Human Activity Recognition using CNN+LSTM

Sahithi Bakaram
sahithib@buffalo.edu
50442401

Sagar Pinnamaneni
sagarpin@buffalo.edu
50416774

Abstract

With new technologies emerging daily, the deluge of data is increasing. These recent advancements have also resulted in increased growth in fields such as artificial intelligence and the Internet of Things (IoT). Among those Human Activity Recognition (HAR) has emerged as one of the most important research topics in the fields of health and human-machine interaction in recent years as it attempts to classify various human actions in videos Many artificial intelligence-based models for activity recognition have been developed; however, these algorithms fail to extract spatial and temporal features, resulting in poor performance on real-world long-term HAR. Furthermore, studies have found that 2D convolution neural network (2D CNN) is superior for low-level spatial-temporal feature extraction while a recurrent neural network (RNN) is superior for modeling high-level temporal feature sequences. As a result, we propose a novel model in our work to address the two problems mentioned above. The proposed hybrid model for activity recognition combines Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), with CNN extracting spatial features and the LSTM network learning temporal information. This project allows us to compare the utility and accuracy of various Human Activity Recognition models. There will be a focus on two models in particular: 2-D Convolutional Neural Network and Long-Short Term Memory i.e. (CNN and CNN+LSTM). In addition, a new challenging dataset is created, which contains 101 different classes of human physical activities, and is used as the raw data for this study.

1 Introduction

In recent years, with the rapid development of information science and technology and video signal collection technology, human behavior recognition based on computer innovation has attracted more and more attention based on Behavior hiring in computer vision to find, track, analyze and monitor human activity. Human activity recognition (HAR) is the practice of using Artificial Intelligence (AI) to recognize and name human activity from activity raw data collected from a variety of sources. Decisions would be made based on their past behavioral acts. The rapid development of technologies like artificial intelligence (AI), which permits the use of these devices in a variety of application sectors, is the cause of HAR's advancement. As a result, we can conclude that HAR devices and AI methodologies or models have a mutual interaction. Earlier, these models relied on a single image or a limited number of photos, but advances in AI have opened new possibilities. In his or her day-to-day routine, a typical human may perform major activities such as walking, sitting, standing, studying, playing etc. The actions, gestures, or behaviors of a person can be translated into motion by sensors as movements. The computer's analysis of the human activity recognition code is done by translating the movement data into commands for actions.

The two key problems are the difficulty in detecting activity and the sheer volume of people analyzed. This challenge of estimating a human stance was first explored using conventional

methods. With a series of micro-activities, they could not, however, record complex movements. At this point, machine learning and data mining become more important. As a result, deep learning has become a popular approach for dealing with the problem of activity detection. Video-based human activity detection enables autonomous vehicles to sense and anticipate pedestrian behavior considerably more comprehensively, leading to more reliable driving. Even practicing your dance or exercise skills can be done using it to teach a new employee how to do a task effectively. Additionally, this technology can be applied in a wide range of situations, including those involving virtual reality, human-robot interactions, and gaming controllers.

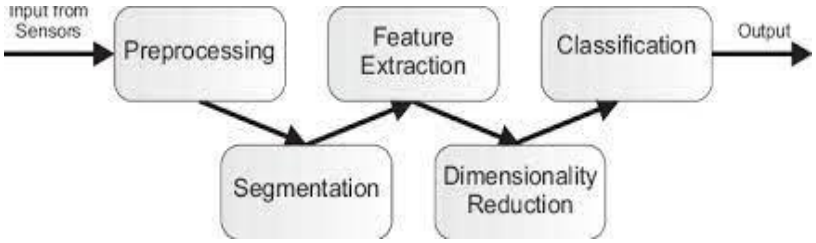


Figure 1 Basic Deep learning classification execution steps

A deep learning model can extract features and predict outcomes in an effective manner. Some methods that have been used for human activity recognition are Convolutional neural networks (CNN) which are deep spatial memory and long short-term memory (LSTM) networks which are temporally deep. Each of these methods has both advantages and disadvantages. In addition, they have been created with a specific application goal in mind. So, we discovered that we could capitalize on the strength of a single network to enhance the robustness of the other. Finally created a hybrid model with the combination of convolutional neural networks and LSTM networks to solve the problem of human activity recognition as this can extract the spatial features and perform the temporal operation. This is the best solution as both operations are performed resulting in the best prediction for the recognition of the project.

