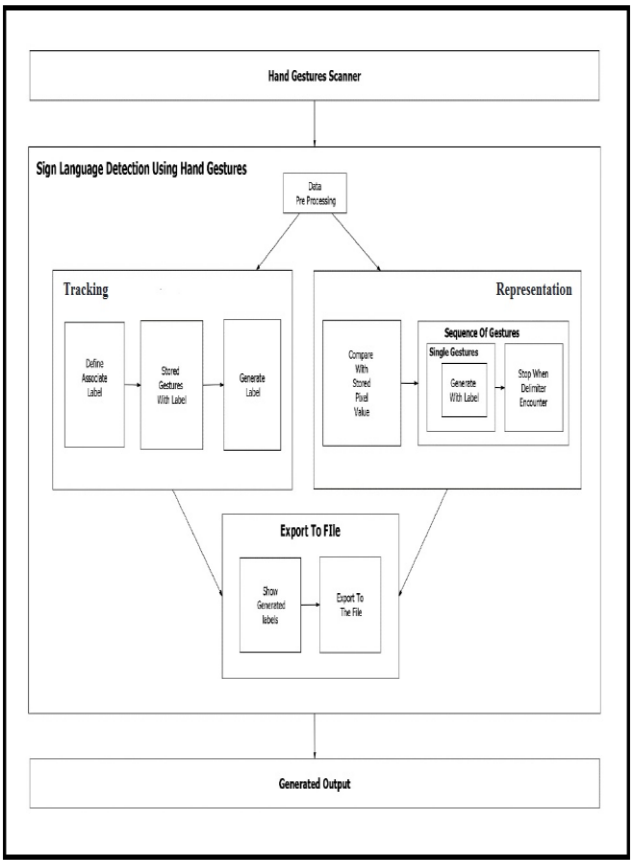

Fig 3 : System architecture for object classification using faster r-cnn

In further subsections of this section, each step of the methodology has been shown diagrammatically for a better understanding of that step.

1. Frame Extraction and Background Removal

The input symbols involve various background data which can be considered as noise or outliers. Then with the assistance of the color intensity variation and edge detection, gestures are recognized. The Gestures are recognized with the edge detection. Further, the image needs pre-processing for the accurate gesture detection.

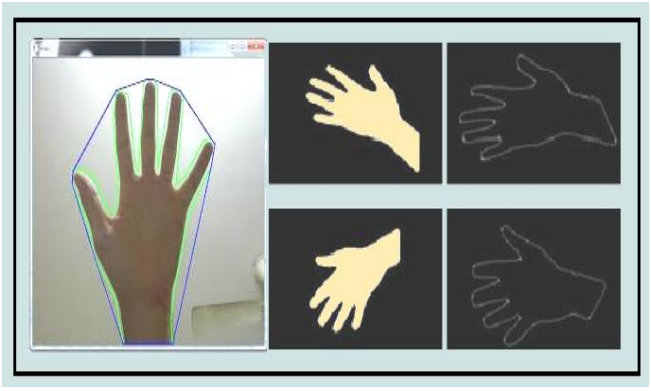

Fig 4 : Frame after extracting hands

The image with a detected edge is then moved for edge filtering. An image filter for background subtraction is implemented. The background image is omitted with the removal of the components outside the sting detected. The edge contains a number of the input file which will not be recognized for filtering. A threshold value is a reference to the intensity of the info that's processed. The portion of the image that matches the intensity value is retained are filtered.[11]

2. Train CNN(Spatial Features) and Prediction

A Convolutional Neural Network (ConvNet/CNN) may be a Deep Learning algorithm that may absorb an input image, assign importance (learnable weights and biases) to varied aspects/objects within the image and be able to differentiate one from the other.[12]

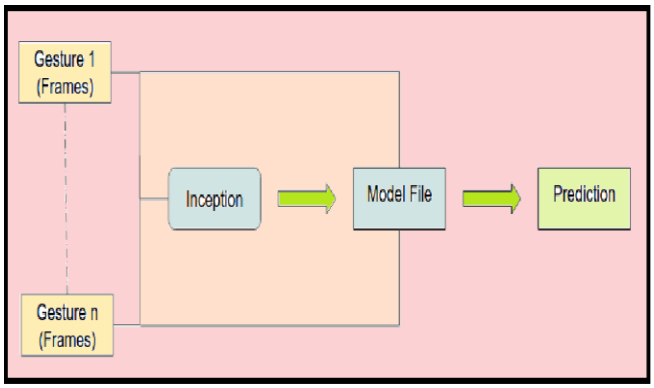

Fig 5: Frame Extraction and prediction using faster CNN.

