Human Activity Recognition using LSTM-CNN

Dheeraj Vurukuti
Washington State University
Pullman, WA
dheeraj.vurukuti@wsu.edu
Priyanka Ghosh Dastidar
Washington State University
Pullman, WA
p.ghoshdastidar@wsu.edu

Abstract

Recent advancements in artificial intelligence (AI) have increased human curiosity about new research topics by allowing us to recognize objects, understand the world, analyze time series, and predict future sequences. AI researchers are becoming increasingly interested in neural networks, which have useful applications in speech recognition, language modeling, video processing, and time series analysis. One of the challenging questions in this fascinating area of AI that seeks solutions is the recognition of human behavior, also known as ”human activity recognition” (HAR). A wide range of real-world applications are also covered by HAR, including those in the areas of security, gaming, personal fitness, and healthcare. The advancement of Human Computer Interaction (HCI) technology has become more popular for capturing behaviors using sensors like accelerometers, magnetometers, and gyroscopes. HAR can be accomplished with sensors, images, smartphones, or videos.

1 Introduction

In this project, we will discover how to use a convolutional neural network along with a long-short-term memory network to accomplish human action recognition on videos. TensorFlow will be used twice, with two distinct structures and methods. Then we'll use the model that performs the best to make predictions about YouTube videos. In the input sequence (video), a CNN will be used to extract spatial features at a specific time step, and an LSTM will be used to determine the temporal relationships between the frames.

With the current trend of deep learning, CNN and RNN architectures have become more prevalent, and the use of deep learning models to train time series of inertial sensor data is still being investigated by researchers. CNN and RNN deep learning models focus on a data-driven approach using sequential information to learn discriminating qualities from raw sensor data.

HAR has grown in popularity as sensor technology and ubiquitous computing technology have advanced, and it is extensively employed while maintaining privacy. To increase recognition accuracy, researchers investigated the effect of several forms of sensor technology on activity recognition. Human activity identification technologies may be broadly classified into two categories based on how sensors are used in an environment: approaches based on fixed sensors and approaches based on mobile sensors.

Fixed sensor approaches imply that information is received through sensors installed in a fixed position, such as acoustic sensors, radars, static cameras, and other ambient-based sensors. Camera-

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

based approaches are the most common, with the background removal method, optical flow method, and energy-based segmentation method being the most commonly used to extract features.

The other method of activity recognition is to use mobile sensors. In these approaches, data from various types of activities is often collected using a collection of specific body-worn motion sensors, such as accelerometers, gyroscopes, and magnetometers. Human movement would cause changes in acceleration and angular velocity data. As a result, they might be utilized to deduce human actions. Sensor miniaturization and adaptability enable people to wear or carry mobile devices equipped with numerous sensing units.

Since these sensors generate a vast dataset, it will be critical to process and analyze the entire dataset with proper automated systems. In this scenario, HAR systems will play a vital role in avoiding data analysis issues linked to the system. From the enormous amount of raw data gathered, a feature vector will be retrieved, and an activity recognition model based on the feature vector will be developed at the end of the learning algorithms. To grasp the maximum accuracy of the process of recognition, it is critical to use a well-trained, efficient model.

2 Dataset

Here, we are using the UCF50 Action Recognition Dataset, which consists of realistic videos taken from YouTube, which differentiates from most of the other available action recognition datasets as they are not realistic and staged by actors. This 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

3 Literature review

The wide range of applications for human activity recognition (HAR) in numerous fields has made it a significant research issue. It allows you to assess players' abilities in the sports sector in addition to helping sick people acquire medical care.

In recent years, deep learning models have shown great promise in achieving high accuracy and robustness in HAR tasks. Among the various deep learning architectures proposed for HAR, the combination of long short-term memory (LSTM) and convolutional neural networks (CNN) has received increasing attention due to its ability to capture both temporal and spatial information.

Despite promising results, there are still some challenges associated with LSTM-CNN architectures for HAR tasks. One major challenge is the need for large amounts of labeled data to train the model effectively, which can be time-consuming and costly. Another challenge is the interpretability of the model, as the black-box nature of deep learning models makes it difficult to understand how the model arrives at its predictions.