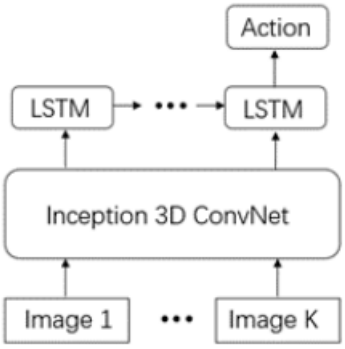


Figure 2 LRCN model-Proposed hybrid model

2 Methodology

2.1 CNN for Human Activity recognition

CNNs, or Convolutional Neural Network models, are a type of deep neural network designed for use with image data, such as handwriting recognition. When trained at scale, they have proven to be very effective on difficult computer vision problems such as identifying and localizing objects in images and automatically describing the content of images.

They are models made up of the following layers: convolutional layers and pooling layers followed by fully connected layers. Convolutional layers use a kernel that reads in small segments at a time

and steps across the entire input field to read an input, such as a 2D image or a 1D signal. Each read yields an input, which is projected onto a filter map and represents an internal interpretation of the input. Pooling layers use a signal averaging or signal maximizing process to reduce feature map projections to their most essential elements.

The convolution and pooling layers can be repeated indefinitely, resulting in multiple layers of abstraction of the input signals. These networks' output is frequently one or more fully connected layers that interpret what is read and map this internal representation to a class value.

CNNs used to recognize human activity in data learns to map a specific window of signal data to an activity in which the model reads across each window and prepares an internal representation of the window.

CNN has two advantages over other models when applied to time series classification, such as HAR: local dependency and scale invariance. Local dependency in HAR refers to the likelihood of nearby signals being correlated, whereas scale invariance refers to the scale-invariant for different paces or frequencies.

```
# Let's create a function that will construct our model
def createCNNModel():

    # We will use a Sequential model for model construction
    model = Sequential()

    # Defining The Model Architecture
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', input_shape = (image_height, image_width, 3)))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(GlobalAveragePooling2D())
    model.add(Dense(256, activation = 'relu'))
    model.add(BatchNormalization())
    model.add(Dense(len(classes_list), activation = 'softmax'))

    # Printing the models summary
    model.summary()

    # Return the constructed LRCN model.
    return model
```

Figure 3 Code to construct the CNN model

2.2 Long Short-Term Memory

RNNs, or recurrent neural networks, are a type of neural network that has been developed to know and understand from sequence data, such as sequences of observations over time or a sequence of words in a sentence.

The long short-term memory network, or LSTM for short, is perhaps the most widely used type of RNN because its proper design overcomes the general difficulties in training a stable RNN on sequence data.
When trained at scale, LSTMs have proven effective on difficult sequence prediction problems such as handwriting recognition, language modeling, and machine translation.

An LSTM model layer is made up of special units with gates that govern input, output, and recurrent connections, the weights of which are learned. Each LSTM unit has internal memory or state that accumulates as an input sequence is read and can be used as a type of local variable or memory register by the network.
LSTM is a popular first-order Markovian model in 1D sequence learning. An LSTM cell updates its states {h(t), c(t)} using the immediate previous states{ h(t -1), c(t -1)} and the current input x(t) as

$$[i_t; f_t; c_t; o_t] = \sigma\left(W x_t + K h\left(t-1\right)\right) h\left(t\right)$$

$$c(t) = c_{t-1} * f_t + i_t * c_t$$

where σ (.) denotes a sigmoid(.) applied to the input gate i(t), forget gate f(t), and output gate o(t), and a tanh() applied to the memory cell c (t) and cell state c. (t).LSTMs have two major limitations: (a) they can only model 1D sequences, not spatiotemporal data such as videos; and (b) they are difficult to capture long-term dynamics as first-order models.