3. Train RNN (Temporal Features)

Recurrent Neural Network(RNN) is a kind of Neural Network where the output from the previous step is

fed as input to the present step. In traditional neural networks, all the inputs and outputs are independent of every other, but in cases like when it's required to predict subsequent word of a sentence, the previous words are required and hence there's a requirement to recollect the previous words. Thus RNN came into existence, which solved this issue with the assistance of a Hidden Layer.[14] [15]

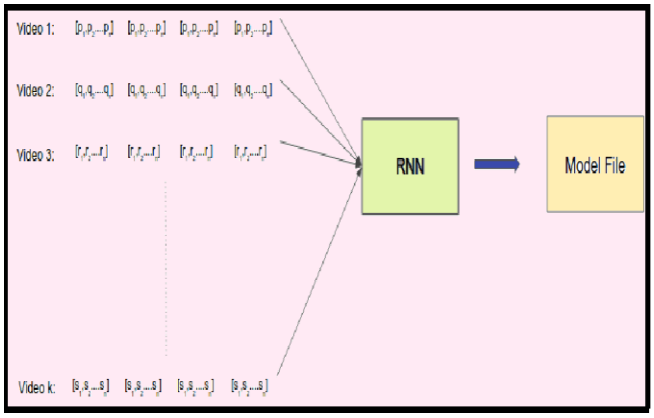

Fig 6: Object deduction using faster RNN

IV. ALGORITHMS

A. Convolutional Neural Network (CNN)

A simple ConvNet is a sequence of layers, and each layer of a ConvNet transforms one volume of activations to a different through a differentiable function. We use three main types of layers to form ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks), We now describe the individual layers and the details of their hyperparameters and their connectivity.[9]

Convolutional Layer

The Convolutional Layer is that the core building block of a Convolutional Network that does most of the computational work. It gets as input a matrix of the dimensions [h1 * w1 * d1], which is the blue matrix in the fig[13]

Spatial arrangement.

Three hyperparameters control the dimensions of the output volume: the depth, stride, and zero-padding. We discuss these next:

a. First, the depth of the output volume could be a hyperparameter: it corresponds to the number of filters we might prefer to use, each learning to appear for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate within the presence of varied oriented edges, or blobs of color. We will ask a group of neurons that are all watching an equivalent region of the input as a depth column (some people also prefer the term fiber).

b. Second, we must specify the stride with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is often rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

c. As we will soon see, sometimes it will be convenient to pad the input volume with zeros around the border. The size of this zero-padding is a hyperparameter. The nice feature of zero paddings is that it'll allow us to regulate the spatial size of the output volumes (most commonly as we'll see soon we'll use it to precisely preserve the spatial size of the input volume, therefore, the input and output width and height are the same).

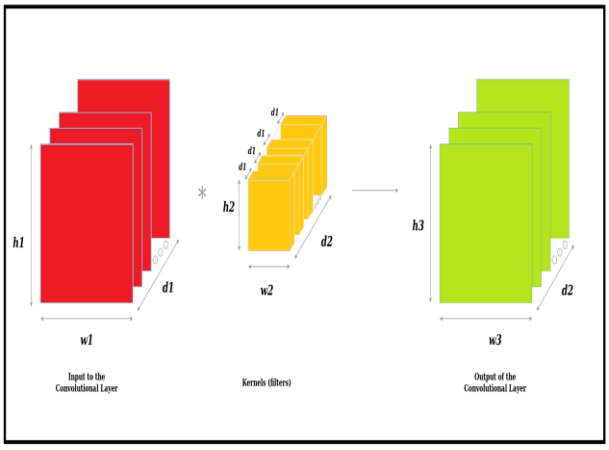

Fig 7: Convolutional Layer

A kernel may be a matrix with the size [h2 * w2 * d1], which is one yellow cuboid of the multiple cuboids (kernels) stacked on top of every other (in the kernels layer) in the above image. For each