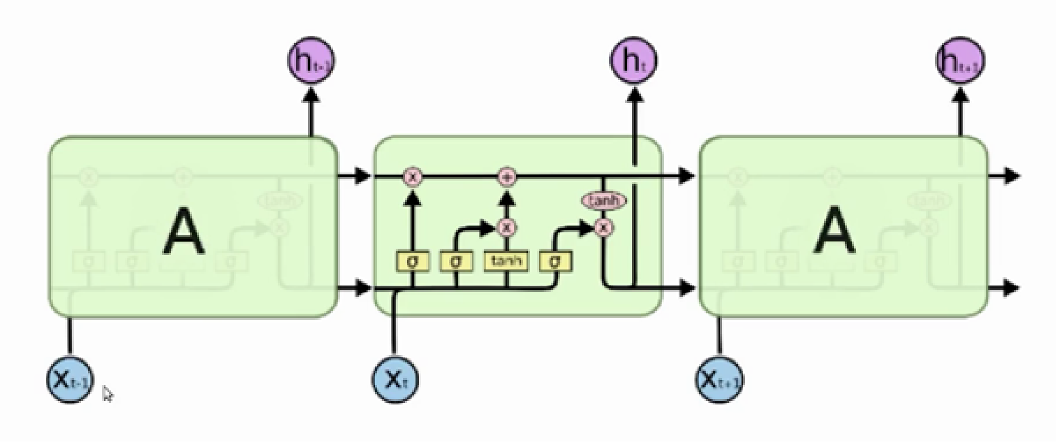
3.1 CNN Architecture

CNN architecture was influenced by the basic structure and operation of the human brain's visual cortex. A neuron in a layer will only be connected to a tiny region of the layer before it, rather than all the neurons in a fully connected layer. CNN is also referred to as ConvNet. CNN is not fully connected. The classic CNN design includes the input layer, output layer, and numerous hidden layers that might include the convolution layer, relu layer, pooling layer, and finally the fully linked layer. Backpropagation is frequently used in the final convolution to effectively converge the measurement error and accurately weight the finished product. The goal of the convolution process would be to extract high-level characteristics that seek to provide a more crucial and delicate link between the classification's input and output.

3.2 LSTM Architecture

LSTM is a kind of RNN that can learn and remember very long-term dependencies across long sequences of input data. As a result, LSTM is commonly employed for time series analytic challenges. LSTM does not employ activation functions in any of its recurrent modules. The values that were stored are not updated. LSTM does not suffer from the vanishing gradient problem during training. LSTM are often built in 'blocks' or cells with 3 or 4 gates, such as an input gate, an output gate, a forget gate, and so on. To deal with the vanishing gradient problem, LSTM employs the gating concept. The cell can recollect values throughout different time periods. The cell can recall values across different time intervals.

The advantage of utilizing LSTMs for sequence classification is that they can learn directly from raw time series data, eliminating the need for domain expertise to manually construct input features. Multiple parallel sequences of input data, such as accelerometer and gyroscope data, can be supported by the model. The model learns to extract characteristics from observation sequences and map internal features to distinct activity types.



3.3 Human Activity Recognition with CNN and LSTM

HAR has been extensively researched in a wide research area in the past decade. HAR on smart phone data and various sensor data has been rapidly evolving around many different applications that serve humankind. Mobile sensing advancements enable users to measure their sleep and exercise patterns, monitor personal commute routines, watch their emotional condition, and even track any type of human activity. Statistical learning approaches have also been utilized to address the challenge of activity recognition. Naive Bayes is one of them, as is K-Nearest Neighbor (KNN), which has been used to distinguish actions such as walking, running, and jumping. However, expert expertise was required to develop the features, and systems became more heuristic.

The approach used here is CNN LSTM, with the output of convolution set as the input to LSTM. Convolution was performed on several time frames and through a "time distributed layer" that is used for time series analysis. Some probabilistic models for behavior prediction using the LSTM network for behavior modeling have also been created. This method has also been carried out for human activity from inertial sensor time data using batch normalized deep LSTM recurrent networks. The LSTM model is a multi-stacked LSTM network for multi-class HAR classification.