2.3 Long-term Recurrent Convolutional Network (CNN+LSTM)

The approach known as the Long-term Recurrent Convolutional Network (LRCN), which combines CNN and LSTM layers in a single model. The Convolutional layers are used for spatial feature extraction from the frames, and the extracted spatial features are fed to LSTM layer(s) at each time-steps for temporal sequence modeling. This way the network learns spatiotemporal features directly in end-to-end training, resulting in a robust model.
Convolutional LSTM (ConvLSTM) overcomes limitations by extending LSTM to model spatiotemporal structures within each cell, such as states, cell memory, gates, and parameters. All the following are encoded as high-dimensional tensors:

$$I(t); F(t); \tilde{C}(t); O(t)] = \sigma(W * X(t) + K * H(t-1)).$$

The LSTM reads a sequence of input observations and develops its own internal representation of the input sequence, like how the CNN can read across an input sequence. Unlike CNN, the LSTM is trained by paying close attention to observations and prediction errors made across time steps in the input sequence, a process known as backpropagation through time.

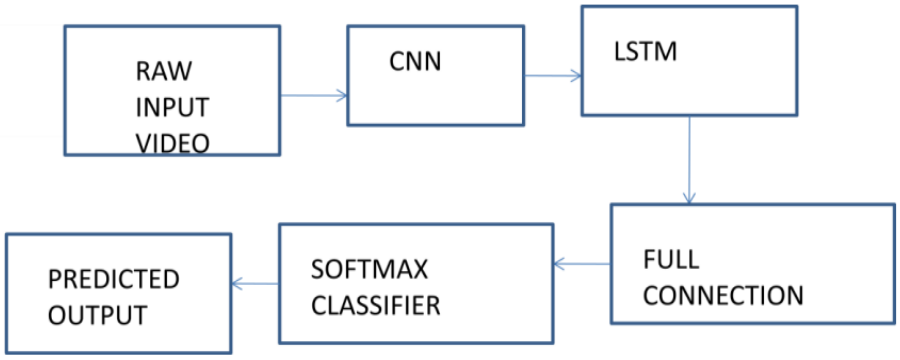


Figure 4 Flowchart of the process of CNN+LSTM model

On HAR problems, it may be more common to use an LSTM in conjunction with a CNN in a CNN-LSTM model A CNN model is used to extract features from a subsequence of raw sample data, and the output features from the CNN for each subsequence are then aggregated and interpreted by an LSTM.

```
def createLRCNModel():

    # We will use a Sequential model for model construction.
    model = Sequential()

    # Define the Model Architecture.
    ########################################################################################################################

    model.add(TimeDistributed(Conv2D(16, (3, 3), padding='same',activation = 'relu'),input_shape = (sequence_length, image_height, image_width, 3)))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))
    #model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(32))

    model.add(Dense(len(classes_list), activation = 'softmax'))

    ########################################################################################################################

    # Display the models summary.
    model.summary()

    # Return the constructed LRCN model.
    return model
```

Figure 5 Code to construct the LRCN model

3 Experiment

3.1 Dataset

UCF101 is the dataset we used to build our model. The UCF101 dataset is an expansion of the UCF 50 dataset The UCF101 dataset recordings are taken from YouTube and consist of 101 activity classes. . UCF101 is the most testing informational index to date, with 13320 recordings from 101 activity classes and the closest ness of huge variations in camera movement, object appearance and present, object scale, perspective, jumbled foundation, enlightenment conditions, and so on. Because most of the available activity recognition informational indexes are not sensible and are organized by on-screen characters, UCF101 intends to energize further examination vigorously acknowledgment by learning and investigating new sensible activity classifications. The recordings in 101 activity classes are organized into 25 gatherings, each of which can contain 4-7 recordings of an activity. Some common highlights may be shared by recordings from a similar gathering, for example, comparative foundation, comparative perspective, and so on. UCF101 includes a total of 101 activity classes divided into five categories: human-object interaction, body motion, human-human interaction, musical instruments, and sports.

3.2 Implementation

In this experiment we have used the UCF101 dataset. Implementation is started by installing the required libraries. The libraries that are different from other basic libraries which are used in this project are pafy for downloading the YouTube content and retrieve metadata, moviepy for video editing, video compositing and video processing.

After installing and importing the required libraries the dataset extraction is done by downloading the UCF101 dataset. The following figure shows the visualization of the dataset with their respective class labels. Here the list of 25 classes are selected randomly and those are used for visualization. After the visualization of dataset preprocessing of the dataset is performed.

