Gesture Recognition Using Deep Learning

Herick Asmani
Illinois Institute of Technology, Chicago
A20399752
hasmani@hawk.iit.edu

Malhar Patwari
Illinois Institute of Technology, Chicago
A20410420
mpatwari@hawk.iit.edu

I. PROBLEM STATEMENT

You must have wondered what it would be like to control the TV, computer or any other device at your home with just a wave of your hand. These sci-fi dreams are becoming a reality with the advancement in the field of Deep Learning and their application in Computer Vision. Such vision-based hand gesture recognition also finds its application in advanced driver assistance systems (ADASs) which enables the driver to interact with the vehicle without affecting their concentration. Based on the above motivations, our aim is to build a system such that Given a real time video sequence of a hand gesture, the system should be able to interpret what that particular hand gesture signifies. To achieve that we are using Deep Learning models, as they provide end-to-end modeling as compared to traditional task specific feature engineered Machine Learning models such as SVM, etc. However, the video-based gesture recognition problem is not trivial and is challenging due to: the intra and inter-persons variations in human hand gesture motion; inter-person variations in the shape and size of the human hand; illumination variations; and background noise.

II. RELATED WORK

Hand gesture recognition is an important research area and has received significant attention in literature as described in the following survey [3]. In the standard vision-based hand gesture recognition framework, spatio-temporal features are first extracted from the video frames [4]. The inter-frame dynamics of these extracted features are then modeled using action classifiers such as the support vector machines (SVM) [5], neural network [6] or compressed sensing [7]. Dardas et al. [5] extract SIFT-based bag-of-word features from the input image. These discriminative features are then used to train a multiclass SVM-based gesture classifier. A similar work is reported by Ahmed et al. [6], where the image moments-based feature are used to train an artificial neural network-based classifier. The standard vision-based hand gesture recognition algorithms are limited by varying illumination and inter and intra-person appearance and motion variations. To address these issues, and enhance the classification accuracy, researchers have utilized multi-modal sensors to perform the gesture recognition [8]– [10]. For example, in the work by Neverova et al. [9] and Ohn-Bar et al. [10], color and depth sensors (RGB-D) are utilized perform the gesture recognition. In [10], the authors extract HOG features from the RGB-D sensors, which are then used to train a SVM classifer. Apart from RGB-D sensors, researchers have also utilized depth, colour and radar sensors to enhance the classification accuracy [8].

In recent years, deep learning-based classification frameworks have reported state-of-the-art results in various image based classification tasks [11]. Consequently, researchers have sought to adapt the image-based deep learning framework to perform activity recognition [12]–[15]. Two categories of literature exist for the deep learning-based activity recognition algorithms. In the first category, the deep learning framework is trained on the input data without explicitly modeling the inter-frame temporal information. Typically, multiple CNNs are adopted and trained with multiple scale data acquired from multiple sensors [12]–[14]. For example, Karpathy et al. [13] propose a multiresolution CNN architecture that are trained with original and cropped video frames to perform action recognition. A similar multiple network-based CNN is reported by Molchanov et al. [12], where the color and depth channels are combined and used to train two resolutionbased CNN subnetworks. In the work by Neverova et al. [9], a multi-scale and multimodal deep learning framework is utilized. Multi-scale gesture information from the motion capture system along with the video are provided as inputs to multiple CNNs. The output of the different CNNs are fused by a multilayered perceptron in the final layer, and the activity is classified. A multiscale CNN is also adopted by Pigou et al. [14] to classify the American sign language. In their work, multiple frames of RGB and depth data derived from videos containing hand gestures and full body gestures are given as an input to two CNN networks to perform the classification. Finally, in the work by Simonyan et al. [15], two CNN architectures are trained on the video and the optical flow, derived from the video, respectively.

In the second category of deep learning-based activity recognition, researchers model the inter-frame temporal information using the recurrent neural network (RNN) [16], [17]. Murukami et al. [16] utilize the RNN to classify Japanese sign languages, while Ordonez et al. [17] utilize the long-term short memory-based RNN (LSTM) to classify gestures from accelerometer data. Recently, Donahue et al. [1] integrate the CNN within the LSTM to formulate the long recurrent convolutional neural network (LRCN). In the LRCN, multiple frames are sampled from a longer video sequence and given as inputs to multiple CNN, with each CNN obtaining an input from an individual frame. The outputs of the multiple CNN are then given as an input to the LSTM to predict the class label. The LSTM-based action classifiers report state-of the-art classification accuracy on different activity recognition problems.

III. DATASET

The dataset used for this project is Cambridge Hand Gesture Dataset. The data set consists of 900 image sequences of 9 gesture classes, which are defined by 3 primitive hand shapes and 3 primitive motions as shown in figure below. Therefore, the target task for this data set is to classify different shapes as well as different motions at a time. Each class contains 100 image sequences (5 different illuminations x 10 arbitrary motions x 2 subjects). Each sequence was recorded in front of a fixed camera having roughly isolated gestures in space and time. Thus, fairly large intra-class variations in spatial and temporal alignment is reflected to the data set.

