Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 12, No. 4, December : 2023

SIGN LANGUAGE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS

P V S N R Chakradhari M.Tech Scholar in CSE,Aditya Engineering College(A), Surampalem, Kakinada, India-433537
G. Aparna Associate Professor in CSE,Aditya Engineering College(A),Surampalem, Kakinada, India-433537

ABSTRACT: Sign Language Recognition (SLR) targets on interpreting the sign language into text or speech, so as to facilitate the communication between deaf-mute people and ordinary people. This task has broad social impact, but is still very challenging due to the complexity and large variations in hand actions. Existing methods for SLR use hand-crafted features to describe sign language motion and build classification models based on those features. However, it is difficult to design reliable features to adapt to the large variations of hand gestures. To approach this problem, we propose a novel convolutional neural network (CNN) which extracts discriminative spatial-temporal features from raw video stream automatically without any prior knowledge, avoiding designing features. To boost the performance, multi-channels of video streams, including color information, depth clue, and body joint positions, are used as input to the CNN in order to integrate color, depth and trajectory information. We validate the proposed model on a real dataset collected with Microsoft Kinect and demonstrate its effectiveness over the traditional approaches based on hand-crafted features.
Key words: Sign Language Recognition, Convolutional Neural Network (CNN), Hand Actions, Deaf, Mute.

1. Introduction

Sign language, as one of the most widely used communication means for hearing-impaired people, is expressed by variations of hand-shapes, body movement, and even facial expression. Since it is difficult to collaboratively exploit the information from hand-shapes and body movement trajectory, sign language recognition is still a very challenging task. This paper proposes an effective recognition model to translate sign language into text or speech in order to help the hearing impaired communicate with normal people through sign language.
Technically speaking, the main challenge of sign language recognition lies in developing descriptors to express hand-shapes and motion trajectory. In particular, hand-shape description involves tracking hand regions in video stream, segmenting hand-shape images from complex background in each frame and gestures recognition problems. Motion trajectory is also related to tracking of the key points and curve matching. Although lots of research works have been conducted on these two issues for now, it is still hard to obtain satisfying result for SLR due to the variation and occlusion of hands and body joints. Besides, it is a nontrivial issue to integrate the hand-shape features and trajectory features together. To address these difficulties, we develop a CNNs to naturally integrate hand-shapes, trajectory of action and facial expression. Instead of using commonly used color images as input to networks like [1, 2], we take color images, depth images and body skeleton images simultaneously as input which are all provided by Microsoft Kinect.
Kinect is a motion sensor which can provide color stream and depth stream. With the public Windows SDK, the body joint locations can be obtained in real-time as shown in Fig.1. Therefore, we choose Kinect as capture device to record sign words dataset. The change of color and depth in pixel level are useful information to discriminate different sign actions. And the variation of body joints in time dimension can depict the trajectory of sign actions. Using multiple types of visual sources as input leads CNNs paying attention to the change not only in color, but also in depth and trajectory. It is worth mentioning that we can avoid the difficulty of tracking hands, segmenting hands from background and designing descriptors for hands because CNNs have the capability to learn features automatically from raw data without any prior knowledge [3].

CNNs have been applied in video stream classification recently years. A potential concern of CNNs is time consuming. It costs several weeks or months to train a CNNs with million-scale in million videos. Fortunately, it is still possible to achieve real-time efficiency, with the help of CUDA for parallel processing. We propose to apply CNNs to extract spatial and temporal features from video stream for Sign Language Recognition (SLR). Existing methods for SLR use hand-crafted features to describe sign language motion and build classification model based on these features. In contrast, CNNs can capture motion information from raw video data automatically, avoiding designing features. We develop a CNNs taking multiple types of data as input. This architecture integrates color, depth and trajectory information by performing convolution and subsampling on adjacent video frames. Experimental results demonstrate that 3D CNNs can significantly outperform Gaussian mixture model with Hidden Markov model (GMM-HMM) baselines on some sign words recorded by ourselves.

