Human Activity Recognition using CNN and LSTM

Mini Project Report

MASTER OF TECHNOLOGY in

COMPUTER SCIENCE AND ENGINEERING

by

Nikita Patil-212CS015

Department of Computer Science

National Institute of Technology Karnataka, Surathkal.

30-05-2022

Abstract

Human activity recognition is gaining importance, not only in the view of security and surveillance but also due to psychological interests in understanding the behavioral patterns of humans. Automatic human action detection and recognition systems can aid in real-time CCTV video surveillance and reduce reliance on costly, labor-intensive manual analysis. In addition, human-machine interaction (HMI) could benefit greatly from human action recognition. Furthermore, human action recognition also has the potential to assist in behavior analysis and athletic rehabilitation. A variety of real-life mobile sensing applications are becoming available, especially in the life-logging, fitness tracking, and health monitoring domains. These applications use mobile sensors embedded in smartphones to recognize human activities in order to get a better understanding of human behavior. Convolutional Neural Networks (CNN) are great for image data and Long-Short Term Memory (LSTM) networks are great when working with sequence data. We can combine them to solve computer vision problems like video classification. This approach not only improves the predictive accuracy of human activities from raw data but also reduces the complexity of the model while eliminating the need for advanced feature engineering. The combination of CNN & LSTM network is both spatially and temporally deep. ConvLSTM and LRCN are the two approaches that can be implemented and compared for this problem.

Contents

Chapter 1

INTRODUCTION

In today ‘s world, computers have become an important aspect of life and are used in various fields. To understand human behavior and intrinsically anticipate human intentions, research into human activity recognition HAR) using sensors in wearable and handheld devices has intensified. The current HAR problem aims at using sensors; accelerometer, gyroscope, magnetometer, and others; that are built into IMU devices, and smartphones to recognize the activity being performed by the user of the device. With deep learning, it has become a lot easier to train a model to recognize certain activities from raw sensor data in a fast and efficient way. In this mini project, a detailed literature survey was done to understand different methodologies for activity recognition of humans. The ability for a system to use as few resources as possible to recognize a user’s activity from raw data is what many researchers are striving for. Machine learning models that rely on manually engineered feature datasets include the support vector machine (SVM), the histogram of gradients (HOG) feature extraction with a k-nearest neighbor classifier. Whereas in deep learning, given a raw sensor signal, a deep learning model can extract features and make predictions in a more efficient manner. ConvLSTM and LRCN models are implemented for UCF50 – Action Recognition Dataset , consisting of realistic videos taken from youtube which differentiates this data set from most of the other available action recognition datasets as they are not realistic and are staged by actors. The Dataset contains:

- 50 Action Categories
- 25 Groups of Videos per Action Category
- 133 Average Videos per Action Category
- 199 Average Number of Frames per Video
- 320 Average Frames Width per Video
- 240 Average Frames Height per Video
- 26 Average Frames Per Seconds per Video

Machine learning (ML) has been at the core of research into human activity recognition for a very long time. However, since AlexNet won the ImageNet competition in 2012, deep learning has seen successful applications in a multitude of domains like computer vision, natural language processing, speech recognition, etc. This success led several researchers to use various deep learning approaches in solving the HAR problem. In this mini project, we take a deep learning approach to HAR in a way that seeks to reduce the efforts needed for feature engineering which requires domain knowledge. The temporal nature of the HAR problem too presents a huge challenge to the ML approach and thus calling for DL. Convolutional Neural Networks (CNN) are prevalent in image recognition tasks as they improve on state-of-the-art performance in several areas. CNNs are networks that are known to be spatially deep. Long Short-Term Memory cells (LSTMs) exploit the temporal dependencies in time-series data and appear as the natural choice for modeling human movement captured with sensor data.