3.2.1 Dataset Visualization

101 activity classes make up the UCF101 dataset recordings, which are taken from YouTube. The UCF50 Dataset is expanded with the UCF101 Dataset, which has 101 categories. With 13320 recordings across 101 activity classes, UCF101 offers the widest variety of activities to date. It is also the most challenging informational index to date due to its proximity to significant variations in camera movement, object scale, perspective, and illumination conditions, among other factors.
By examining the class-labeled snapshots of activities that are part of the ucf101 data set, we may acquire a better understanding of the dataset.

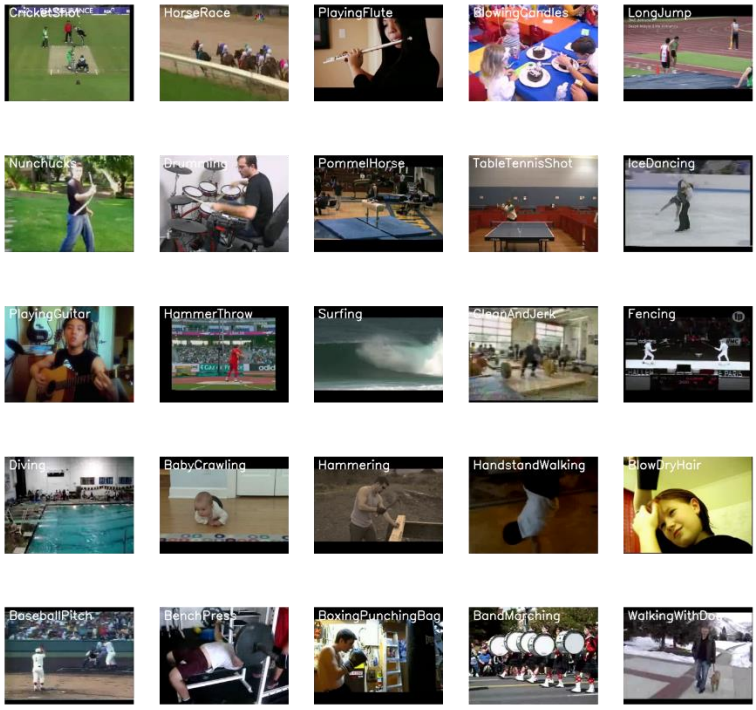


Figure 6 Visualization of video frames with class labels of 25 random classes

3.3.2 Preprocessing the Dataset

First, the video files from the dataset are read and the resizing of the video frames to fix the height and width of 64 and 64 is done respectively. This is done to reduce the computations. Normalization of the data is done by dividing the pixel values by 255 which results in the range of 0 to 1. This normalization helps in making the convergence faster when training the network.

Firstly, the image height and image width are declared with 64 and 64 which is common for CNN and CNN+LSTM but the addition of the extra variable sequence length is added in CNN+LSTM as the LSTM works on the sequential data. Here in this experiment, five classes are considered for training i.e., "Apply Lipstick", "Baby Crawling", "Military Parade", "playing guitar", "Cricket Bowling" with this initialization of constants is done.

Now we shall create a function frames Extraction () to extract the resized and normalized frames by passing the video Path as an argument where the videos are read frame by frame. Then followed by function for dataset creation where it will iterate through the classes listed and uses the above-mentioned frames Extraction () on each video file and gives the frames, class index and the video paths. Calling that function extracts the selected classes and creates the dataset. As the last step of preprocessing, we do one hot encoding to convert the class labels to one-hot encoded vectors.

3.3.4 Feature selection

Deep neural networks, inspired from natural human neural networks is very appropriate for developing computer systems. These networks include but not limited to CNN, RNN, and LSTM. In many research works, the CNN streams are fused with RGB frames and skeletal sequences at feature level and decision level. Due to CNN and LSTM's superior performance on visual and sequential data, we integrate them for feature selection and precise action detection. Through the use of parallel duplex LSTMs, the feature selection procedure is carried out in parallel in order to speed up processing. We use LSTM mainly because, Artificial neural networks and LSTM have greatly gained success in the processing of sequential multimedia data. They have obtained advanced results in speech recognition, signals processing, image processing, and text data analysis.

The "LSTM" method is appropriate for this purpose as it maintains important information about each frame of a video for a long time.

3.3.5 Splitting the dataset

Now dataset is cleaned and ready to train and test the model. So split the dataset by creating train and test dataset with the data split of 75% and 25% respectively. Also shuffling of the dataset is done to avoid the bias and to get the splitting done on the overall distribution of the data.