2. Literature Review

The review of literature focuses on techniques used for gesture recognition and the color spaces used while detecting different colors in the surrounding environment. The survey done in provides a good knowledge about various color models used for detection. It is a review paper for different color models used as well as the mathematical representation of each with their corresponding advantages and disadvantages along with their comparison and their suitable application area. We will use RGB to HSV color conversion algorithm as HSV is better suited for such operations. The conversion from RGB to HSV takes time for higher resolution images or higher Frames per second applications. [1].
An approach in narrates skin detection in HSV color space. In order to detect skin from an RGB image it is first converted to HSV as it can be perceived closely as human colors. RGB to HSV conversion is done using values ranging from 6 to 38 for Hue and mixture of different filters to detect skin color. Next step is thresholding where non skin pixels were assigned value 0 and skin pixels 1. Dilate and erode of kernel size 5x5 is used to soften the skin image to a certain extent. A median filter with kernel size 3x3 to soften the image. Now, only the skin region will appear as white pixels and all the other pixels are represented as black pixels. The openCV library was used for image processing and the system was developed in C language. [2]
Image filtering algorithms are needed to filter out noise which is the main focus of the paper. Filtering is required to reduce the noise and improve the visual quality of the image. It gives a detailed explanation about the various filtering techniques. The proposed system will use mean filtering as it removes noise while preserving the edge which is crucial as we need to detect contours of the hand. [3]
The adaptive boosting for hand detection and Haar cascade classifier algorithm to train the classifier was implemented in the system. It uses HSV color model for background subtraction and noise removal, convex hull algorithm for drawing contours around the palm and fingertip detection. A laptop webcam of resolution 480p was used to capture the stream. OpenCV and C++ were used to implement this system. [4]
Boundary detection algorithm of an object was proposed in this paper. The algorithm finds a detailed boundary that includes object’s outer border also known as 1-component. It also consists of a hole-border between the hole and the 1-component surrounding it directly. This can be modified for detecting convex and concave parts of the hand to detect the contours. [5]
Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a nonlinearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply. [6]

3. Methodology

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 12, No. 4, December : 2023

3.1 Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

3.2 Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

3.3 Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3.4 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

3.5 Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

3.6 Detailed Design

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.

Goals:

The Primary goals in the design of the UML are as follows:

Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningfulmodels.

Provide extendibility and specialization mechanisms to extend the coreconcepts.

Be independent of particular programming languages and developmentprocess.

Provide a formal basis for understanding the modelinglanguage.

Encourage the growth of OO toolsmarket. Support higher level development concepts such as collaborations, frameworks, patterns and components.

Use diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

User use case Diagram

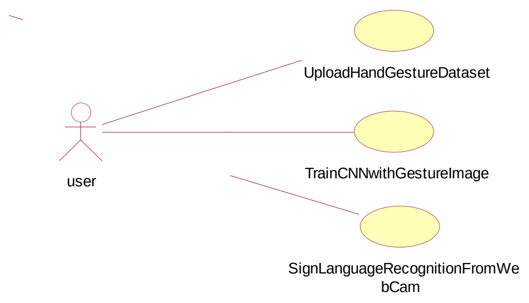
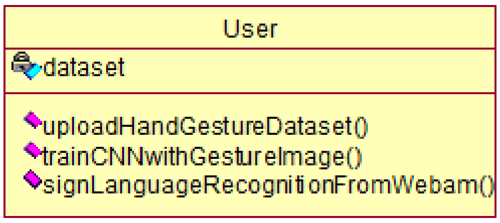


Figure 1: Use diagram

Class


User
dataset
uploadHandGestureDataset()
trainCNNwithGestureImage()
signLanguageRecognitionFromWebam()
Figure 2: Class diagram

Sequence

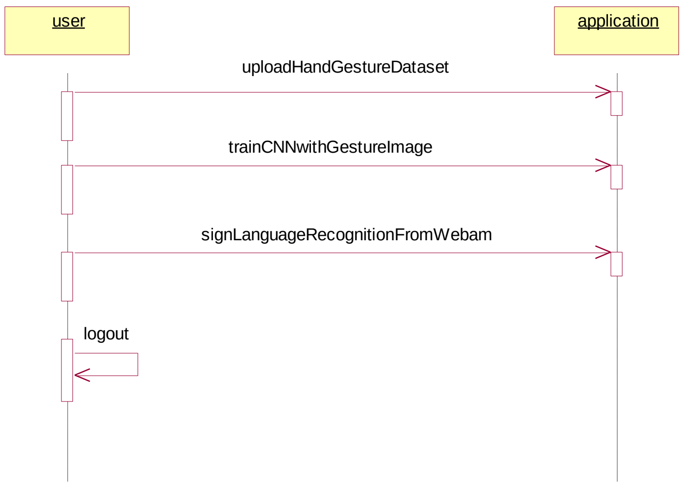


Figure 3: Sequence diagram

4. Open CV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.
The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both **academic** and **commercial** use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Look at the following images

