Human Action Recognition in Videos

Dr. Manimala S[1], Kartik Nagaraj Nayak[2], Rishitha S Ramesh[3], Shriya Mittapalli[4], Vanshika V[5]

[1] *Faculty, Dept. of Computer Science Engineering, JSS Science & Technology University, Mysore, India - 570006*
[2,3,4,5] *BE Student, Dept. of Computer Science Engineering, JSS Science & Technology University, Mysore, India - 570006*

---***---

Abstract – *The technique of discovering and understanding human activities is known as human action recognition. The major goal of this procedure is to recognize people in films that may be utilized for a variety of applications. It can be used for security, surveillance, and other operations. Our video classifier is based on the UCF101 dataset. The dataset contains films of various acts, such as playing the guitar, punching, bicycling, and so on. This dataset is widely used in the construction of action recognizers, a kind of video classification application. We introduce a class of end-to-end trainable recurrent convolutional architectures that are ideal for optical comprehension tasks and illustrate how helpful these models are for action detection. In contrast to past models that assumed a fixed visual representation or conducted simple temporal averaging for sequential processing, recurrent convolutional models develop compositional representations in space and time. To understand temporal dynamics and convolutional perceptual representations, our recurrent sequence models may be trained together. We'll evaluate two models: LRCN and ConvLSTM, to see which one performs the best.*

Key Words*: *CNN, LSTM, LRCN, ConvLSTM.

1. INTRODUCTION

The challenge of anticipating what someone is doing is known as Human Action Recognition (HAR). Activity detection, or recognizing human activity, is a critical element in computer vision. The effort of following and comprehending what is happening in the film is really difficult. Develop an automated method to detect an activity in a video where a human is executing it. The goal is to compare the outcomes of a ConvLSTM and LRCN-based artificial intelligence-based human action recognition system in videos.

Objectives of HAR in Videos:

A. Learn visual characteristics from video frames.

B. Detect specific motions/ Recognize human actions.

C. Recognize people's activity in video sequences.

D. Boost the understanding of the ongoing event.

E. Compare the performance of the two models.

HAR's applications include predicting users' energy consumption and promoting health and fitness, detecting a fall and the user's subsequent movements, automatic customization of the mobile device's behavior based on a user's activity, augmented localization based on context detected using activity information, personal Biometric Signature, and so on.

2. LITERATURE SURVEY

A key driving factor in video recognition research has been the development of image recognition algorithms, which have been repeatedly changed and enlarged to cope with video data. For example, [1]'s technique involves recognising sparse spatiotemporal interest locations, which are subsequently represented using local spatiotemporal features: Histogram of Oriented Gradients [3] and Histogram of Optical Flow. The traits are then stored in a Bag of Features representation, which is pooled over many spatiotemporal grids and combined with an SVM classifier. Dense sampling of local characteristics beats sparse interest locations, according to a subsequent study [2].

State-of-the-art shallow video representations [5] employ dense point trajectories instead of calculating local video characteristics over spatio-temporal cuboids. The method, which was initially described in [7], entails changing local descriptor support areas to follow dense trajectories estimated using optical flow. The Motion Boundary Histogram (MBH) [4], a gradient-based feature computed independently on the horizontal and vertical components of optical flow, has the highest performance in the trajectory-based pipeline. Two recent breakthroughs in trajectory-based hand-crafted representations (in [5]) are the compensation of global (camera) motion [5, 9] and the application of the Fisher vector encoding [8].

Attempts to design a deep learning model for video recognition have also been numerous. Because the majority of these experiments employ a stack of consecutive video frames as input, the model must learn spatio-temporal motion-dependent properties implicitly in the initial layers, which might be problematic. [6] presented an HMAX architecture with predetermined spatiotemporal filters in the first layer for video recognition. It was then merged [9] with an HMAX model to create spatial and temporal identification channels.

The temporal stream ConvNet works with multiple-frame dense optical flow, which is commonly computed by solving for a displacement field in an energy minimization framework. A typical approach [1] for expressing energy based on intensity and gradient constancy assumptions, as well as smoothness of the displacement field. There have been efforts in [10] to use sensor data from smartphones to detect human activity. These uses the spatial movements of a person to detect his actions. Deep neural networks are also being used.

Many attempts have been done over a long period of time to identify human action, and the field continues to evolve as computer vision improves. Soon we will have a technology that accurately predicts human behaviours and actions.