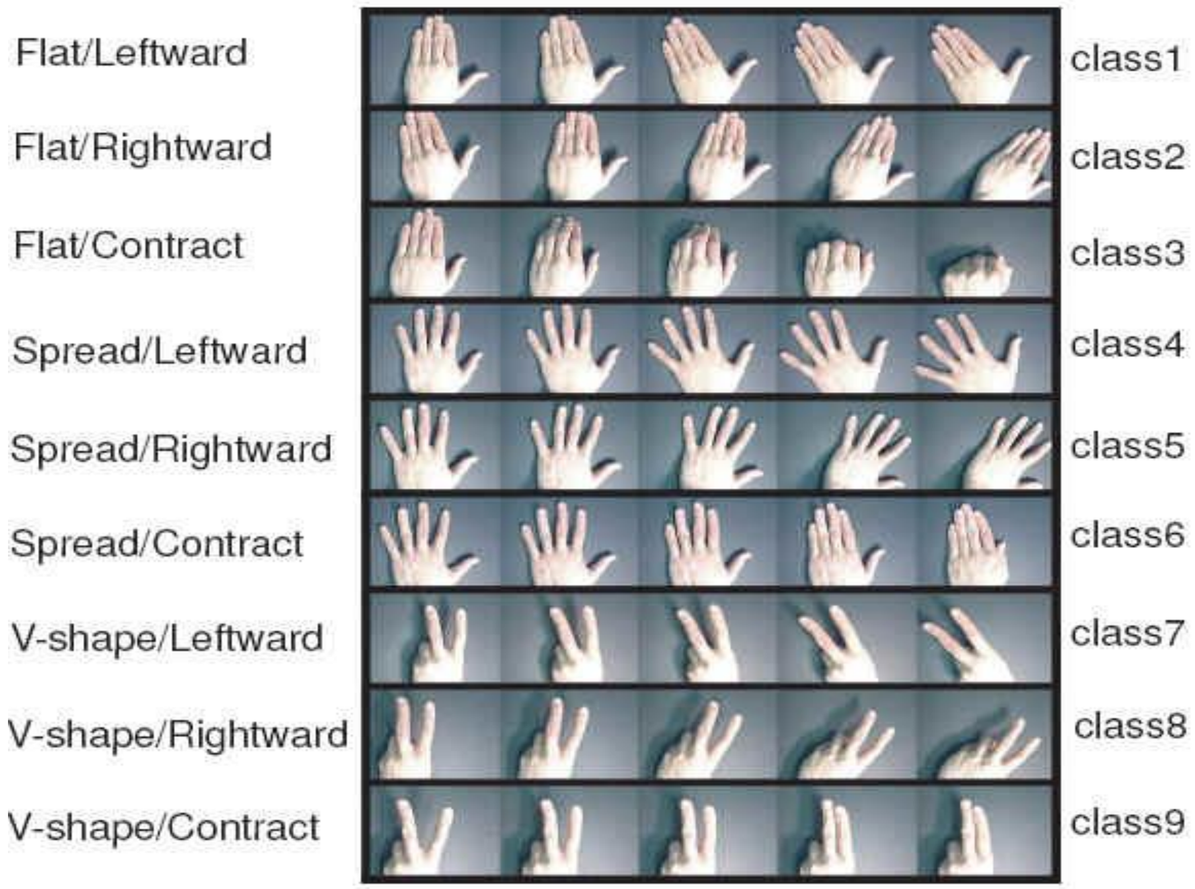


Fig. Sample sequences of the 9 gesture classes.

IV. PROPOSED APPROACH

The main technique used for this project is the Long term Recurrent Convolutional Network which is the combination of multiple Convolutional Neural Networks (CNNs) and Long Short Term Memory Recurrent Neural Networks (LSTM RNNs). The video frames are given as input to the CNNs (one frame per CNN) which then outputs feature vectors per frame and those feature vectors serve as input to the LSTM which will then output the probability distribution of labels. The reason why LSTMs are used is because they provide temporal dependencies which is essential for video classification. Therefore, CNNs acts as a encoder and LSTM acts as a decoder.