4 Methodology

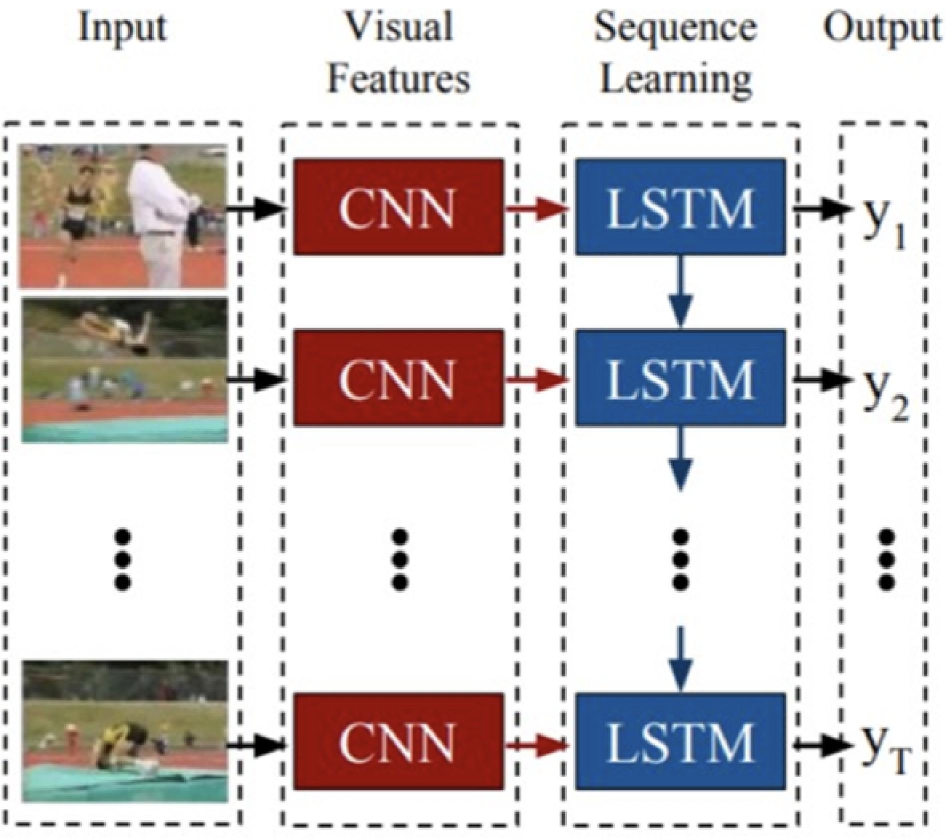
Here, we would use Convolutional Neural Networks (CNN) which are great for image data, and combine them with Long-Short Term Memory (LSTM) networks, which works great when working

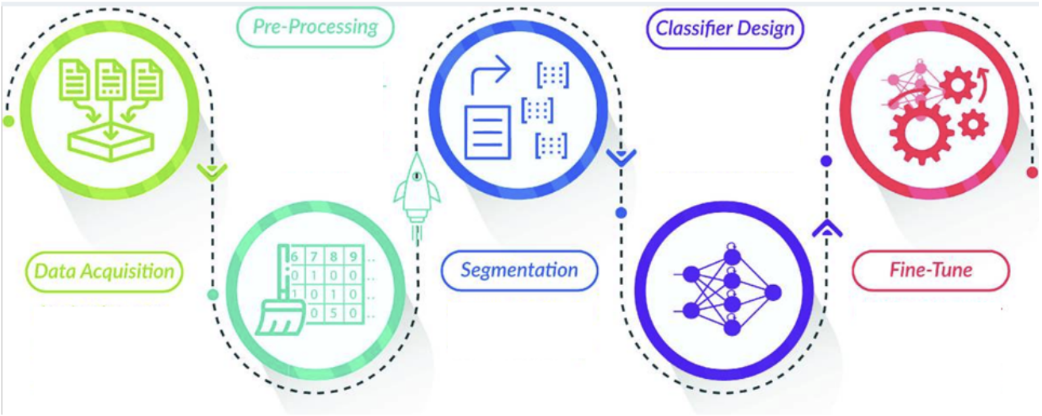
with sequence data.
Below are the steps we would perform in delivering the project:

- Step 1: Download the data and display it with its labels.
- Step 2 : Pre-process the dataset
- Step 3 : Divide the data into a train set and a test set
- Step 4: Use the ConvLSTM method to create and train the model. Along with it, draw the loss and accuracy curves for the model.
- Step 5: Implement the LRCN Method and create, compile, and train the model; and draw the loss and accuracy curves of the model.
- Step 6: Test the best-performing model on YouTube videos

5 Technical Plan

Below is the diagrammatic representation of the technical plan for the project.