3. EXISTING METHODS

Method	Title of the Paper	Year	Limitations
2D CNN	"Human Activity Recognition using Accelerometer and Gyroscope Data from Smartphones"	2020	Limited activity recognition, doesn't consider the impact of carrying cell phones in different locations.
Single Frame CNN	"Large-scale Video Classification with Convolutional Neural Networks"	2014	Fails to use a pre-trained model to extract features and use it in other deep learning models, ignores temporal relations of frame sequences.
CNN + RNN	"Video Classification using Machine Learning"	2020	Transfer learning of a pre-trained Deep Learning model to our system loses association with spatial and temporal data.

TABLE 1 – EXISTING METHODS

4. PROPOSED METHODS

We suggest two architectures:

1. Long-term Recurrent Convolutional Networks (LRCNs), a type of optical recognition and characterization architecture that manages to combine convolutional layers (CNNs) and long short term memory (LSTMs), a more advanced form of recurrent neural networks. It has CNN layers whose output is fed into LSTM units.

2. ConvLSTMs, which integrates time series with computer vision employing convolution recurrent cells in an LSTM layer. These includes convolution operations in matrix operations of LSTM cells. Both of these models can analyze variable-length input sequences as well as variable-length output sequences, and they try to predict diverse video characteristics. We are trying to compare the two models in order to identify which one performs best. We compare these models by testing them on random YouTube Videos.

4.1 SYSTEM DESIGN AND FLOW DIAGRAM

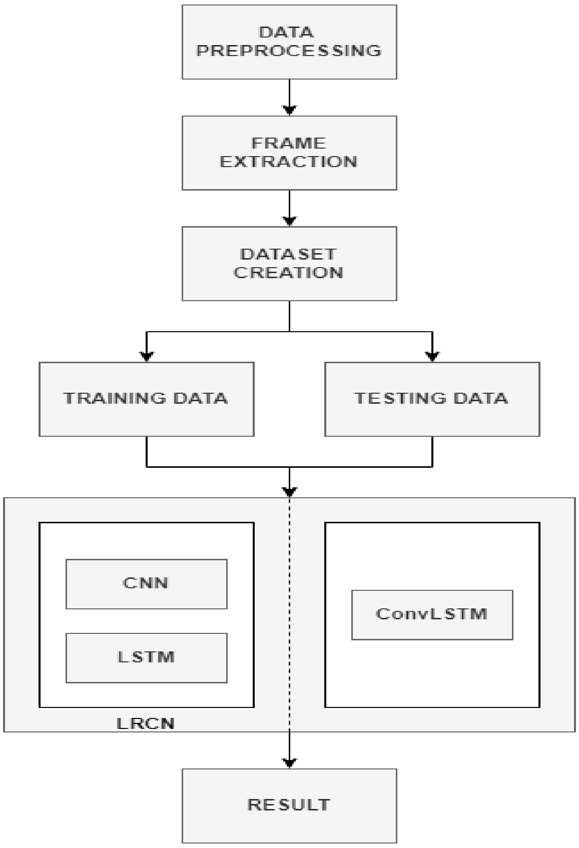


Fig. 1 - Flow Chart

A. Data Pre-processing - The information was gathered from the UCF 101 Dataset. The accuracy of the intended system is directly influenced by the quality and amount of data collected. We understand and clean data in this step.

B. Frame Extraction - In this step, for each of the videos, we extract equally spaced frames, resize the frames, and normalize the resized frames of the video.

C. Dataset Creation – We take frames from each video and extract features and labels to create dataset, i.e. indices of the class associated with videos and paths of the videos in the disk. We use the above data to fabricate our dataset.

D. Split Data (Train and Test) - The dataset is divided into two parts: training and testing.

E. Training Model - The training dataset is input into CNN and LSTM sequentially to generate an LRCN model in the first scenario, as shown in Fig. 1. In the second scenario, as shown in Fig. 1, the Convolutional-Long Short Term Memory model, which combines convolutional operations in matrix operations of LSTM cells, is given the training dataset.

F. Result - We train both ConvLSTM and LRCN models on the created dataset and compare the performance of both the models by classifying random YouTube videos.