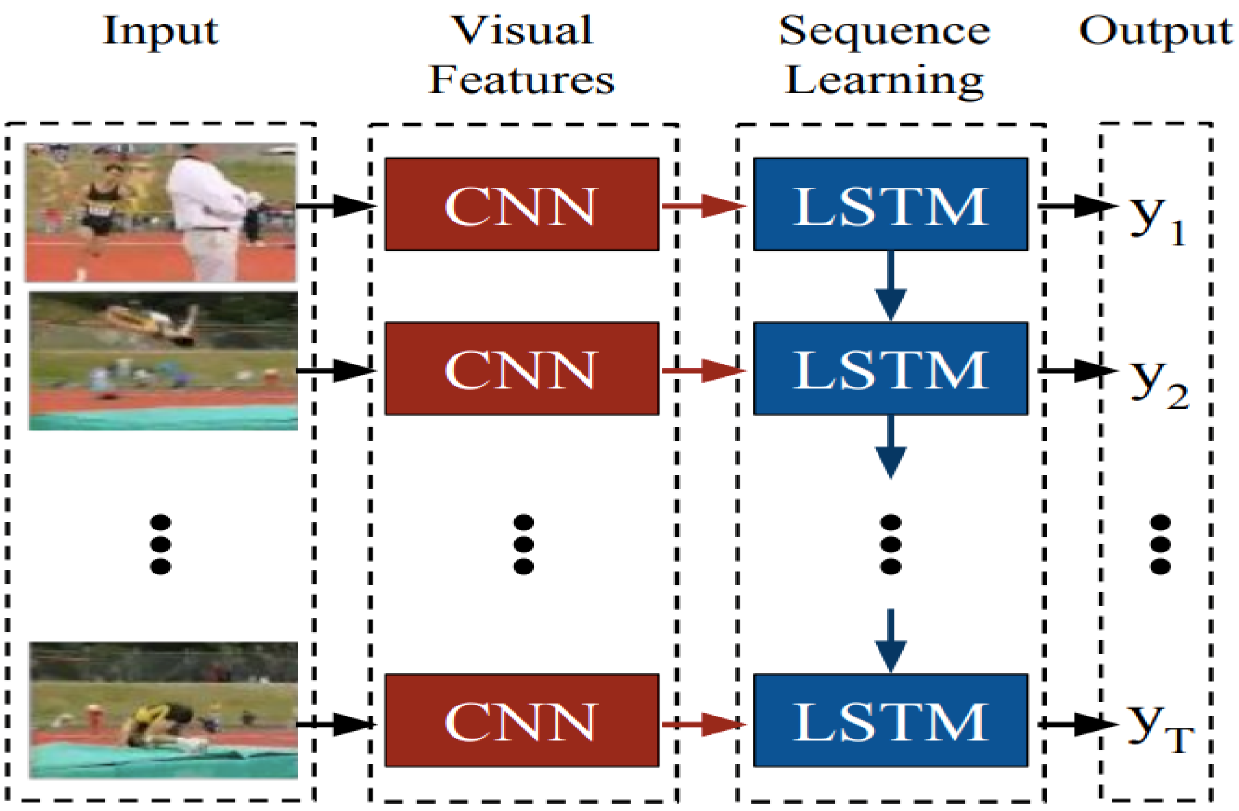
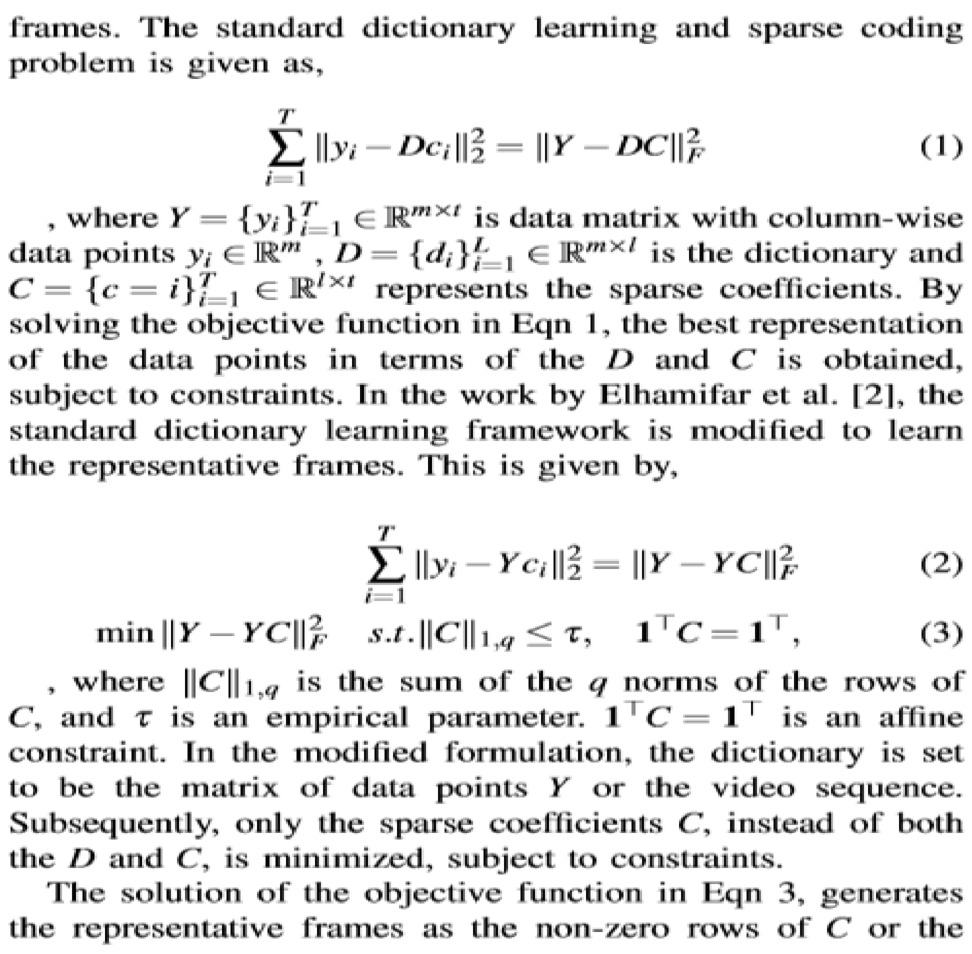


Fig. Long Term Recurrent Convolutional Network

The paper we are referring, in the training phase uses tiled image pattern of 16 sampled frames which is given to the Sparse Modeling (SMRF) algorithm to obtain 3 Representative frames which is converted to a tiled binary pattern. They have used both of these patterns (tiled image pattern and tiled binary pattern) to fine tune the pretrained Deconvolution Network. The obtained 3 representative frames are then given to the pre-trained Long term Recurrent Convolutional Network for fine tuning. In the testing phase, they are doing same sampling of frames as in training phase, converting it to tiled image pattern but now for extracting the representative frames they have used the fine-tuned Deconvolution Network. The output of Deconvolution Network is tiled binary pattern based on which the corresponding 3 representative frames are obtained which is then given to the fine-tuned Long term Recurrent Convolutional Network to classify the gestures.

Our approach is as follows:
Training Phase:
Based on each class label, all the corresponding video frames are given to the Sparse Modeling (SMRF) algorithm and instead of 3 we are extracting 5 representative frames. According to our reference paper, following algorithm is used to obtain the representative frames.


frames. The standard dictionary learning and sparse coding problem is given as,

$$\sum_{i=1}^{T} \|y_i - Dc_i\|_2^2 = \|Y - DC\|_F^2 \qquad (1)$$

, where $Y = \{y_i\}_{i=1}^{T} \in \mathbb{R}^{m\times t}$ is data matrix with column-wise data points $y_i \in \mathbb{R}^m$, $D = \{d_i\}_{i=1}^{L} \in \mathbb{R}^{m\times l}$ is the dictionary and $C = \{c = i\}_{i=1}^{T} \in \mathbb{R}^{l\times t}$ represents the sparse coefficients. By solving the objective function in Eqn 1, the best representation of the data points in terms of the D and C is obtained, subject to constraints. In the work by Elhamifar et al. [2], the standard dictionary learning framework is modified to learn the representative frames. This is given by,

$$\sum_{i=1}^{T} \|y_i - Yc_i\|_2^2 = \|Y - YC\|_F^2 \qquad (2)$$

$$\min \|Y - YC\|_F^2 \quad s.t. \|C\|_{1,q} \leq \tau, \quad \mathbf{1}^\top C = \mathbf{1}^\top, \qquad (3)$$

, where $\|C\|_{1,q}$ is the sum of the q norms of the rows of C, and τ is an empirical parameter. $\mathbf{1}^\top C = \mathbf{1}^\top$ is an affine constraint. In the modified formulation, the dictionary is set to be the matrix of data points Y or the video sequence. Subsequently, only the sparse coefficients C, instead of both the D and C, is minimized, subject to constraints.