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 12, No. 4, December : 2023

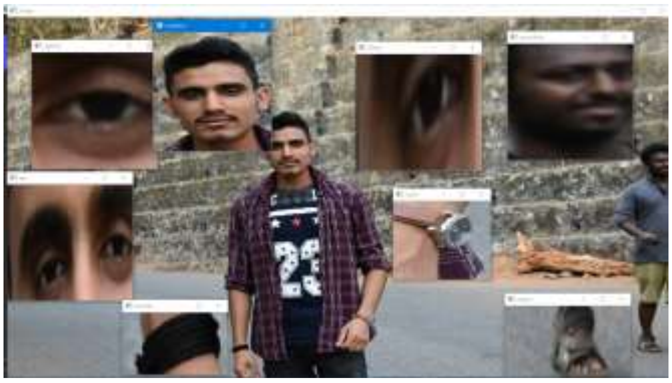


Figure 4: Open CV

from the above original image, lots of pieces of information that are present in the original image can be obtained. Like in the above image there are two faces available and the person(I) in the images wearing a bracelet, watch, etc so by the help of OpenCV we can get all these types of information from the original image.
It's the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles.

Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below:

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality:

Image/video I/O, processing, display (core, imgproc, highgui)

- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

5. Implementation

In this project using CNN we are recognizing hand gesture movement and to train CNN we are using following images shown in below screen shots

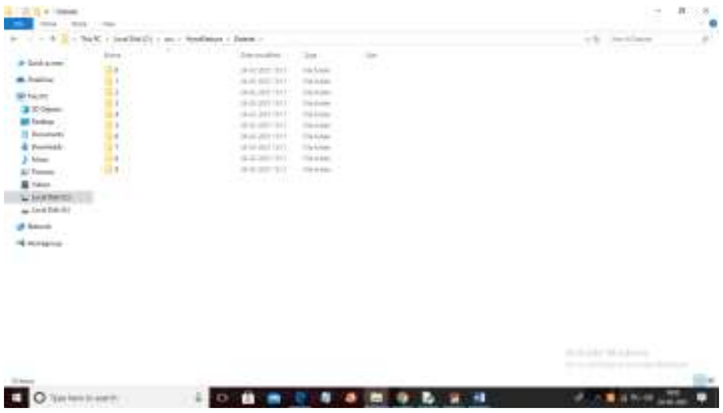


Figure 5: Loading data

In above screen we can see we have 10 different types of hand gesture images and to see those images just go inside any folder

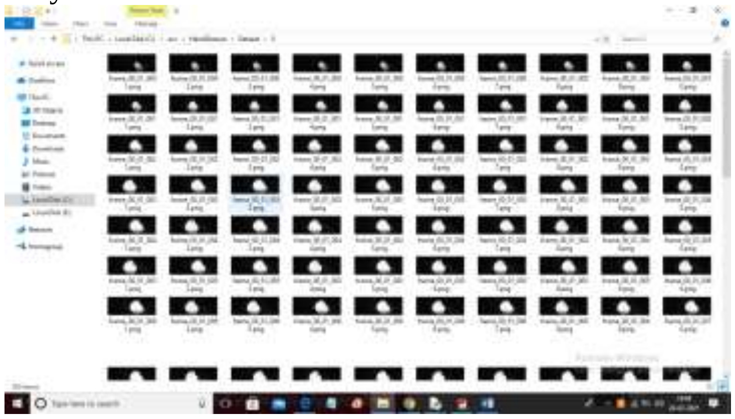
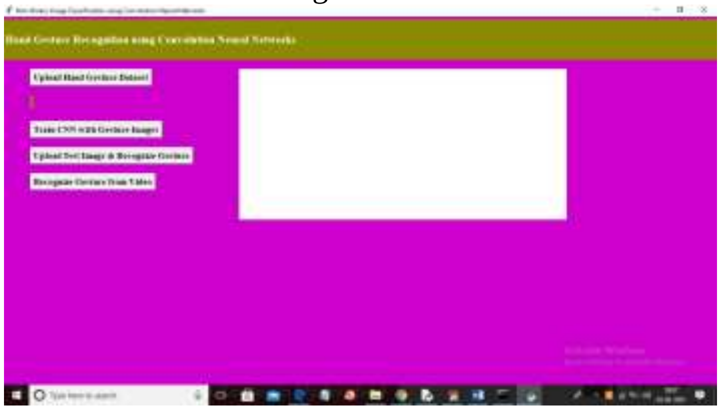


Figure 6: Uploading gestures

In above screen showing images from 0 folder and similarly you can see different images in different folders.