4.2 METHODOLOGY

4.2.1 LRCN MODEL

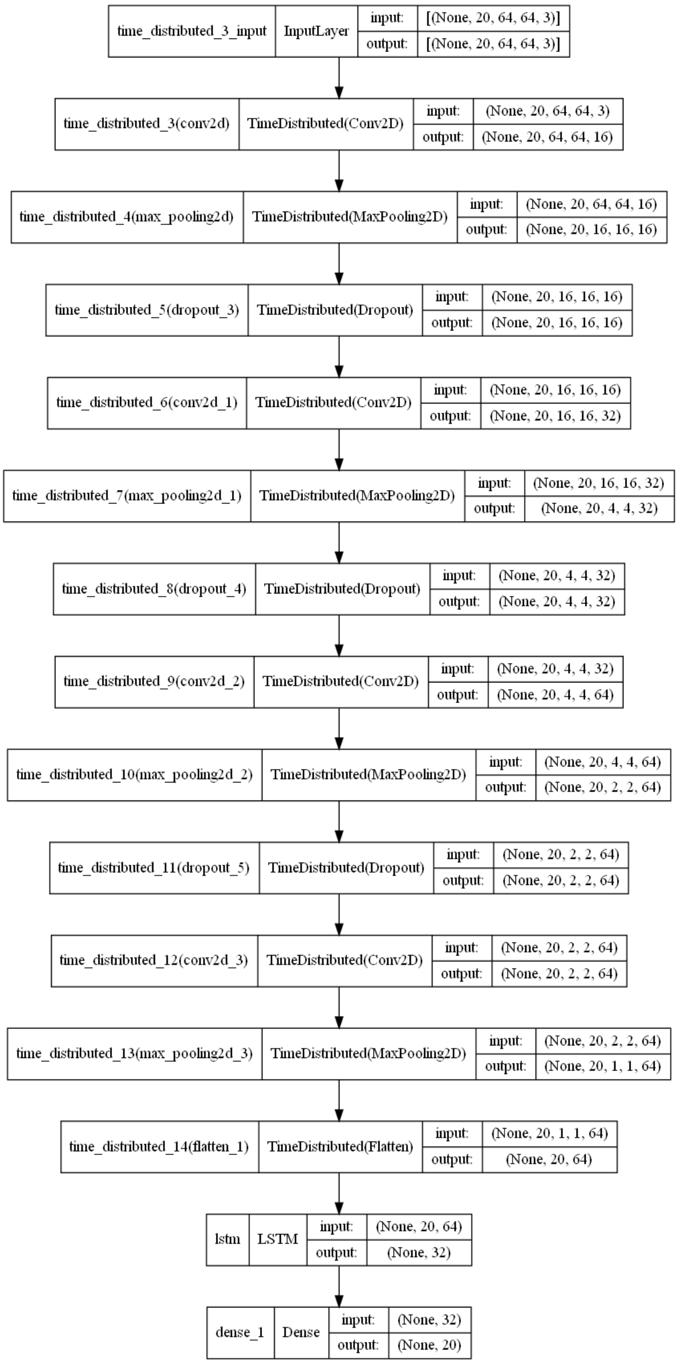

time_distributed_3_input | InputLayer | input: [(None, 20, 64, 64, 3)] | output: [(None, 20, 64, 64, 3)]
time_distributed_3(conv2d) | TimeDistributed(Conv2D) | input: (None, 20, 64, 64, 3) | output: (None, 20, 64, 64, 16)
time_distributed_4(max_pooling2d) | TimeDistributed(MaxPooling2D) | input: (None, 20, 64, 64, 16) | output: (None, 20, 16, 16, 16)
time_distributed_5(dropout_3) | TimeDistributed(Dropout) | input: (None, 20, 16, 16, 16) | output: (None, 20, 16, 16, 16)
time_distributed_6(conv2d_1) | TimeDistributed(Conv2D) | input: (None, 20, 16, 16, 16) | output: (None, 20, 16, 16, 32)
time_distributed_7(max_pooling2d_1) | TimeDistributed(MaxPooling2D) | input: (None, 20, 16, 16, 32) | output: (None, 20, 4, 4, 32)
time_distributed_8(dropout_4) | TimeDistributed(Dropout) | input: (None, 20, 4, 4, 32) | output: (None, 20, 4, 4, 32)
time_distributed_9(conv2d_2) | TimeDistributed(Conv2D) | input: (None, 20, 4, 4, 32) | output: (None, 20, 4, 4, 64)
time_distributed_10(max_pooling2d_2) | TimeDistributed(MaxPooling2D) | input: (None, 20, 4, 4, 64) | output: (None, 20, 2, 2, 64)
time_distributed_11(dropout_5) | TimeDistributed(Dropout) | input: (None, 20, 2, 2, 64) | output: (None, 20, 2, 2, 64)
time_distributed_12(conv2d_3) | TimeDistributed(Conv2D) | input: (None, 20, 2, 2, 64) | output: (None, 20, 2, 2, 64)
time_distributed_13(max_pooling2d_3) | TimeDistributed(MaxPooling2D) | input: (None, 20, 2, 2, 64) | output: (None, 20, 1, 1, 64)
time_distributed_14(flatten_1) | TimeDistributed(Flatten) | input: (None, 20, 1, 1, 64) | output: (None, 20, 64)
lstm | LSTM | input: (None, 20, 64) | output: (None, 32)
dense_1 | Dense | input: (None, 32) | output: (None, 20)