The solution of the objective function in Eqn 3, generates the representative frames as the non-zero rows of C or the

Based on obtained 5 representative frames, the corresponding original frames are selected for each video and then data is converted to a numpy array based on 5 frames.

Data Preprocessing is done by normalizing the frames dimensions so that they are approximately the same scale. We have done this by dividing each dimension by its standard deviation, once it has been zero-centered i.e. subtracted the mean across every individual feature in data. After normalizing the data, we have shuffled the data to reduce bias. We then divided the data into three parts i.e. training (700), validation (100), test (100). Also, the labels are one hot encoded.

The preprocessed training data is then fed to the Long term Recurrent Convolutional Network(LRCN) as shown in above figure for training and is validated on validation set. We are training the LRCN from scratch unlike that proposed in paper.

Testing Phase:
Part 1:
The trained model is evaluated on the test data (100).

Part 2:
An unseen real time video is used. The video is converted to frames and the same SMRF algorithm is used to obtain the 5 representative frames and the frames are fed to the LRCN to obtain the gesture label of unseen real time video. And based on each label following computer applications are opened.

Label	Computer Application opened
0	Whatsapp
1	Google maps
2	Take Screenshot
3	Facebook
4	Gmail
5	Youtube
6	Calculator
7	Notepad
8	Play random music (given there is a music folder and the path is provided)

V. RESULTS

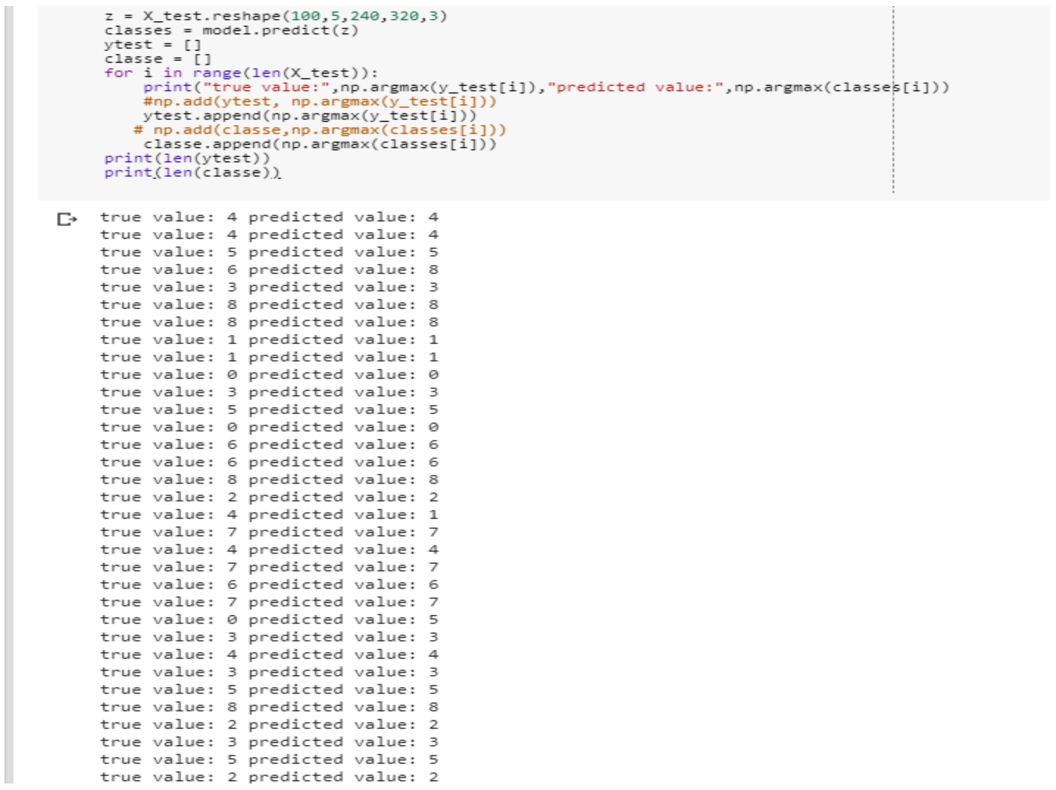
1. Best Model parameters with using SMRF

Data: Training : 700 videos , each having 5 frames
Validation : 700 videos , each having 5 frames
Testing : 700 videos , each having 5 frames

Parameter	Value
2D Convolution layer	5
Max Pooling layer	5
LSTM units	256
Optimizer	Adam
Dropout	0.5
Epoch	10
Batch Size	64

Train Accuracy:	99.29%
Validation Accuracy :	90%
Test Accuracy :	98%
Test Precision :	0.97 (avg of all classes)
Test Recall :	0.97 (avg of all classes)
Test F1 Score:	0.97 (avg of all classes)

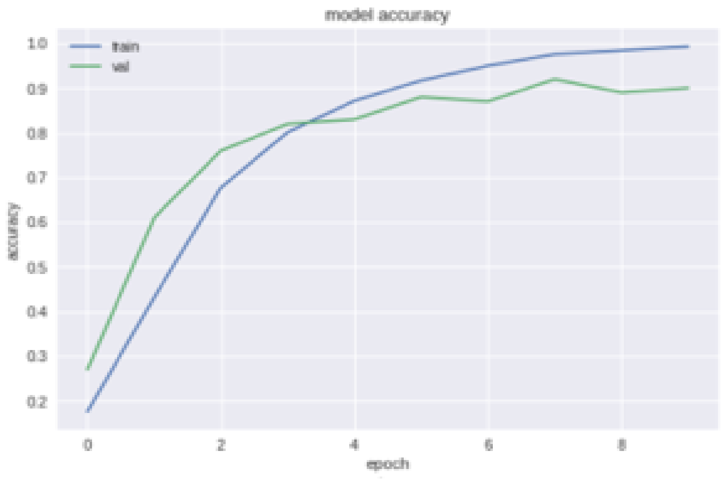
Comparison:

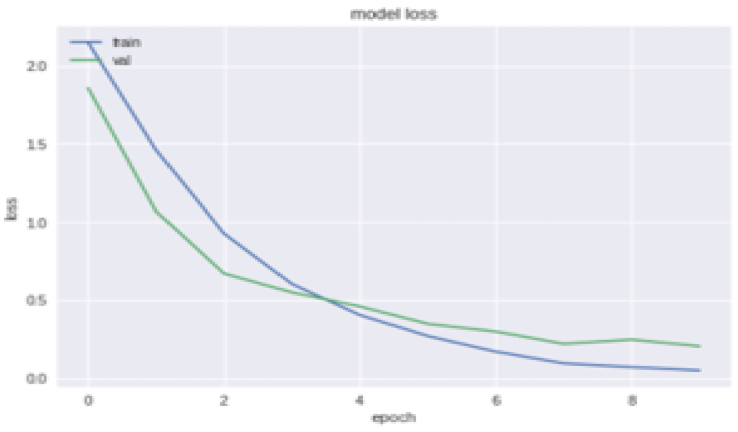


```
z = X_test.reshape(100,5,240,320,3)
classes = model.predict(z)
ytest = []
classe = []
for i in range(len(X_test)):
    print("true value:",np.argmax(y_test[i]),"predicted value:",np.argmax(classes[i]))
    #np.add(ytest, np.argmax(y_test[i]))
    ytest.append(np.argmax(y_test[i]))
   # np.add(classe,np.argmax(classes[i]))
    classe.append(np.argmax(classes[i]))
print(len(ytest))
print(len(classe))
```

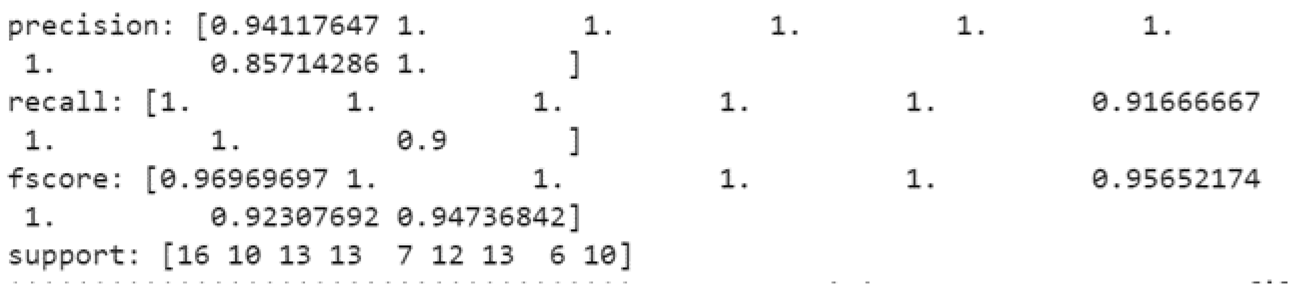
```
true value: 4 predicted value: 4
true value: 4 predicted value: 4
true value: 5 predicted value: 5
true value: 6 predicted value: 8
true value: 3 predicted value: 3
true value: 8 predicted value: 8
true value: 8 predicted value: 8
true value: 1 predicted value: 1
true value: 1 predicted value: 1
true value: 0 predicted value: 0
true value: 3 predicted value: 3
true value: 5 predicted value: 5
true value: 0 predicted value: 0
true value: 6 predicted value: 6
true value: 6 predicted value: 6
true value: 8 predicted value: 8
true value: 2 predicted value: 2
true value: 4 predicted value: 1
true value: 7 predicted value: 7
true value: 4 predicted value: 4
true value: 7 predicted value: 7
true value: 6 predicted value: 6
true value: 7 predicted value: 7
true value: 0 predicted value: 5
true value: 3 predicted value: 3
true value: 4 predicted value: 4
true value: 3 predicted value: 3
true value: 5 predicted value: 5
true value: 8 predicted value: 8
true value: 2 predicted value: 2
true value: 3 predicted value: 3
true value: 5 predicted value: 5
true value: 2 predicted value: 2
```

Graphs:





Precision , Recall, f1 score, support:



```
precision: [0.94117647 1.         1.         1.         1.         1.
 1.         0.85714286 1.        ]
recall: [1.         1.         1.         1.         1.         0.91666667
 1.         1.         0.9       ]
fscore: [0.96969697 1.         1.         1.         1.         0.95652174
 1.         0.92307692 0.94736842]
support: [16 10 13 13  7 12 13  6 10]
```