To run project double click on run.bat file to get below screen


Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video
Figure 7: Head gesture using CNN

In above screen click on ‘Upload Hand Gesture Dataset’ button to upload dataset and to get below screen

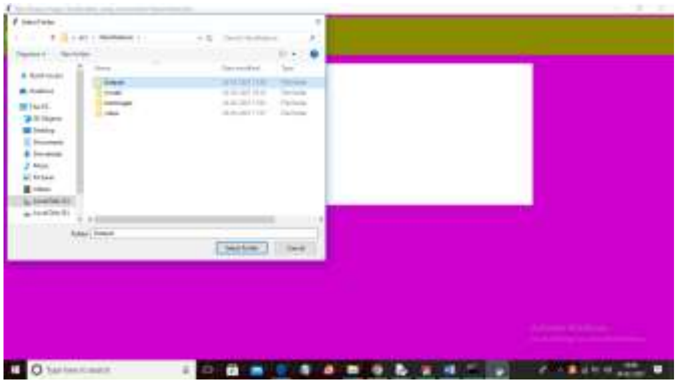


Figure 8: Loading data

In above screen selecting and uploading 'Dataset' folder and then click on 'Select Folder' button to load dataset and to get below screen

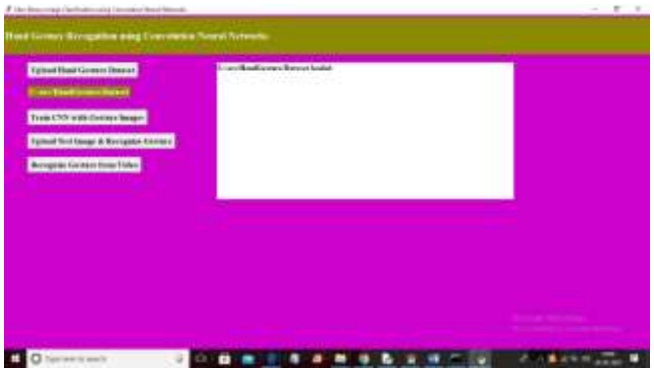
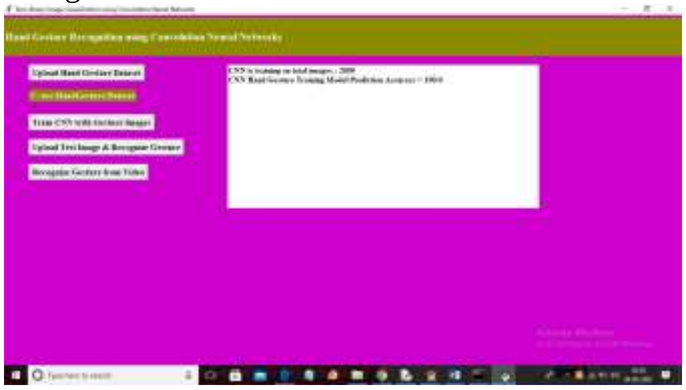

Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video

Figure 9: Train CNN with gesture images'

In above screen dataset loaded and now click on 'Train CNN with Gesture Images' button to trained CNN model and to get below screen


Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video

CNN is training on total images : 2000
CNN Hand Gesture Training Model Prediction Accuracy = 100.0

Figure 10: CNN model trained on 2000 images

In above screen CNN model trained on 2000 images and its prediction accuracy we got as 100% and now model is ready and now click on 'Upload Test Image & Recognize Gesture' button to upload image and to gesture recognition