Fig. 2 – LRCN Model Structure

Our architecture is defined by the Keras Sequential model. Convolution layers will be used in LRCN, and they will be coupled to an LSTM. We use time distributed layers so that we can apply the same Conv2D layers to each of the frames in a sample. Because time distributions apply the same instance of Conv2D to each frame, the same set of weights are used at each frame.

There are four Conv2D layers, each with 16, 32, 64, and 64 filters. Each of these layers use max pooling to reduce computational complexity and dropout to avoid overfitting. The output from the convolution layer is flattened and supplied into the LSTM layer. Finally, we categorize the videos using a dense layer with softmax activation. The structure of model is Fig. 2.

4.2.2 ConvLSTM MODEL

To define our architecture, we use the Keras Sequential model. ConvLSTM is similar to LSTM, except convolutional operations substitute internal matrix multiplications. As a result, rather than being reduced to a 1Dimension array, data going through ConvLSTM keeps its input dimensions. Figure 3 depicts the model's structure.

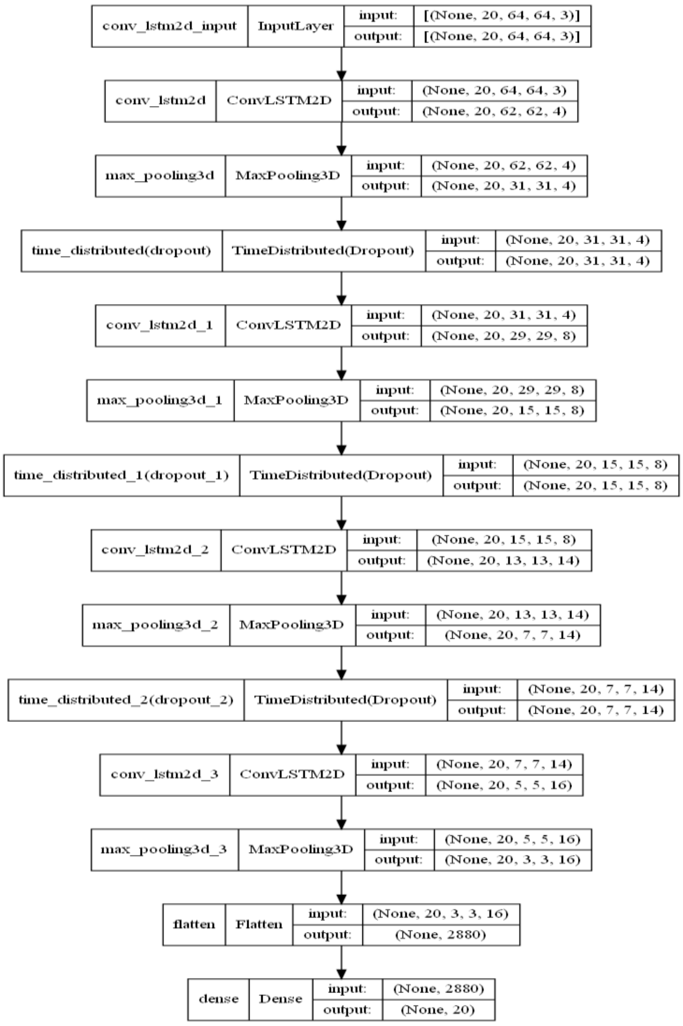
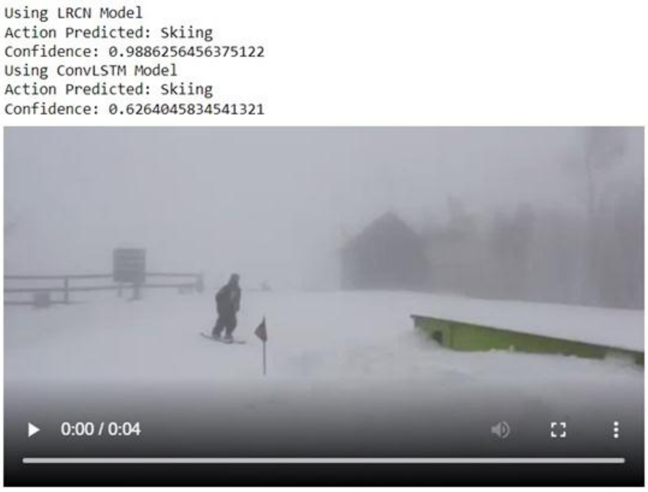