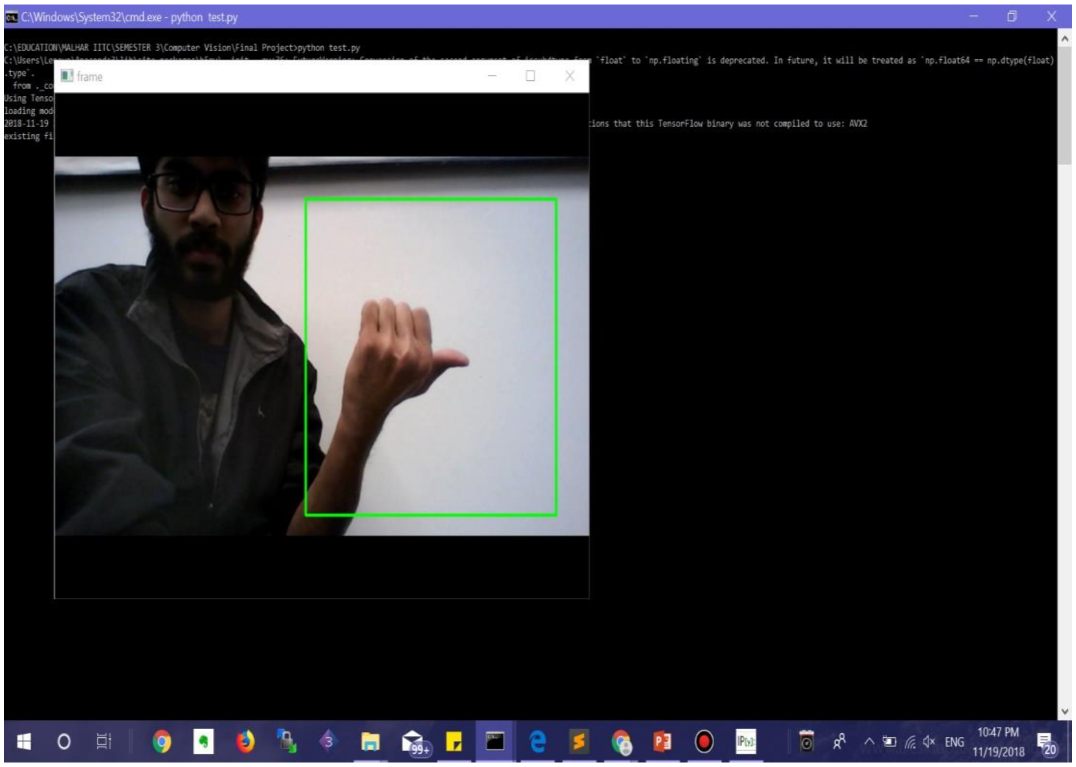
2. <u>Best Model parameters without using SMRF</u>
   Data: Training : 700 videos , each having 5 frames
   Validation : 700 videos , each having 5 frames
   Testing : 700 videos , each having 5 frames

Parameter	**Value**
2D Convolution layer	5
Max Pooling layer	5
LSTM units	256
Optimizer	Adam
Dropout	0.5
Epoch	10
Batch Size	64

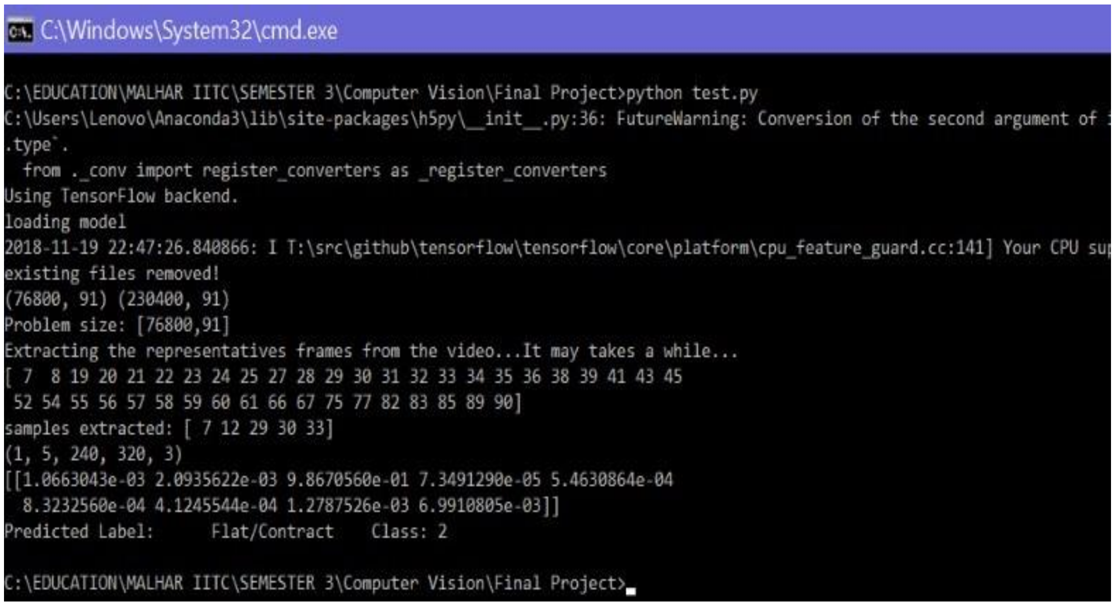
Train Accuracy:	90.12%
Validation Accuracy :	87.37%
Test Accuracy :	94.78%
Test Precision :	0.92 (avg of all classes)
Test Recall :	0.93 (avg of all classes)
Test F1 Score:	0.92 (avg of all classes)

3. <u>Application results</u>

For Label 2:

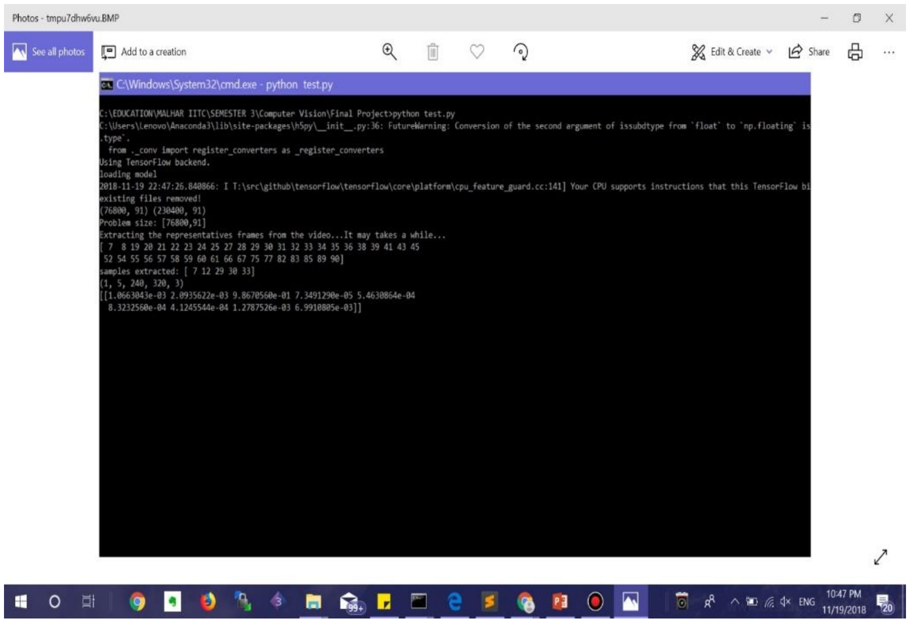


```
C:\EDUCATION\MALHAR IITC\SEMESTER 3\Computer Vision\Final Project>python test.py
C:\Users\Le                                                    `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float)
.type`.
  from ._co
Using Tenso
loading mod
2018-11-19                                                     ions that this TensorFlow binary was not compiled to use: AVX2
existing fi
```