Chapter 2

LITERATURE SURVEY

Convolutional Neural Networks (CNN) are great for image data and Long-Short Term Memory (LSTM) networks are great when working with sequence data but when you combine both of them, you get the best of both worlds and you solve difficult computer vision problems like video classification. A Convolutional Neural Network (CNN or ConvNet) is a type of deep neural network that is specifically designed to work with image data and excels when it comes to analyzing the images and making predictions on them. It works with kernels (called filters) that go over the image and generates feature maps (that represent whether a certain feature is present at a location in the image or not) and initially it generates few feature maps and as we go deeper in the network the number of feature maps is increased and the size of maps is decreased using pooling operations without losing critical information. An LSTM network is specifically designed to work with a data sequence as it takes into consideration all of the previous inputs while generating an output. LSTMs are actually a type of neural network called Recurrent Neural Network, but RNNs are not known to be effective for dealing with the Long term dependencies in the input sequence because of a problem called the Vanishing gradient problem. LSTMs were developed to overcome the vanishing gradient and so an LSTM cell can remember context for long input sequences. The combinations of Convolution Neural Network (CNN) & Long Short Term Memory (LSTM) Network can be used to perform Action Recognition while utilizing the Spatial-temporal aspect of the videos.

We implement the first approach by using a combination of ConvLSTM cells. A ConvLSTM cell is a variant of an LSTM network that contains convolutions operations in the network. it is an LSTM with convolution embedded in the architecture, which makes it capable of identifying spatial features of the data while keeping into account the temporal relation. For video classification, this approach effectively captures the spatial relation in the individual frames and the temporal relation across the different frames. As a result of this convolution structure, the ConvLSTM is capable of taking in 3-dimensional input: (width, height, num of channels) whereas a simple LSTM only takes in 1-dimensional input hence an LSTM is incompatible for modeling Spatio-temporal data on its own.

We implement another 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 an end-to-end training, resulting in a robust model. We will also use a TimeDistributed wrapper layer, which allows applying the same layer to every frame of the video independently. So it makes a layer (around which it is wrapped) capable of taking input of shape (no of frames, width, height, number of channels) if originally the layer's input shape was (width, height, number of channels) which is very beneficial as it allows to input the whole video into the model in a single shot. We use time-distributed Conv2D layers which will be followed by MaxPooling2D and Dropout layers. The feature extracted from the Conv2D layers will be then flattened using the Flatten layer and

will be fed to a LSTM layer. The Dense layer with softmax activation will then use the output from the LSTM layer to predict the action being performed.

Chapter 3

PROPOSED SYSTEM

3.1 Problem Statement

The proposed system tries to implement human action recognition on videos using a Convolutional Neural Network combined with a Long-Short Term Memory Network. In image classification problems, an image is passed to the classifier to get the class predictions out of it. Video is just a sequence of multiple still images called frames. CNN can be used to extract spatial features at a given time step in the input sequence (video) and LSTM can be used to identify temporal relations between frames. Following steps are involved:

Step 1: Download and Visualize the Data with its Labels
Step 2: Preprocess the Dataset
Step 3: Split the Data into Train and Test Set
Step 4: Implement the ConvLSTM Approach
Step 4.1: Construct the Model
Step 4.2: Compile & Train the Model
Step 4.3: Plot Model's Loss & Accuracy Curves
Step 5: implement the LRCN Approach
Step 5.1: Construct the Model
Step 5.2: Compile & Train the Model
Step 5.3: Plot Model's Loss & Accuracy Curves

3.2 Preprocessing of dataset

After visualizing the dataset, we preprocess it. First, we will read the video files from the dataset and resize the frames of the videos to a fixed width and height, to reduce the computations and normalize the data to range [0-1] by dividing the pixel values with 255, which makes convergence faster while training the network. We consider following 4 classes for our experiment: "WalkingWithDog", "TaiChi", "Swing", "HorseRace". We extract frames from each video in above selected classes with number of frames equal to sequence length and prepare the dataset. We convert labels (class indexes) into one-hot encoded vectors. We split our data to create training and testing sets with (0.75,0.25) division. We will also shuffle the dataset before the split to avoid any bias and get splits representing the overall distribution of the data.

We implement two different combinations of CNN & LSTM i.e. ConvLSTM and LRCN.