convolutional layer, there are multiple kernels stacked on top of every other, this is often what forms the yellow 3-dimensional matrix in Fig 2, which is of dimensions [h2 * w2 * d2], where d2 is that the number of kernels. For each kernel, we've its respective bias, which may be a scalar quantity. And then, we have an output for this layer, the green matrix in Fig 2, which has dimensions [h3 * w3 * d2].

Let's throw light on some obvious things from above.

1. The depth (d1) (or the number of channels) of the input and one kernel is the same.
2. The depth (d2) of the output is equal to the number of kernels (i.e. the depth of the orange 3-dimensional matrix).

All right, so we have inputs, kernels, and outputs. Now let's check out what happens with a 2D input and a 2D kernel, i.e. d1=1. For each position of the kernel on the image, each number on the kernel gets multiplied with the corresponding number on the input matrix (blue matrix) then all of them are summed up for the worth within the corresponding position within the output matrix (green matrix). With d1 > 1, an equivalent thing occurs for every one of the channels then they're added up together then summed up with the bias of the respective filter and this forms the value in the corresponding position of the output matrix. [16]

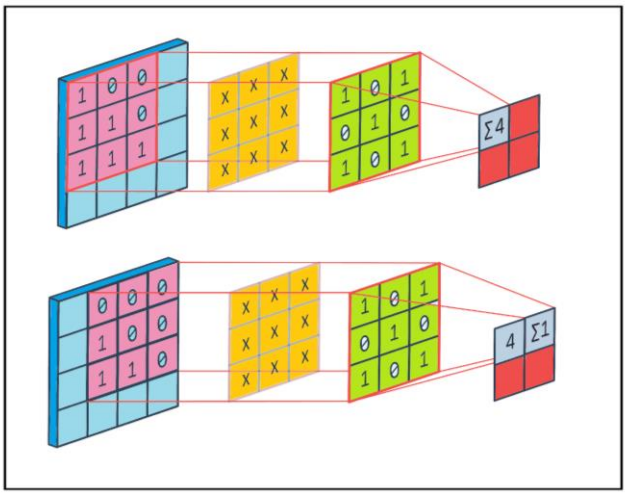

Fig 8: An example of how Inputs are mapped to Outputs

And this often forms one (of d2) matrix of the output layer. This entire process is repeated with all the d2 kernels which form the d2 channels in the output layer.

For each layer of the artificial Neural Network, the subsequent calculation takes place

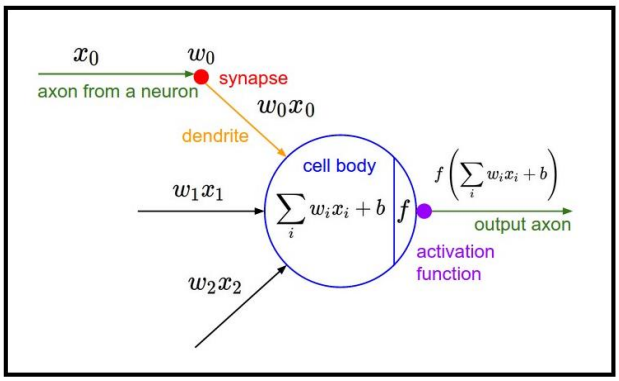

Fig 9: CNN Calculation for each layer[9]

where,

- **x** — is the input vector with dimension [p_l, 1]
- **W** — Is the weight matrix with dimensions [p_l, n_l] where p_l is the number of neurons in the previous layer and n_l is the number of neurons in the current layer.
- **b** — Is the bias vector with dimension [p_l, 1]
- **f** — Is the activation function, which is usually ReLU.

This calculation is repeated for each layer.

After passing through the fully connected layers, the final layer uses the softmax activation function (instead of ReLU) which is used to get probabilities of the input being in a particular class (classification). And so finally, we have the possibilities of the object within the image belonging to the various classes!!