3.4 Implementation of models

We have built the models using the keras API with the TensorFlow as backend. Building of the model in keras is started with the sequential model.

3.4.1 Implementation of CNN model

To construct the model of CNN create CNN Model () is created .Fistly here we are using the sequential model for our model construction. Now lets define the model architecture. Here conv2D is the first layer with the filters 64 and kernel size 3x3 with the activation function relu inserting with input shape (64,64,3) then again a conv2D layer followed by Batch Normalization(),MaxPooling2D with the pool size (2,2) and the GlobalAveragePooling2D() .Next there is the Dense layer with the 256 filters followed by BatchNormalisation() on the dense layer finally giving the output in the output layer with output size and the activation function as SoftMax as it is multiclass classification.

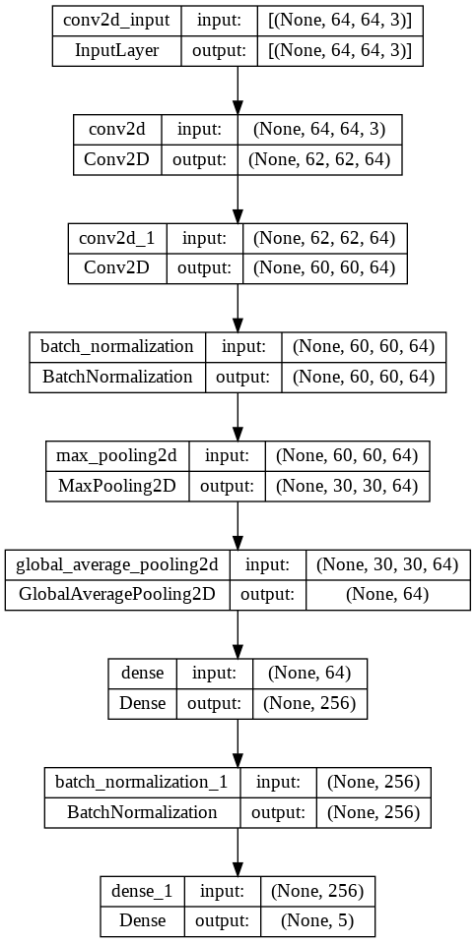


Figure 7 CNN Architecture

3.4.2 Implementing the LRCN(CNN+LSTM)

In this step, we will use the Long-term Recurrent Convolutional Network (LRCN), which combines CNN and LSTM layers into a single model. Convolutional layers are used to extract spatial features from frames, and the extracted spatial features are fed to LSTM layer(s) for temporal sequence modeling at each time-step. In this manner, the network directly learns spatiotemporal features in an end-to-end training, resulting in a robust model.

We'll also use a Time Distributed wrapper layer, which allows us to apply the same layer to each frame of the video separately. So it makes a layer (around which it is wrapped) capable of accepting input of shape (no of frames, width, height, num of channels) if the layer's original input shape was (width, height, num of channels), which is very useful because it allows you to input the entire video into the model in a single shot.
To create this LRCN model we are creating a function createLRCNModel() which gives the architecture of the model. In this architecture also the sequential model is used. The model architecture is started with time-distributed Conv2D model with the filters of 16 ,kernel size 3x3 and Relu activation function. The input size includes the sequence length with the image height and width. The conv2D is followed by MaxPooling and Dropout layers this set of layers are implemented four times followed by the Flatten layer to flatten the features extracted. Then finally these are given as input to the LSTM layer. Finally the output dense layer with the SoftMax activation function is the final layer which gives the predicted output.