Chapter 4

RESULTS AND SCREENSHOTS

Step 1: Visualize the Data with its Labels

In the first step, we visualize the data along with labels to get an idea about what we will be dealing with. We use the UCF50 - Action Recognition Dataset, consisting of realistic videos taken from youtube which differentiates this data set from most of the other available action recognition datasets as they are not realistic and are staged by actors.

Step 2: Preprocess the Dataset

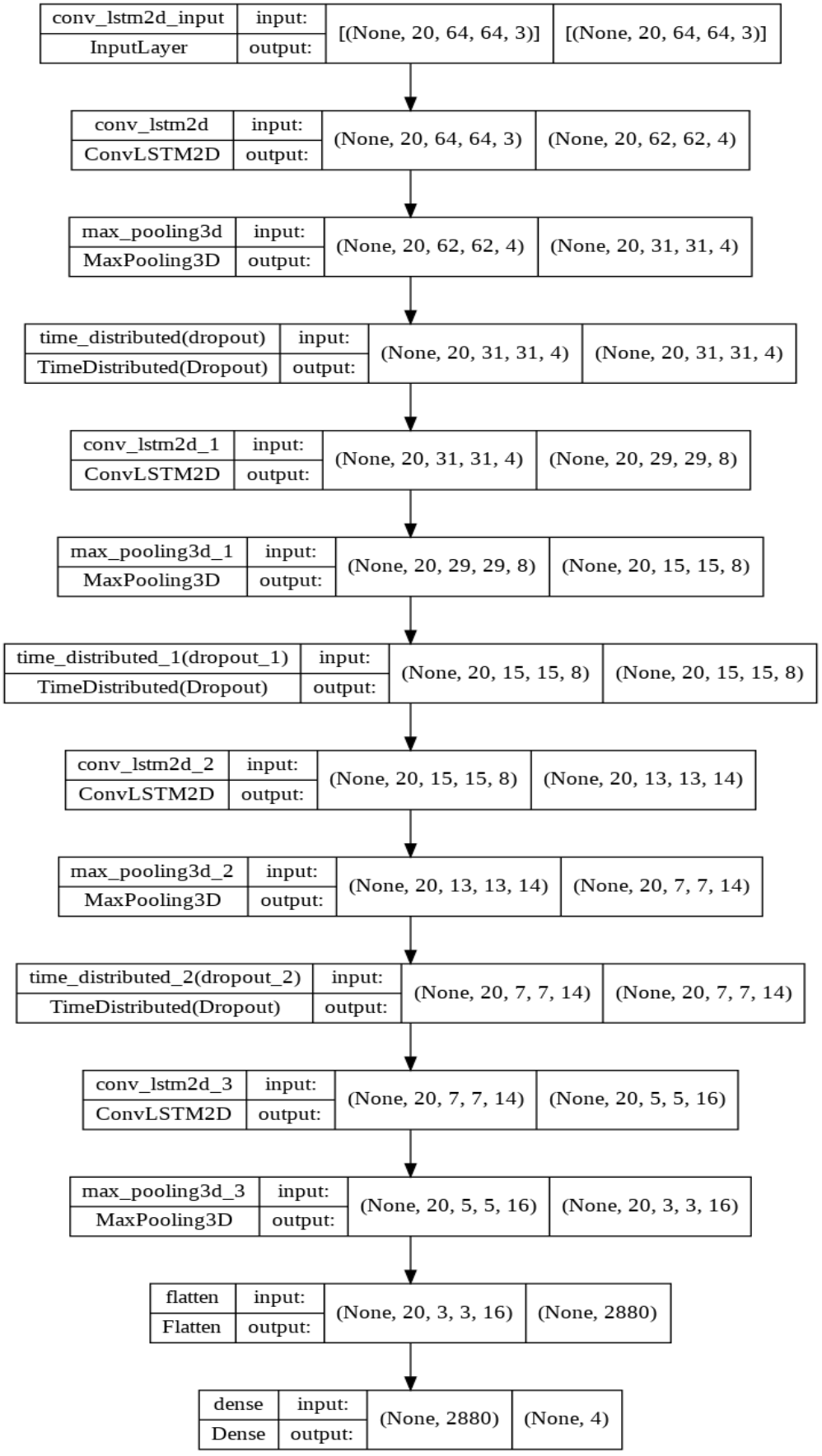
Next, we will perform some preprocessing on the dataset. First, we will read the video files from the dataset and resize the frames of the videos to a fixed width and height, to reduce the computations and normalize the data to range [0-1] by dividing the pixel values with 255, which makes convergence faster while training the network.