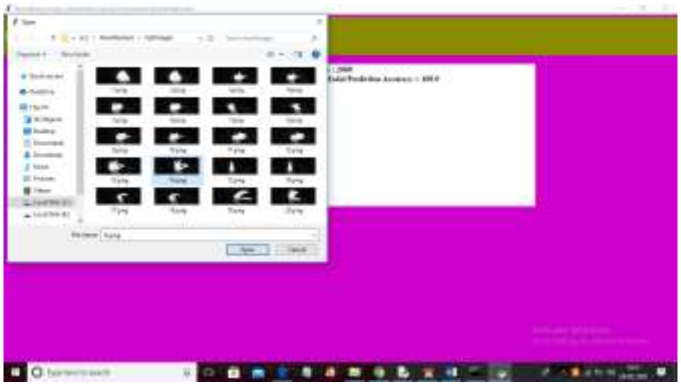
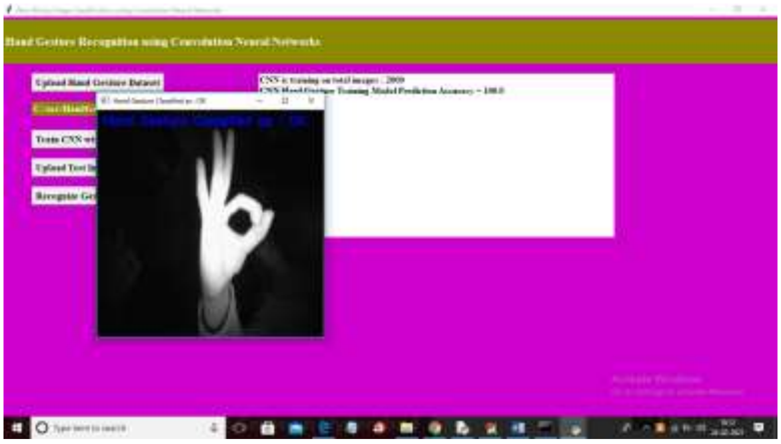


Figure 11: uploading '14.png' file

In above screen selecting and uploading '14.png' file and then click Open button to get below result


Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
Train CNN wi
Upload Test Im
Recognize Ge

CNN is training on total images : 2000
CNN Hand Gesture Training Model Prediction Accuracy = 100.0

Figure 12: Gesture recognize as OK

In above screen gesture recognize as OK and similarly you can upload any image and get result and now click on 'Recognize Gesture from Video' button to upload video and get result

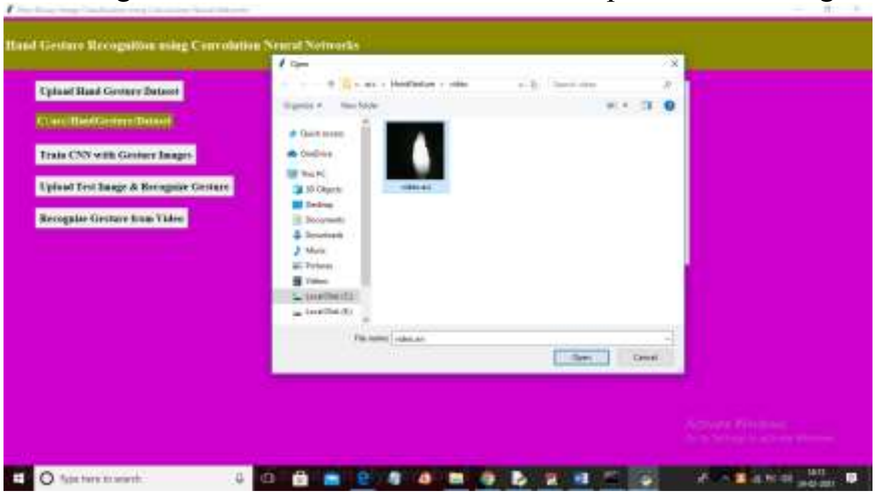

Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
C:/arc/HandGesture/Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video
Open
video.avi
Open
Cancel

Figure 13: Uploading 'video.avi' file

In above screen selecting and uploading 'video.avi' file and then click on 'Open' button to get below result

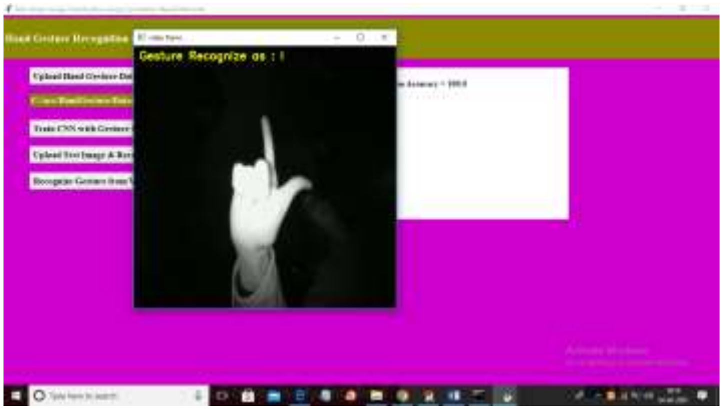


Figure 14: recognition result

In above screen as video play then will get recognition result

6. Results and discussions

We started out by building our own Convolutional Neural Network and testing it on a static gesture database. The results were astonishing and we got an accuracy of about 91%.