Now, let's visualize how to calculate the size of the output tensor from the input tensor.

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

Fig 10: Output Dimension Calculations from Input Dimensions

where,

- ***W1*** — is the width/height of the input tensor
- ***F*** — is the width/height of the kernel
- ***P*** — is the padding

- S — is the stride
- $W2$ — is the output width/height

Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Convolutional layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation within the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form could be a pooling layer with filters of size 2x2 applied with a stride of two downsamples every depth slice within the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would, during this case, be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged. More generally, the pooling layer:

- Accepts a volume of size W1×H1×D1W1×H1×D1
- Requires two hyperparameters:
  - o their spatial extent F,
  - o the stride S,
- Produces a volume of size W2×H2×D2W2×H2×D2 where:
  - o W2=(W1−F)/S+1W2=(W1−F)/S+1
  - o H2=(H1−F)/S+1H2=(H1−F)/S+1
  - o D2=D1D2=D1
- Introduces zero parameters since it computes a set function of the input
- For Pooling layers, it's not common to pad the input using zero-padding.

General pooling. In addition to max pooling, the pooling units also can perform other functions, like average pooling or maybe L2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to figure better in practice. Pooling layer downsamples the quantity spatially, independently in each depth slice of the input volume.[10]

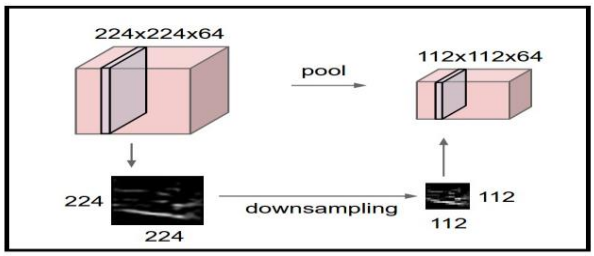

Fig 11: General Pooling

In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved.[10]

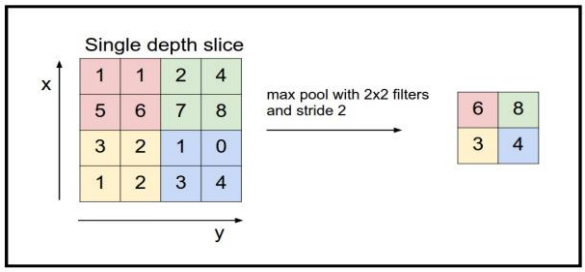

Fig 12: Max Pooling

The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of two. That is, each max is taken over 4 numbers (little 2x2 square).[10]

Backpropagation. Recall from the backpropagation chapter that the backward pass for a max(x, y) operation features a simple interpretation as only routing the gradient to the input that had the highest value in the forward pass. Hence, during the aerial of a pooling layer, it's common to stay track of the index of the max activation (sometimes also called the switches) in order that gradient routing is efficient during backpropagation.

Getting rid of pooling. Many people dislike the pooling operation and think that we will escape without it. For example, striving for Simplicity: The All Convolutional Net proposes to discard the pooling layer in favor of architecture that only consists of repeated Convolutional layers. To reduce the dimensions of the representation they suggest using larger stride within the Convolutional layer once during a while. Discarding pooling layers has also been found to be important in training good generative models, like variation autoencoders (VAEs) or generative adversarial networks (GANs). It seems

likely that future architectures will feature only a few to no pooling layers.[12]

Fully-connected layer

Neurons during a fully connected layer have full connections to all or any activations within the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix operation followed by a bias offset.

Fully Connected Layer is simply, feed-forward neural networks. Fully Connected Layers form a previous couple of layers within the network. The input to the fully connected layer is that the output from the ultimate Pooling or Convolutional Layer, which is flattened then fed into the fully connected layer.