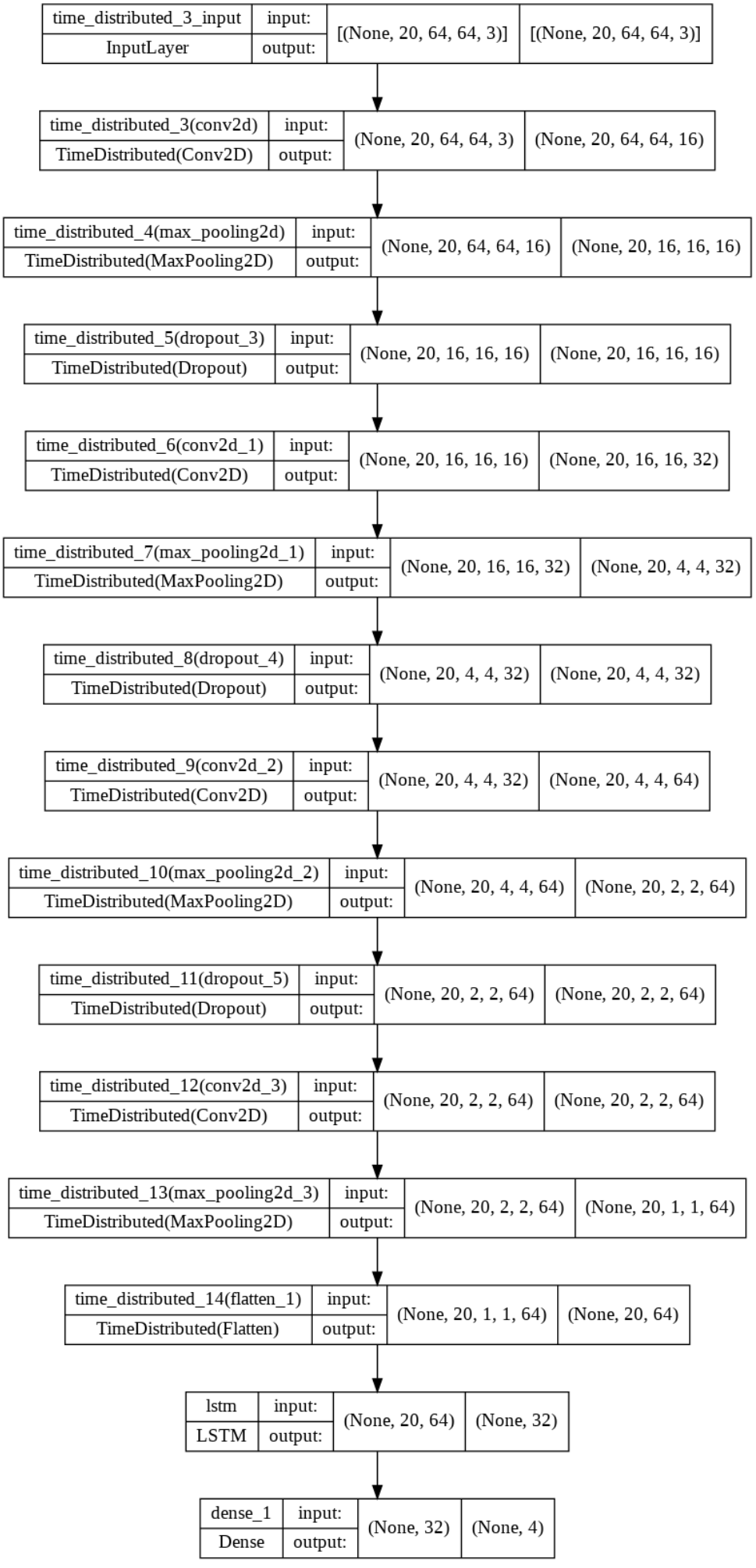
Step 3: Split the Data into Train and Test Set

As of now, we have the required features (a NumPy array containing all the extracted frames of the videos) and one_hot_encoded_labels (also a Numpy array containing all class labels in one hot encoded format). So now, we will split our data to create training and testing sets. We will also shuffle the dataset before the split to avoid any bias and get splits representing the overall distribution of the data.