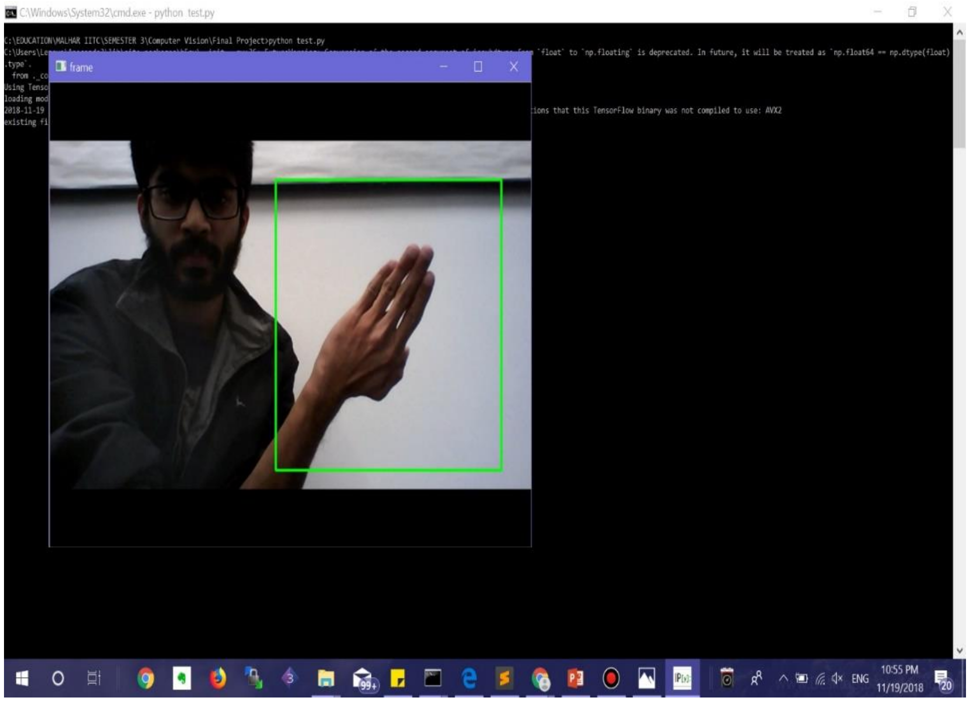
```
C:\EDUCATION\MALHAR IITC\SEMESTER 3\Computer Vision\Final Project>python test.py
C:\Users\Lenovo\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of i
.type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
loading model
2018-11-19 22:47:26.840866: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:141] Your CPU sup
existing files removed!
(76800, 91) (230400, 91)
Problem size: [76800,91]
Extracting the representatives frames from the video...It may takes a while...
[ 7  8 19 20 21 22 23 24 25 27 28 29 30 31 32 33 34 35 36 38 39 41 43 45
 52 54 55 56 57 58 59 60 61 66 67 75 77 82 83 85 89 90]
samples extracted: [ 7 12 29 30 33]
(1, 5, 240, 320, 3)
[[1.0663043e-03 2.0935622e-03 9.8670560e-01 7.3491290e-05 5.4630864e-04
  8.3232560e-04 4.1245544e-04 1.2787526e-03 6.9910805e-03]]
Predicted Label:      Flat/Contract    Class: 2

C:\EDUCATION\MALHAR IITC\SEMESTER 3\Computer Vision\Final Project>
```

Correctly classified the gesture as class label 2 and therefore took screenshot.



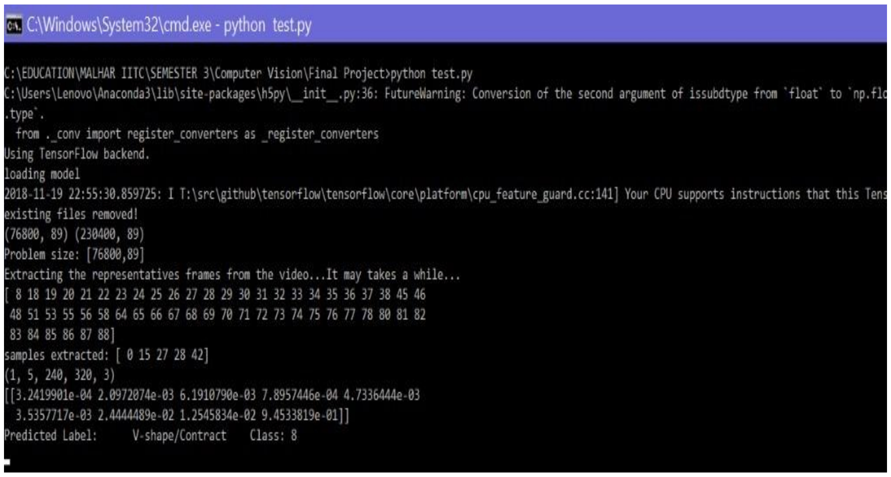
```
C:\EDUCATION\MALHAR IITC\SEMESTER 3\Computer Vision\Final Project>python test.py
C:\Users\Lenovo\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is
.type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
loading model
2018-11-19 22:47:26.840866: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow bi
existing files removed!
(76800, 91) (230400, 91)
Problem size: [76800,91]
Extracting the representatives frames from the video...It may takes a while...
[ 7  8 19 20 21 22 23 24 25 27 28 29 30 31 32 33 34 35 36 38 39 41 43 45
 52 54 55 56 57 58 59 60 61 66 67 75 77 82 83 85 89 90]
samples extracted: [ 7 12 29 30 33]
(1, 5, 240, 320, 3)
[[1.0663043e-03 2.0935622e-03 9.8670560e-01 7.3491290e-05 5.4630864e-04
  8.3232560e-04 4.1245544e-04 1.2787526e-03 6.9910805e-03]]
```