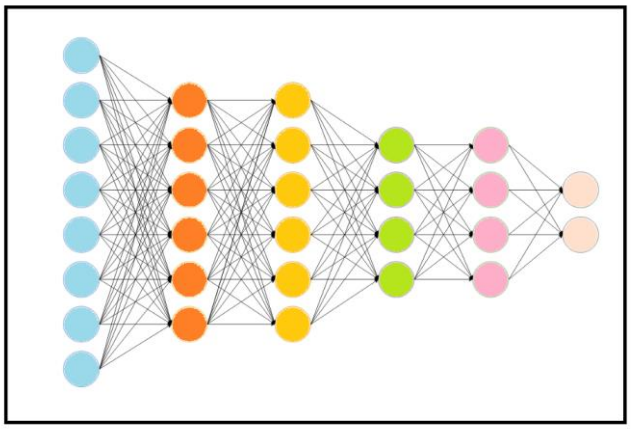

Fig 13: Fully-connected layer

The output from the ultimate (and any) Pooling and Convolutional Layer could be a 3-dimensional matrix, to flatten that's to unroll all its values into a vector.

This Flattened vector is then connected to a few fully connected layers which are the same as Artificial Neural Networks and perform the same mathematical operations!

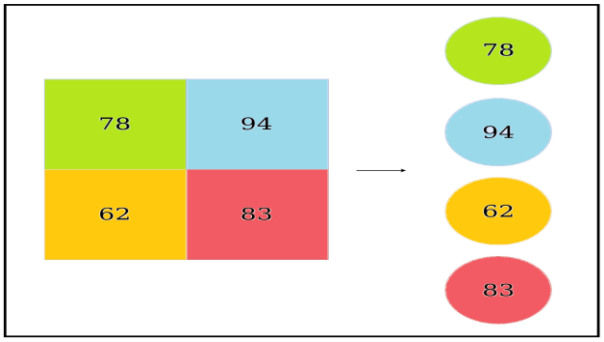

Fig 14. Flattening

A Convolutional Neural Network (ConvNet/CNN) may be a Deep Learning algorithm that may absorb an input image, assign importance (learnable weights and biases) to varied aspects/objects within the image and be able to differentiate one from the other. The pre-processing required during a ConvNet is far lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the power to find out these filters/characteristics.

The architecture of a ConvNet is analogous there to of the connectivity pattern of Neurons within the Human Brain and was inspired by the organization of the visual area. Individual neurons answer stimuli only during a restricted region of the visual field called the Receptive Field. A collection of such fields overlap to cover the entire visual area.

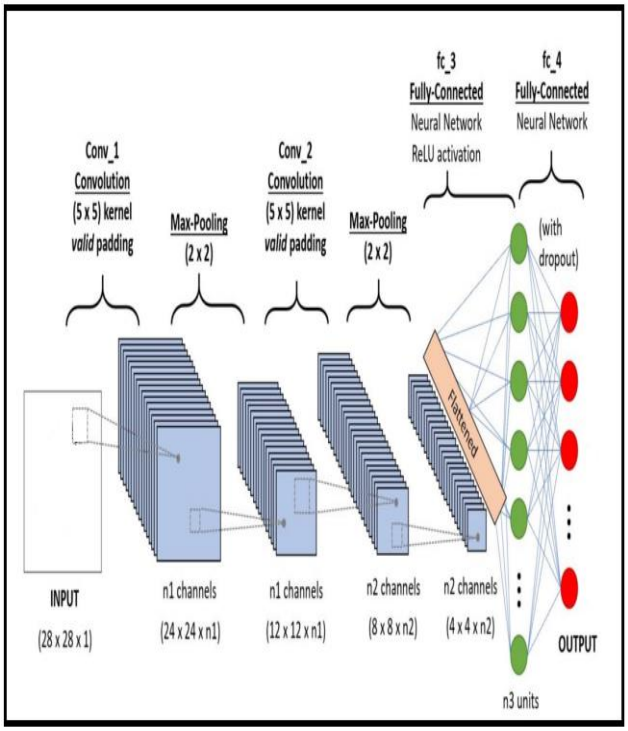

Fig 15: Architecture of a ConvNet

B. Recurrent Neural Network (RNN)

Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the present step. In traditional neural

networks, all the inputs and outputs are independent of every other, but in cases like when it's required to predict subsequent word of a sentence, the previous words are required and hence there's a requirement to recollect the previous words. Thus RNN came into existence, which solved this issue with the assistance of a Hidden Layer. The main and most vital feature of RNN is the Hidden state, which remembers some information a few sequences.