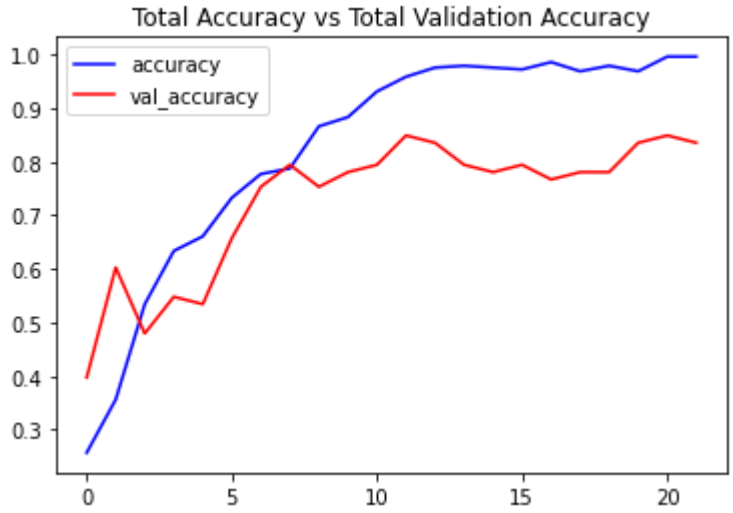
Step 4: Implement the LRCN Approach

In this step, the LRCN Approach was implemented by combining Convolution 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 an end-to-end training, resulting in a robust model.To implement our LRCN architecture, time-distributed Conv2D layers were used which were followed by MaxPooling2D and Dropout layers. The feature extracted from the Conv2D layers will be then flattened using the Flatten layer and will be fed to a LSTM layer. The Dense layer with softmax activation then used the output from the LSTM layer to predict the action being performed. Accuracy of .91.8% was found on model training and evaluation. The model was saved for future use. Model's structure:

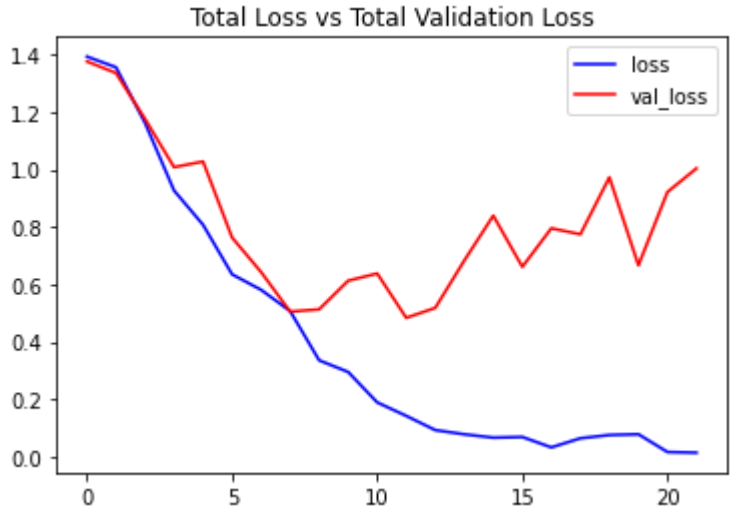

conv_lstm2d_input
InputLayer
input:
output:
[(None, 20, 64, 64, 3)]
[(None, 20, 64, 64, 3)]
conv_lstm2d
ConvLSTM2D
input:
output:
(None, 20, 64, 64, 3)
(None, 20, 62, 62, 4)
max_pooling3d
MaxPooling3D
input:
output:
(None, 20, 62, 62, 4)
(None, 20, 31, 31, 4)
time_distributed(dropout)
TimeDistributed(Dropout)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