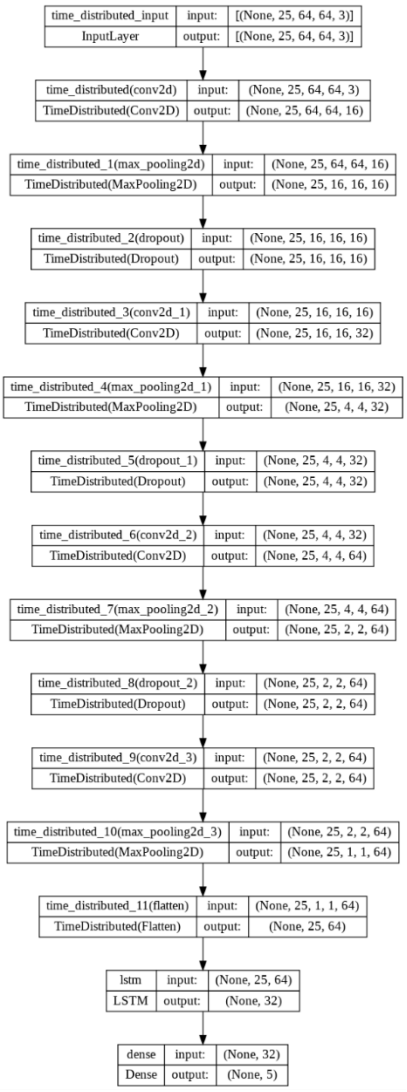


Figure 8 LRCN Architecture

3.4.3 Compiling and Training the model

Here we are considering the Adam optimizer and the categorical cross entropy for compilation of the model followed by the training of the model with the validation split of 0.25.
Evaluating the trained model :
So, after the training of the model, we evaluate the model on the test data.

3.5 Analysis of results

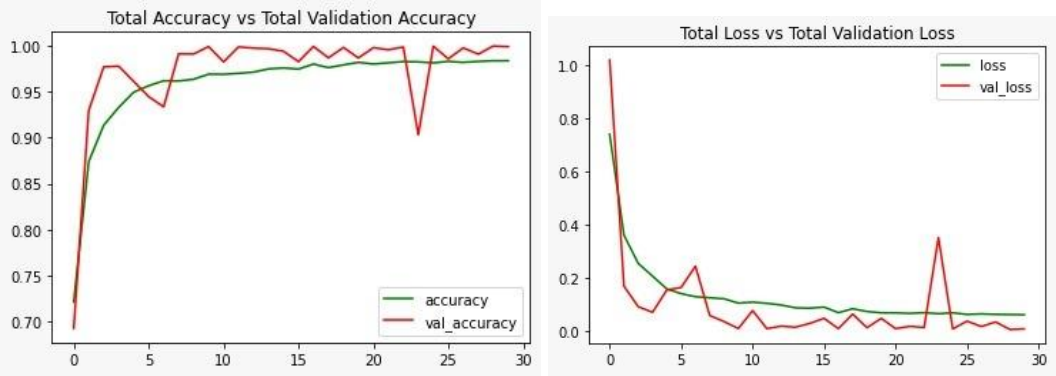
3.5.1 Accuracy and loss comparisons:

We compare the aggregated accuracy and loss for the prediction of the activity class of the train and test data set.

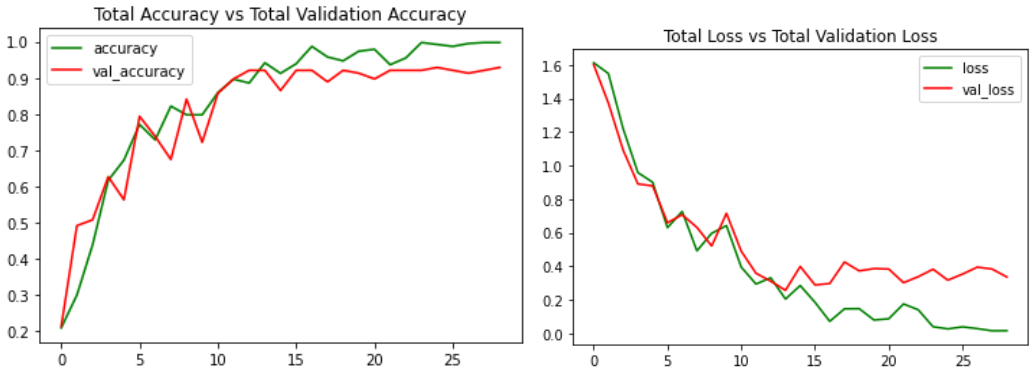
The accuracy of the prediction model obtained here in the LRCN model is 0.9524, with a a loss of 0.1685 which is a magnitude of change in the accuracy for the LRCN model whereas the accuracy of the CNN model on the UCF test data is 0.9654 that had large variation to the intial training accuracy.