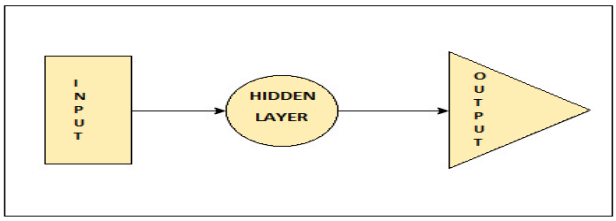

Fig 16: Common Flow of RNN

RNN has a "memory" which remembers all information about what has been calculated. It uses equivalent parameters for every input because it performs an equivalent task on all the inputs or hidden layers to supply the output. This reduces the complexity of parameters, unlike other neural networks.[18]

Suppose there's a deeper network with one input layer, three hidden layers, and one output layer. Then like other neural networks, each hidden layer will have its own set of weights and biases, let's say, for hidden layer 1 the weights and biases are (w1, a1), (w2, a2) for second hidden layer and (w3, a3) for the third hidden layer. This means that every one of those layers is independent of every other, i.e. they do not memorize the previous outputs.

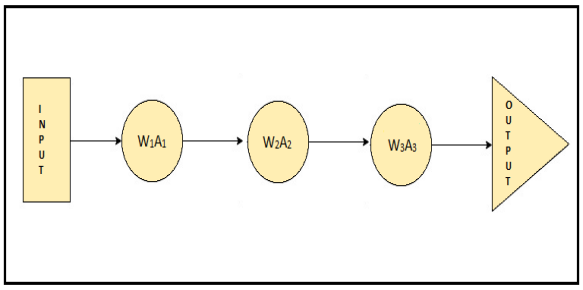

Fig 17: Deep Flow of RNN

Now the RNN will do the following:

RNN converts the independent activations into dependent activations by providing equivalent weights and biases to all the layers, thus reducing the complexity of accelerating parameters and memorizing each previous outputs by giving each output as input to the subsequent hidden layer.

Hence these three layers are often joined together such as the weights and bias of all the hidden layers is that the same, into one recurrent layer.[14]

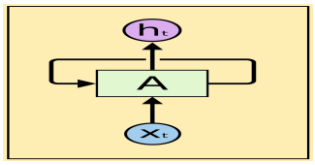

Fig 18: A chunk of Recurrent Neural Network

A recurrent neural network is often thought of as multiple copies of an equivalent network, each passing a message to a successor. Consider what happens if we unroll the loop:

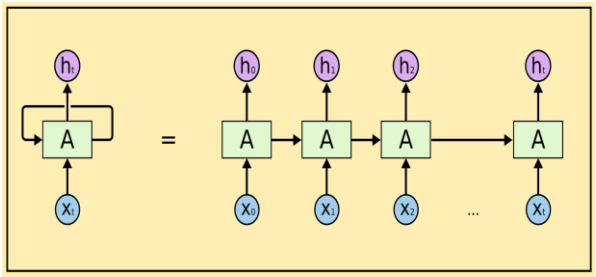

Fig 19: An Unrolled recurrent neural network

This chainlike nature reveals that recurrent neural networks are intimately associated with sequences and lists. They're the natural architecture of the neural network to use for such data. The sequential information is preserved within the recurrent network's hidden state, which manages to span many time steps because it cascades forward to affect the processing of every new example

Formula for calculating current state:

$$h_t = f(h_{t-1}, x_t)$$

Fig 20: Calculations for **current state**

where:

- h_t — current state
- h_{t-1} — previous state
- x_t — input state

Formula for applying Activation function(tanh):

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

Fig 21: Calculations for Activation function(tanh)

where:

- w_{hh} — weight at recurrent neuron
- w_{xh} — weight at input neuron

Formula for calculating output:

$$y_t = W_{hy}h_t$$

Fig 22 : Calculations for **output**

where:

- Y_t — output
- W_{hy} — weight at output layer

C. LSTM Networks

Long Short Term Memory networks – usually just called "LSTMs" – are a special quite RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) and were refined and popularized by many of us in the following work.1 They work tremendously well on an outsized sort of problem and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of your time is practically their default behavior, not something they struggle to learn!