Data Acquisition
Pre-Processing
Segmentation
Classifier Design
Fine-Tune

6 Results

Figure 1: Visualizing data with its labels

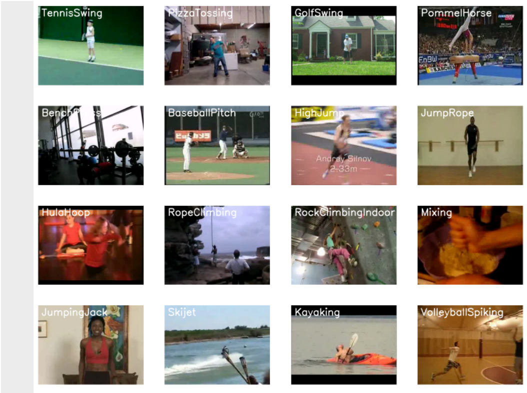
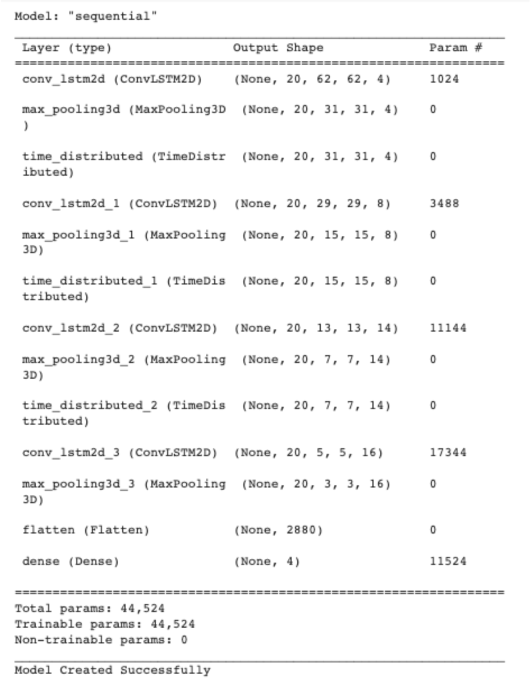


Figure 2: ConvLSTM Model



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d (ConvLSTM2D)    (None, 20, 62, 62, 4)     1024

 max_pooling3d (MaxPooling3D  (None, 20, 31, 31, 4)    0
 )

 time_distributed (TimeDistr  (None, 20, 31, 31, 4)    0
 ibuted)

 conv_lstm2d_1 (ConvLSTM2D)  (None, 20, 29, 29, 8)     3488

 max_pooling3d_1 (MaxPooling  (None, 20, 15, 15, 8)    0
 3D)

 time_distributed_1 (TimeDis  (None, 20, 15, 15, 8)    0
 tributed)

 conv_lstm2d_2 (ConvLSTM2D)  (None, 20, 13, 13, 14)    11144

 max_pooling3d_2 (MaxPooling  (None, 20, 7, 7, 14)     0
 3D)

 time_distributed_2 (TimeDis  (None, 20, 7, 7, 14)     0
 tributed)

 conv_lstm2d_3 (ConvLSTM2D)  (None, 20, 5, 5, 16)      17344

 max_pooling3d_3 (MaxPooling  (None, 20, 3, 3, 16)     0
 3D)