conv_lstm2d_input | InputLayer | input: [(None, 20, 64, 64, 3)] | output: [(None, 20, 64, 64, 3)]
conv_lstm2d | ConvLSTM2D | input: (None, 20, 64, 64, 3) | output: (None, 20, 62, 62, 4)
max_pooling3d | MaxPooling3D | input: (None, 20, 62, 62, 4) | output: (None, 20, 31, 31, 4)
time_distributed(dropout) | TimeDistributed(Dropout) | input: (None, 20, 31, 31, 4) | output: (None, 20, 31, 31, 4)
conv_lstm2d_1 | ConvLSTM2D | input: (None, 20, 31, 31, 4) | output: (None, 20, 29, 29, 8)
max_pooling3d_1 | MaxPooling3D | input: (None, 20, 29, 29, 8) | output: (None, 20, 15, 15, 8)
time_distributed_1(dropout_1) | TimeDistributed(Dropout) | input: (None, 20, 15, 15, 8) | output: (None, 20, 15, 15, 8)
conv_lstm2d_2 | ConvLSTM2D | input: (None, 20, 15, 15, 8) | output: (None, 20, 13, 13, 14)
max_pooling3d_2 | MaxPooling3D | input: (None, 20, 13, 13, 14) | output: (None, 20, 7, 7, 14)
time_distributed_2(dropout_2) | TimeDistributed(Dropout) | input: (None, 20, 7, 7, 14) | output: (None, 20, 7, 7, 14)
conv_lstm2d_3 | ConvLSTM2D | input: (None, 20, 7, 7, 14) | output: (None, 20, 5, 5, 16)
max_pooling3d_3 | MaxPooling3D | input: (None, 20, 5, 5, 16) | output: (None, 20, 3, 3, 16)
flatten | Flatten | input: (None, 20, 3, 3, 16) | output: (None, 2880)
dense | Dense | input: (None, 2880) | output: (None, 20)

Fig. 3 – ConvLSTM Structure Model



```
Using LRCN Model
Action Predicted: Skiing
Confidence: 0.9886256456375122
Using ConvLSTM Model
Action Predicted: Skiing
Confidence: 0.6264045834541321
```

We have four ConvLSTM2D layers with 4, 8, 14 and 16 filters respectively. Each of these layers use max pooling to reduce computational complexity and dropout to avoid overfitting. Flattening the video and feeding it through a dense network with softmax activation classifies it into one of 20 groups.

5. RESULTS

To compare our model, we utilize the Test dataset. We also put our models to the test by watching random YouTube videos. From the video, equal-spaced frames are extracted. Each frame is scaled and normalized before being input into the prediction model. The output of both models is examined and reported in Tables 2 and 3, respectively, and the accuracies of both models are compared as shown in Fig. 4.

LRCN Model					
Sequence Length	Training		Testing		Trainable Parameters
	Loss	Accuracy	Loss	Accuracy	
20	0.0512	0.9852	0.7467	0.8307	73.588
25	0.0712	0.9809	0.7242	0.8189	73,588
30	0.0884	0.9716	0.7689	0.8228	73,588

TABLE 2 – LRCN MODEL

Table 3 – ConvLSTM Model

ConvLSTM Model					
Sequence Length	Training		Validation		Trainable Parameters
	Loss	Accuracy	Loss	Accuracy	
20	0.0985	0.9677	1.4445	0.6969	90,620
25	0.1379	0.9551	1.1463	0.6890	105,020
30	0.0938	0.9716	1.1798	0.6654	119,420