For Label 0:



```
C:\EDUCATION\MALHAR IITC\SEMESTER 3\Computer Vision\Final Project>python test.py
C:\Users\Le                                                        `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float)
.type`.
  from ._co
Using Tenso
loading mod
2018-11-19                                                          ions that this TensorFlow binary was not compiled to use: AVX2
existing fi
```

It incorrectly classified it as 8. The reason for misclassification is that model is trained on frames that are taken from different camera angle compared to laptop camera. Also, the shape of hand matters while classifying gestures.



```
C:\Windows\System32\cmd.exe - python  test.py

C:\EDUCATION\MALHAR IITC\SEMESTER 3\Computer Vision\Final Project>python test.py
C:\Users\Lenovo\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.flo
.type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
loading model
2018-11-19 22:55:30.859725: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:141] Your CPU supports instructions that this Tens
existing files removed!
(76800, 89) (230400, 89)
Problem size: [76800,89]
Extracting the representatives frames from the video...It may takes a while...
[ 8 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 45 46
 48 51 53 55 56 58 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 80 81 82
 83 84 85 86 87 88]
samples extracted: [ 0 15 27 28 42]
(1, 5, 240, 320, 3)
[[3.2419901e-04 2.0972074e-03 6.1910790e-03 7.8957446e-04 4.7336444e-03
  3.5357717e-03 2.4444489e-02 1.2545834e-02 9.4533819e-01]]
Predicted Label:      V-shape/Contract    Class: 8
```

VI. CONCLUSION AND FUTURE WORK

In this project, a Deep Learning based Hand Gesture Recognition algorithm is proposed for operating computer applications. The Long term Recurrent Convolutional Neural Network is utilized to perform the hand gesture recognition. The reasonable classification accuracy and computational efficiency of the long term recurrent convolutional neural network is obtained by extracting 5 representative frames from the video sequence. Our proposed algorithm is evaluated on the Cambridge public dataset. In our future work, we will evaluate with a larger dataset containing flow images along with RGB images in order to improve the robustness of the system. With larger dataset pre-trained model such as VGG19, INCEPTION V3, etc will be used as CNNs.

VII. CITATIONS

[1] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in CVPR, 2015.
[2] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in IEEE Conference on Computer Vision and Pattern Recognition, 2012.
[3] S. Mitra and T. Acharya, "Gesture recognition: A survey," IEEE Transactions on Systems, Man and Cybernetics - Part C, vol. 37, no. 3, pp. 311–324, 2007.
[4] P. Trindade, J. Lobo, and J. P. Barreto, "Hand gesture recognition using color and depth images enhanced with hand angular pose data," in IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2012.
[5] N. H. Dardas and N. D. Georganas, "Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques." IEEE Trans. Instrumentation and Measurement, vol. 60, no. 11, pp. 3592–3607, 2011.
[6] T. Ahmed, "A neural network based real time hand gesture recognition system," International Journal of Computer Applications, vol. 59, no. 4, pp. 17–22, December 2012.
[7] A. Boyali and N. Hashimoto, "Block-sparse representation classification based gesture recognition approach for a robotic wheelchair," in 2014 IEEEIntelligentVehiclesSymposiumProceedings,2014,pp.1133–1138.
[8] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, 2015.
[9] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout, Multi-scale Deep Learning for Gesture Detection and Localization, 2015.
[10] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 6, pp. 2368–2377, Dec 2014.
[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in Proceedings of Neural Information Processing Systems, 2012.
[12] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015.

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014.
[14] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, Sign Language Recognition Using Convolutional Neural Networks, 2015.
[15] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” 2014.
[16] K. Murakami and H. Taguchi, “Gesture recognition using recurrent neural networks,” in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1991.
[17] F. J. Ordez and D. Roggen, “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition,” Sensors, vol. 16, no. 1, p. 115, 2016.
[18] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation.” in ICCV, 2015.

VIII. REFERENCES

1. Deep Learning-Based Fast Hand Gesture Recognition Using Representative Frames. Vijay John ; Ali Boyali ; Seiichi Mita ; Masayuki Imanishi ; Norio Sanma. 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)
2. Hand gesture recognition using deep learning. Soeb Hussain ; Rupal Saxena ; Xie Han ; Jameel Ahmed Khan ; Hyunchul Shin. 2017 International SoC Design Conference (ISOCC)
3. https://keras.io/layers/
4. https://github.com/DavideNardone/PySMRS
5. https://people.eecs.berkeley.edu/~lisa_anne/LRCN_video
6. https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/
7. http://cs231n.stanford.edu/reports/2017/pdfs/709.pdf
8. http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review
9. https://blog.coast.ai/five-video-classification-methods-implemented-in-keras-and-tensorflow-99cad29cc0b5
9. https://github.com/keras-team/keras/issues/401
10. http://cs231n.github.io/neural-networks-2/#datapre
11. https://stackoverflow.com/questions/31421413/how-to-compute-precision-recall-accuracy-and-f1-score-for-the-multiclass-case