 flatten (Flatten)           (None, 2880)              0

 dense (Dense)               (None, 4)                 11524

=================================================================
Total params: 44,524
Trainable params: 44,524
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully
```

Figure 3: ConvLSTM Model Structure

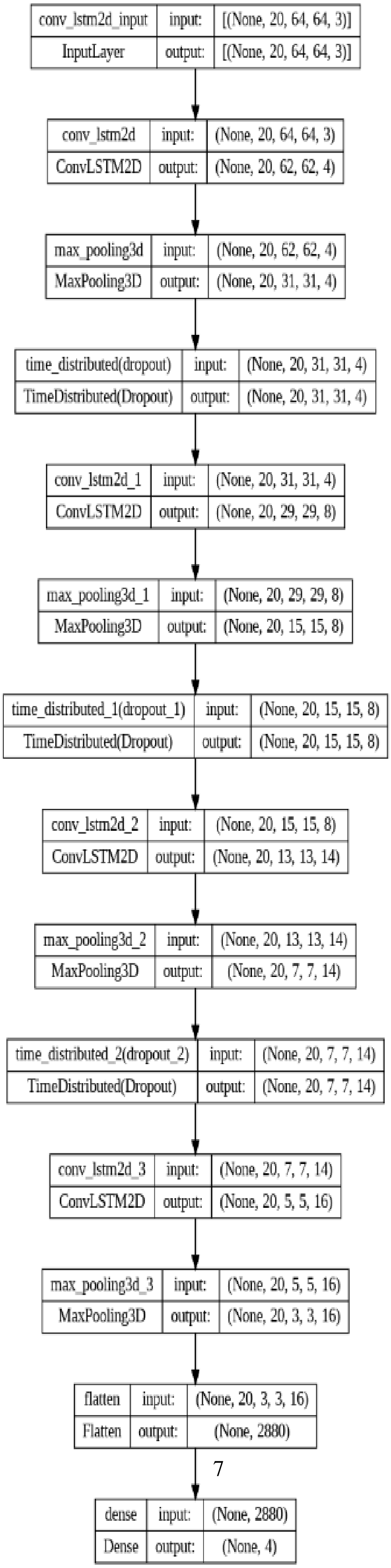

conv_lstm2d_input | input: | [(None, 20, 64, 64, 3)]
InputLayer | output: | [(None, 20, 64, 64, 3)]
conv_lstm2d | input: | (None, 20, 64, 64, 3)
ConvLSTM2D | output: | (None, 20, 62, 62, 4)
max_pooling3d | input: | (None, 20, 62, 62, 4)
MaxPooling3D | output: | (None, 20, 31, 31, 4)
time_distributed(dropout)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

Figure 4: Total Loss VS Total Validation Loss

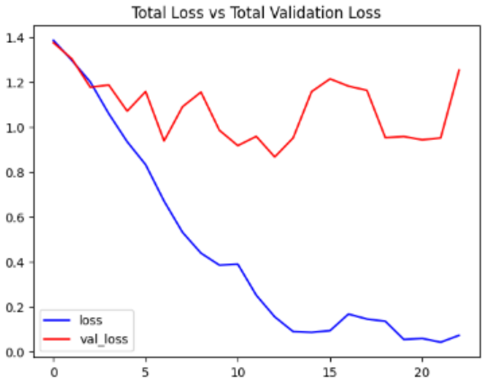


Figure 5: Total Accuracy VS Total Validation Accuracy

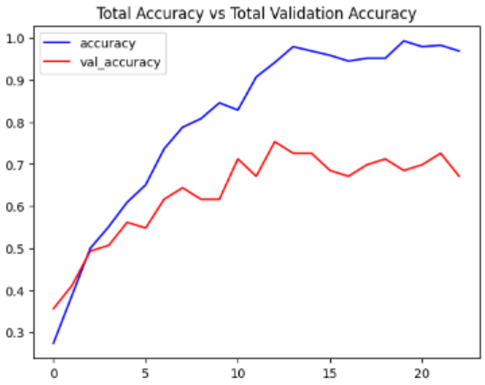
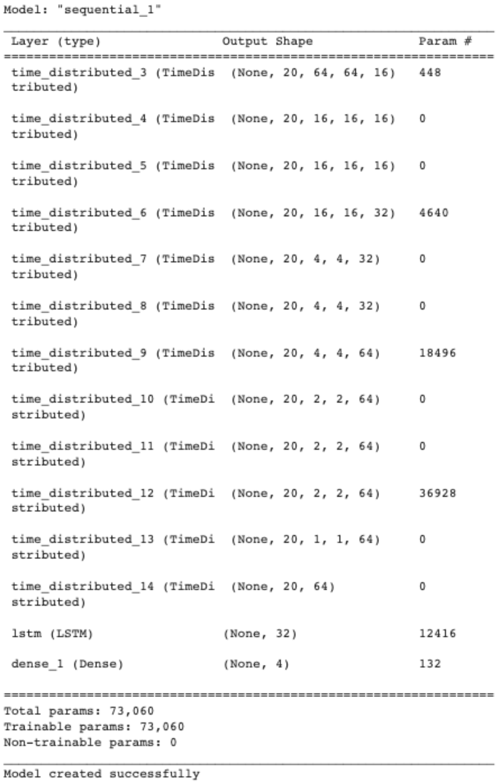


Figure 6: LRCN Model



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_3 (TimeDis  (None, 20, 64, 64, 16)   448
 tributed)

 time_distributed_4 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_5 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 16, 16, 32)   4640
 tributed)

 time_distributed_7 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_9 (TimeDis  (None, 20, 4, 4, 64)     18496
 tributed)

 time_distributed_10 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time_distributed_11 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time_distributed_12 (TimeDi  (None, 20, 2, 2, 64)     36928
 stributed)

 time_distributed_13 (TimeDi  (None, 20, 1, 1, 64)     0
 stributed)

 time_distributed_14 (TimeDi  (None, 20, 64)           0
 stributed)

 lstm (LSTM)                 (None, 32)                12416

 dense_1 (Dense)             (None, 4)                 132

=================================================================
Total params: 73,060
Trainable params: 73,060
Non-trainable params: 0
_________________________________________________________________
Model created successfully
```