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

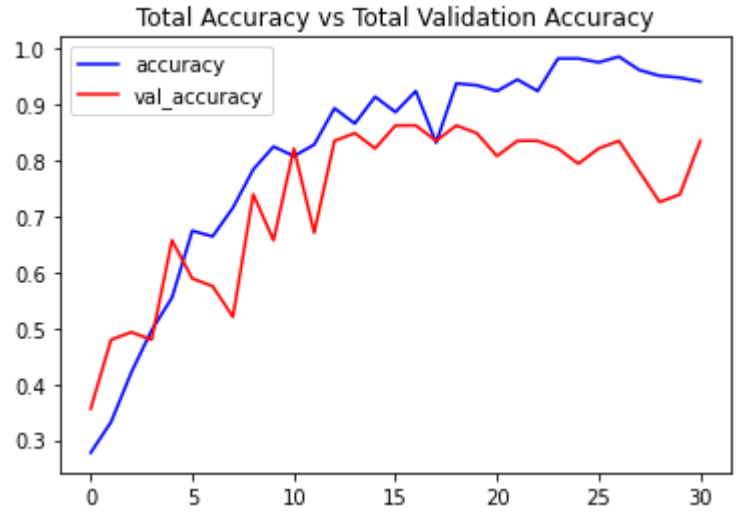
Plot for training and validation accuracy for ConvLSTM model :



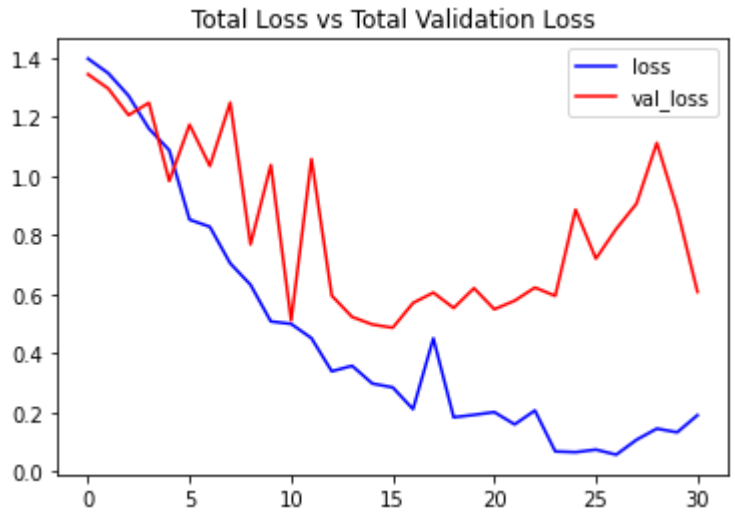
Plot for training and validation loss for ConvLSTM model :



Plot for training and validation accuracy for LRCN mode :



Plot for training and validation loss for LRCN model :



Evaluation of models on test data

- ConvLSTM: 1) accuracy: 0.7377 2) loss: 0.7642
- LRCN: 1) accuracy: 0.8197 2) loss: 0.4610

References

[1] Shi, Xingjian & Chen, Zhourong & Wang, Hao & Yeung, Dit-Yan & Wong, Wai Kin & WOO, Wang-chun. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.

[2] J. Donahue et al., ”Long-Term Recurrent Convolutional Networks for Visual Recognition and Description,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 677-691, 1 April 2017, doi: 10.1109/TPAMI.2016.2599174.

[3] C. J. Dhamsania and T. V. Ratanpara, ”A survey on Human action recognition from videos,” 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 2016, pp. 1-5, doi: 10.1109/GET.2016.7916717.

[4] M. Zeng et al., ”Convolutional Neural Networks for human activity recognition using mobile sensors,” 6th International Conference on Mobile Computing, Applications and Services, 2014, pp. 197-205, doi: 10.4108/icst.mobicase.2014.257786.