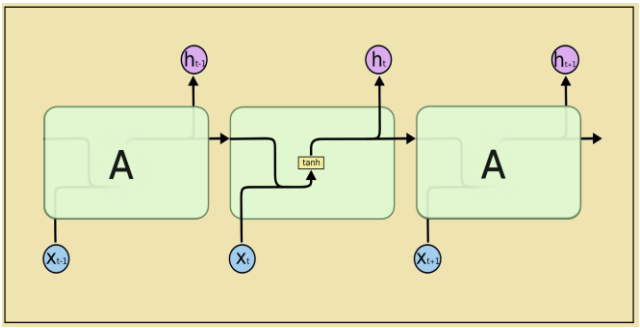

Fig23 : The repeating module in a standard RNN contains a single layer.

All recurrent neural networks have the shape of a sequence of repeating modules of the neural networks. In standard RNNs, this repeating module will have a really simple structure, like one tanh layer.LSTMs even have this chain-like structure, but the repeating module features a different structure. Instead of having one neural network layer, there are four, interacting in a very special way.[17]

Don't worry about the main points of what's happening. We'll rehearse the LSTM diagram step by step later. For now, let's just attempt to get comfortable with the notation we'll be using.

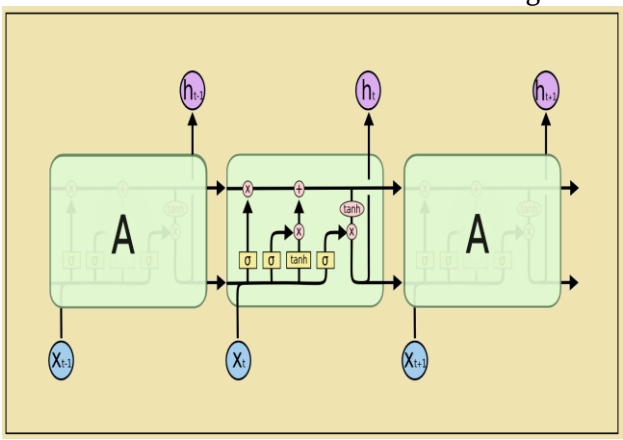

Fig24: The repeating module in an LSTM contains four interacting layers

In the above diagram, each line carries a whole vector, from the output of 1 node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers.Lines merging denote concatenation, while a line forking denotes its content being copied and therefore the copies getting to different locations.[15][19]

V. CONCLUSION AND FUTURE SCOPE

Hand gestures are a powerful way for human communication, with lots of potential applications in the area of human computer interaction. Visionbased hand gesture recognition techniques have many proven advantages compared with traditional devices. However, hand gesture recognition is a difficult problem and the current work is only a small contribution towards achieving the results needed in

the field of sign language gesture recognition. This report presented a visionbased
system able to interpret isolated hand gestures from the Indian Sign Language(ISA).

Videos are difficult to classify because they contain both the temporal as well as the spatial features. In this Paper, We have used two different models to classify on the spatial and temporal features. CNN was used to classify on the spatial features whereas RNN was used to classify on the temporal features. This shows that CNN along with RNN can be successfully used to learn spatial and temporal features and classify Sign Language Gestures.

We wish to extend our work further in recognizing continuous sign language gestures with better accuracy. This method for individual gestures can also be extended for sentence level sign language. Also the current process uses two different models, training inception (CNN) followed by training RNN.

VI. REFERENCES

[1]. S. Suharjito, R. Anderson, F. Wiryana, M. C. Ariesta, G.P. Kusuma, "Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output", Procedia Computer Science, vol. 116, pp. 441-448, Oct. 2017.

[2]. Aditi Kalsh, N.S Garewal, Sign Language Recognition System, International Journal of Computational Engineering Research, pp. 15-21, vol. 3.”.