Figure 7: LRCN Model Structure

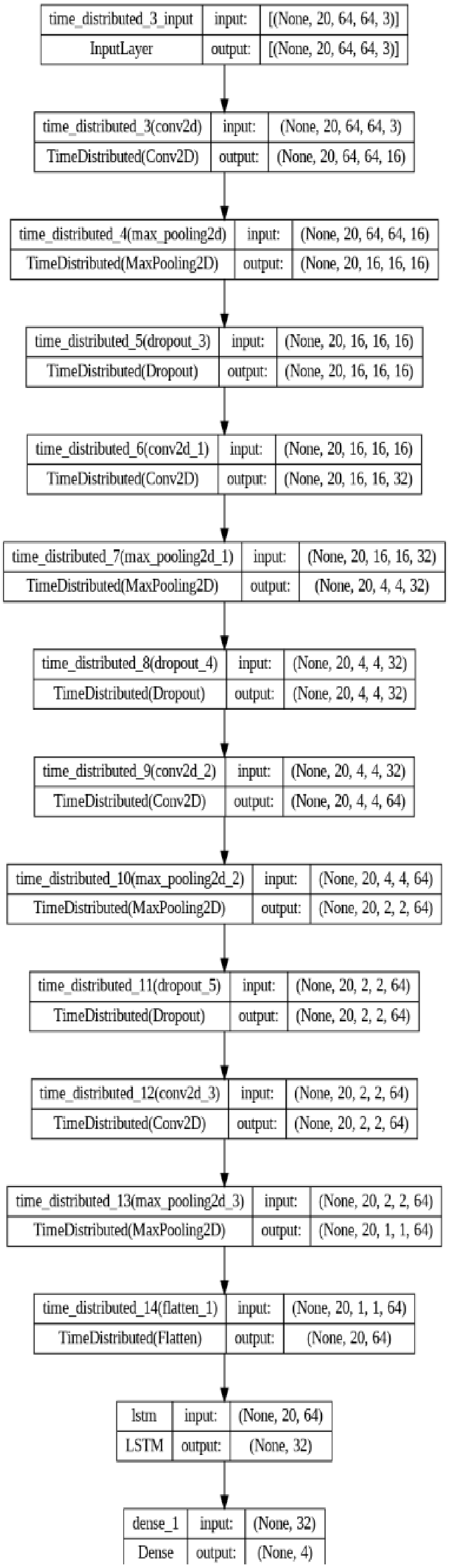

time_distributed_3_input | input: | [(None, 20, 64, 64, 3)]
InputLayer | output: | [(None, 20, 64, 64, 3)]
time_distributed_3(conv2d) | input: | (None, 20, 64, 64, 3)
TimeDistributed(Conv2D) | output: | (None, 20, 64, 64, 16)
time_distributed_4(max_pooling2d) | input: | (None, 20, 64, 64, 16)
TimeDistributed(MaxPooling2D)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