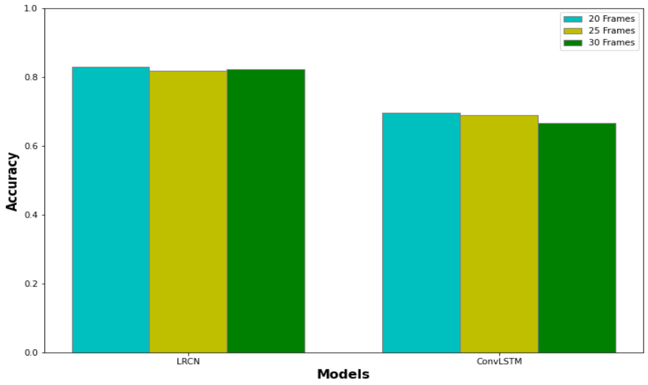


Fig. 4– Accuracy of models for different frames
Fig. 5 – Prediction using both models

When given a random Skiing (One of the training Class) video from YouTube, the LRCN model predicts the class with better confidence than the ConvLSTM model, as shown in Fig. 5. After carefully studying this data, we may conclude that the LRCN model is a better model for Human Action Recognition.

6. CONCLUSION

Action recognition in videos can be achieved in numerous ways. We compared two popular deep learning techniques in this paper: The Long-Term Recurrent Convolutional Network (LRCN) and the Convolutional LSTM (ConvLSTM), a family of models that is both spatially and temporally deep and versatile enough to be used in a variety of vision tasks with sequential inputs and outputs. After carefully analyzing the results, our finding shows that LRCN model out-performs ConvLSTM model and also is easily trainable.

ACKNOWLEDGEMENT

We express our gratitude to Dr. S.B. Kivade, Honourable Principal of JSS Science and Technology University, for providing a wonderful educational atmosphere and for his encouragement during our college years. We would like to express our deepest gratitude to our HOD, Dr. Srinath S for allowing us to pursue this project. We would like to express our gratitude to Dr. Manimala S, our project guide, for her helpful direction and passionate assistance, as well as for giving us with support and constructive recommendations for the project's improvement, without which this project would not have been feasible. We thank our lecturers' prompt assistance and gracious collaboration, as well as that of other department staff members, with whom we have worked closely throughout our project work and without whose help this project would not have been a success. Finally, we'd want to express our gratitude to our friends for their countless excellent recommendations. We also want to express our gratitude to everyone who has helped to make this learning opportunity possible at every stage of the project.

REFERENCES

[1] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. “Learning realistic human actions from movies”. In Proc. CVPR, 2008.

[2] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features ¨for action recognition. In Proc. BMVC., pages 1–11, 2009.

[3] N. Dalal and B Triggs. Histogram of Oriented Gradients for Human Detection. In Proc. CVPR, volume 2, pages 886–893, 2005.

[4] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In Proc. ECCV, pages 428–441, 2006.

[5] H. Wang and C. Schmid. Action recognition with improved trajectories. In Proc. ICCV, pages 3551– 3558, 2013.

[6] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In Proc. ICCV, pages 1–8, 2007.

[7] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In ¨ Proc. CVPR, pages 3169–3176, 2011.

[8] F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In Proc. ECCV, 2010.

[9] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In Proc. ICCV, pages 2556–2563, 2011.

[10] Human Activity Recognition using Accelerometer and Gyroscope Data from Smartphones. Khimraj, Praveen Kumar Shukla, Ankit Vijayvargiya, Rajesh Kumar.

[11] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, Li Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks.

[12] Shaunak Deshpande, Ankur Kumar, Abhishek Vastrad, Prof. Pankaj Kunekar. Video Classification using Machine Learning.

[13] Long-term Recurrent Convolutional Networks for Visual Recognition and Description Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell

BIOGRAPHIES

Dr. Manimala S,
Professor at JSS Science & Technology University,
Dept. of Computer Science Engineering

Kartik Nagaraj Nayak,
Student of JSS Science & Technology University,
Dept. of Computer Science Engineering

Rishitha S Ramesh,
Student of JSS Science & Technology University,
Dept. of Computer Science Engineering

Shriya Mittapalli,
Student of JSS Science & Technology University,
Dept. of Computer Science Engineering

Vanshika V,
Student of JSS Science & Technology University,
Dept. of Computer Science Engineering