[3]. J Singha, K DasIndian “Sign Language Recognition Using Eigen Value Weighted Euclidean Distance Based Classification Technique.” arXiv preprint arXiv:1303.0634, 4 (2) (2013), pp. 188-195

[4]. Copyright © William Vicars, Sign Language resources at LifePrint.com,http://lifeprint.com/asl101/topics/wallpaper1.htm Accessed Jan 26, 2020]

[5]. Kar, Aradhana and Pinaki Sankar Chatterjee. “A Video-based Approach for Translating Sign Language to Simple Sentence in English.” (2010).

[6]. A. S. Nikam and A. G. Ambekar, "Sign language recognition using image based hand gesture recognition techniques," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore,2016,pp.1-5.doi: 10.1109/GET.2016.7916786

[7]. M P, Paulraj & Yaacob, Sazali & Zanar Azalan, Mohd Shuhanaz & Palaniappan, Rajkumar. (2010). A phoneme based sign language recognition system using skin color segmentation. 1 - 5. 10.1109/CSPA.2010.5545253.

[8]. Rakesh.B.S, Tamilarasan.S, Avinash N, “Hand Gesture Recognition based on Real-time Indian Sign Language,” International Journal of Computer Sciences and Engineering, Vol.7, Issue.7, pp.181-185, 2019.

[9]. Sanil Jain and K.V.Sameer Raja,“Indian Sign Language Character Recognition”,.Available:https://cse.iitk.ac.in/users/cs365/2015/_submissions/vinsam/report.pdf Accessed Jan 16, 2020].

[10]. Andrej Karpathy,”CS231n: Convolutional Neural NetworksforVisualRecognition.”,Available: http://cs231n.github.io/convolutionalnetworks/ Accessed Jan 31, 2020]

[11]. D. Anbarasan | R. Aravind | K. Alice "GRS – Gesture based Recognition System for Indian Sign Language Recognition System for Deaf and Dumb People" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470,Volume-2Issue-2Februar2018,

[12]. Sumit saha,”A Comprehensive Guide to Convolutional NeuralNetworktheELI5way.”,Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional -neural-

networks-the-eli5-way-3bd2b1164a53Accessed Jan 31, 2020]

[13]. "Review of Different Deep Learning Approaches for Image Classification", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.6, Issue 3, page no.378-384, March-2019

[14]. Aishwarya," Introduction to Recurrent Neural Network", Available: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/Accessed Jan 31, 2020]

[15]. Long Đỗ," Recurrent Neural Network And Long-Short Term Memory", Available: https://ai.hblab.vn/2019/04/recurrent-neural-network-and-long-short.htmlAccessed Jan 31, 2020]

[16]. Arunava,"ConvolutionalNeuralNetwork",Available:https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05 Accessed Jan 31, 2020]

[17]. Mitchell, Ross; Young, Travas; Bachleda, Bellamie; Karchmer, Michael (2006). "How Many People Use ASL in the United States?: Why Estimates Need Updating" (PDF). Sign Language Studies (Gallaudet University Press.) 6 (3). ISSN 0302-1475. Retrieved November 27, 2012.

[18]. Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2014.Lifeprint.com. American Sign Language (ASL) Manual Alphabet (fingerspelling) 2007

[19]. J. Atwood, M. Eicholtz, and J. Farrell. American Sign Language Recognition System. Artificial Intelligence and Machine Learning for Engineering Design. Dept. of Mechanical Engineering, Carnegie Mellon University, 2012

Authors Profile

Prof. Pradyumna P. Kulkarni pursued Bachelor of Engineering from SGBAU Amravati University of Maharatsra, in 2009 and Babasaheb Ambedkar Marathwada University Aurangabad 2012. He is currently working as Assistant Professor in Department of Computer Science & Engineering, at Anuradha Engineering College Chikhli Since Dec 2012 M.S. India

Mr Suraj S Bhute pursuing Bachelor of Engineering in Computer Science & Engineering Department from Anuradha Engineering College of SGBAU Amravati *University Maharashtra*

Mr Akash P Wagh pursuing Bachelor of Engineering in Computer Science & Engineering Department from Anuradha Engineering College of SGBAU Amravati *University Maharashtra*