If we plot the graph of accuracy for train and validation data, In CNN we can see that there is a deviation between the train accuracy and Val accuracy and the curves are not with and increased inclination starting with the less accuracy at the starting and increasing towards the end. Most of the times of its accuracy is starting from the 90 and deviating in between 90 to 98 and ending with accuracy of 98%.Whereas coming to the LRCN the training goes well with the increasing inclination starting from lower accuracy and increasing to the 95 to 97% .

Accuracy and loss comparison for CNN w.r.t Validation data set

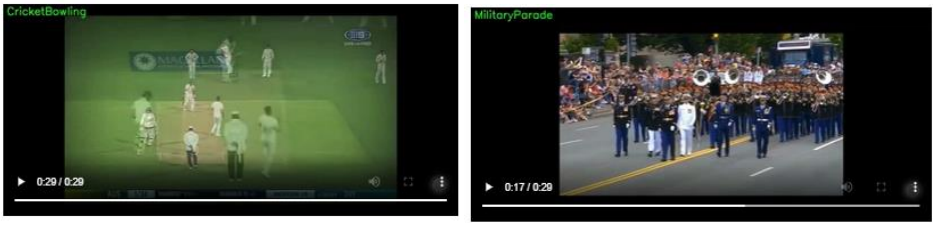


Accuracy and loss comparison for LRCN w.r.t Validation data set



3.5.2 Predictions for the YouTube videos

The training and evaluation of the model is done now the prediction of the activities from the YouTube videos. Prediction of number of activities in the videos.



Prediction of the number of activities in the videos. Videos can be given with a single activity even. So recognition of activities for the YouTube videos is more accurate for single activity in the videos.

```
1/1 [==============================] - 0s 39ms/step
Action Predicted: BabyCrawling
Confidence: 0.9088606672286987
100%|██████████| 743/744 [00:02<00:00, 288.68it/s]
```

4 Conclusion:

The frequently used UCF dataset was used for the experiment. We've presented LRCN, a class of models that is both spatially and temporally deep, and flexible enough to be applied to a variety of vision tasks involving sequential input and output. Our results consistently demonstrate that by learning sequential dynamics with a deep sequence model, we can improve upon previous methods which only learn hierarchy of parameters in the visual domain. The

suggested LRCN method beats six existing state-of-the-art studies in the field when measured against the accuracy meter, time complexity, and confusion matrix, according to the overall results. Action recognition has a wide range of uses, such as in smart homes, security systems, autonomous vehicles, etc., and has already caught the interest of many academics. It can even be used to remotely watch individuals at their jobs, in their homes, or in public places. We propose extending this research in the future to enhance the existing architecture and forecast future actions using spatiotemporal data, current action, and semantic scene segmentation and comprehension.

5 References

[1] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, two stream LSTM: A deep fusion framework for human action recognition. In Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on, 2017.

[2] M. Rezaei and R. Klette, "Look at the driver, look at the road: No distraction! no accident!" in 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 129–136, 2014.

[3] Alahi, A., Ramanathan, V., and Fei-Fei, L. (2014). "Socially-aware large-scale crowd forecasting," in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Columbus, OH), 2211–2218.

[4] Alahi, A., Ramanathan, V., and Fei-Fei, L. (2014). "Socially-aware large-scale crowd forecasting," in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Columbus, OH), 2211–2218.

[5] Aggarwal, J. K., and Cai, Q. (1999). Human motion analysis: a review. Comput. Vis. Image Understand. 73, 428–440. doi:10.1006/cviu.1998.0744

[6] Meng Li, Howard Leung, and Hubert P. H. Shum,"Human action recognition via skeletal and depth based feature fusion,"Proceedings of the 9th International Conference on Motion in Games,pp.123-132, 2016

[7] Samitha Herath, Mehrtash Harandi,and Fatih Porikli,"Going deeper into action recognition,"A survey of Image Vision Computation, 2017.

[8] Heng Wang, Alexander Klaser, Cordelia Schmid,and Cheng-Lin Liu,"Dense trajectories and motion boundary descriptors for action recognition,"International journal of computer vision,pp.60-79,2013.

[9] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in Human Behavior Understanding, 2011

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in CVPR, 2014.

CONTRIBUTION

Sahithi Bakaram (50442401)	**Sagar Pinnamaneni (50416774)**
50%	**50%**
LRCN, Code, Slide deck, Report	CNN, Code, Slide Deck, report