We then used the same Convolutional Neural Network for image sequences, which we would later process as time sequences, but it wasn't learning the gestures, instead was learning the people and faces. We then tried to train it by changing a few hyperparameters to try to impart the gestures on to the network (54%).

This wasn't good either as the data wasn't clean and had other gestures in between the current gesture data. We then sifted through the images and cleaned it to contain only the gestured part of the video. This was a tremendous improvement (61%) as it was starting to learn and identify the gestures.

For processing the image sequences as a time series, we tried using 3D Convolutional Networks which wasn't working due to memory constraints. To tackle these issues we tried to batch the sequences, which still did not work and the memory constraints still remained. We then tried to implement LSTM networks and found out that a single layer LSTM network is more than enough and does the job well. We later tried out retraining the Inception V3 model as it already had pre-trained weights and updating those weights would be much more efficient than training an entire model from scratch. We found out that the model was learning the gestured parts and giving out a decent accuracy (63%). After passing it through the LSTM network, the accuracy came out to be about 81% and we could identify the gestures live pretty well. The current solution is the best because it correctly identifies the gestures by extracting the features and correctly recognizing the sequences in which the gesture takes place.

7. Conclusion

We developed a CNN model for sign language recognition. Our model learns and extracts both spatial and temporal features by performing 3D convolutions. The developed deep architecture extracts multiple types of information from adjacent input frames and then performs convolution and subsampling separately. The final feature representation combines information from all channels. We use multilayer perceptron classifier to classify these feature representations. For comparison, we evaluate both CNN and GMM-HMM on the same dataset. The experimental results demonstrate the effectiveness of the proposed method.

References

[1] Ibraheem, N.A., Hasan, M.M., Khan, R.Z. and Mishra, P.K., 2012. Understanding color models: a review. ARPN Journal of Science and Technology, 2(3), pp.265-275. [
2] Oliveira, V.A. and Conci, A., 2009. Skin Detection using HSV color space. In H. Pedrini, & J. Marques de Carvalho, Workshops of Sibgrapi (pp. 1-2).

[3] Chandel, R. and Gupta, G., 2013. Image filtering algorithms and techniques: A review. International Journal of Advanced Research in Computer Science and Software Engineering, 3(10).
[4] Gurav, R.M. and Kadbe, P.K., 2015, May. Real time finger tracking and contour detection for gesture recognition using OpenCV. In Industrial Instrumentation and Control (ICIC), 2015 International Conference on (pp. 974-977). IEEE.
[5] Suzuki, S., 1985. Topological structural analysis of digitized binary images by border following. Computer vision, graphics, and image processing, 30(1), pp.32-46.
[6] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes inComputer Science, vol 8925. Springer, Cham.
[7] Andrej Karpathy. Stanford university cs231n: Convolutional neural networks for visual recognition. http://cs231n.stanford.edu/, March 24 2017. [Online].
[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015
[9] Kouichi Murakami and Hitomi Taguchi, "Gesture recognition using recurrent neural networks," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 1991, pp. 237–242.
[10] Chung-Lin Huang and Wen-Yi Huang, "Sign language recognition using model-based tracking and a 3D hopfield neural network," Machine vision and applications, vol. 10, no. 5-6, pp. 292–307, 1998.
[11] Jong-Sung Kim, Won Jang, and Zeungnam Bien, "A dynamic gesture recognition system for the korean sign language (ksl)," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 26, no. 2, pp. 354–359, 1996.
[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," arXiv preprint arXiv:1311.2524, 2013.
[13] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in ICML. ACM, 2008, pp. 160–167.
[14] Clement Farabet, Camille Couprie, Laurent Najman, ´ and Yann LeCun, "Learning hierarchical features for scene labeling," IEEE TPAMI, vol. 35, no. 8, pp. 1915– 1929, 2013. [15] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung, "Convolutional networks can learn to generate affinity graphs for image segmentation," Neural Computation, vol. 22, no. 2, pp. 511– 538, 2010.
[16] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li, "A real-time hand posture recognition system using deep neural networks," ACM Transactions on Intelligent Systems and Technology, 2014.
[17] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt, "Sequential deep learning for human action recognition," in Human Behavior Understanding, pp. 29–39. Springer, 2011.