Figure 8: Total Loss VS Total Validation Loss

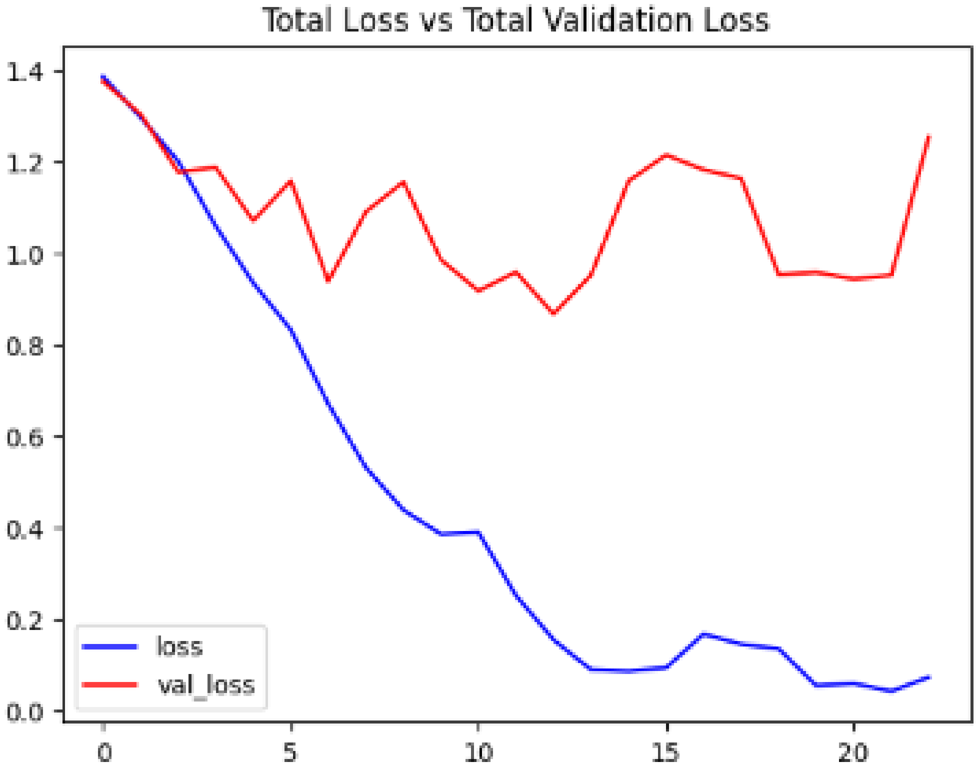


Figure 9: Total Accuracy VS Total Validation Accuracy

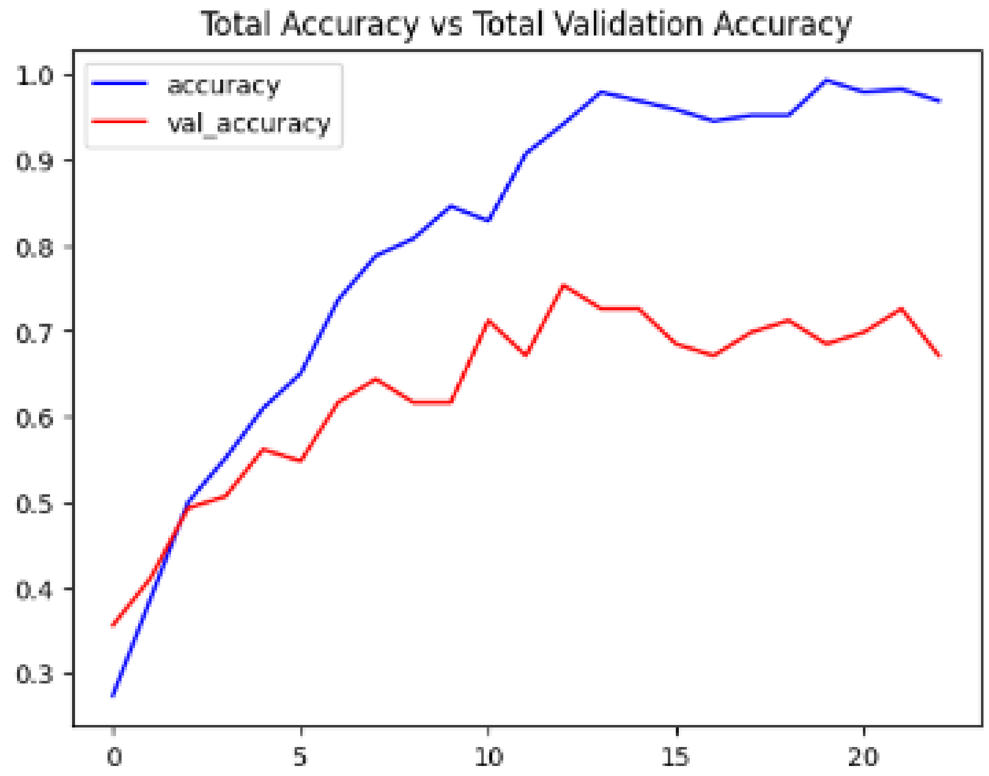
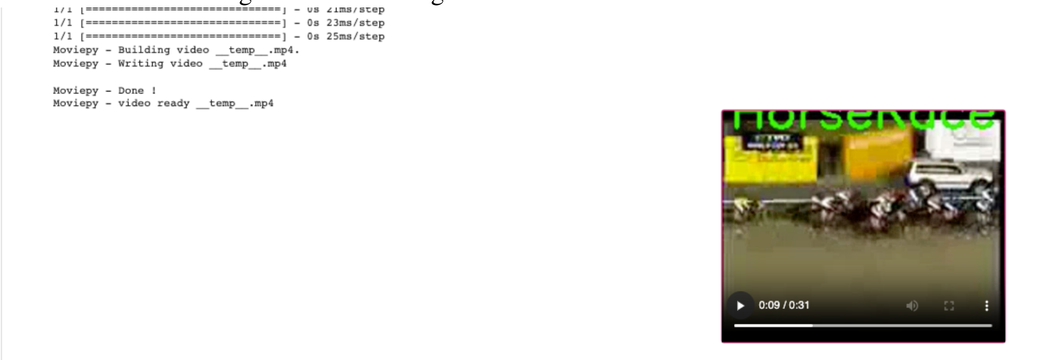


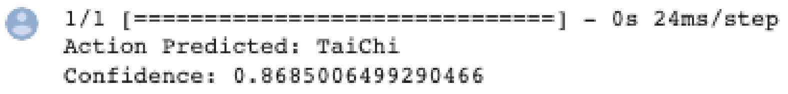
Figure 10: Predicting Action from YouTube Video Frames



```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
Moviepy - Building video __temp__.mp4.
Moviepy - Writing video __temp__.mp4

Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Figure 11: Predicting Action from Single YouTube Video Frame



```
1/1 [==============================] - 0s 24ms/step
Action Predicted: TaiChi
Confidence: 0.8685006499290466
```

7 Future Work

When employing the CNN model in a hybrid manner, merging it with another model LSTM, the performance of the CNN algorithm can be improved.

Due to its high effectiveness in extracting data features, the CNN algorithm does not require PCA, and when combined with the LSTM method, we can have a better method.

This study's shortcoming is that it only takes YouTube video frames into account. Other data sources, such cameras, wearables, and GPS sensors, can be employed for more complicated operations. It'll be taken into account in further work. The context-aware applications where complex behaviors may be identified can employ the data fusion method proposed in this research.

8 Conclusion

Compared to other machine learning models and standard neural network models, the LSTM-CNN model performs significantly better. Many academics are aiming for a system that can identify a user's activity from raw data while using the least amount of resources possible. Using just a few model parameters, the above model could automatically extract activity features and categorize them.

References

[1] L. Alpoim, A. F. da Silva and C. P. Santos, "Human Activity Recognition Systems: State of Art", 2019 IEEE 6th Portuguese Meeting on Bioengineering (ENBENG), pp. 1-4, Feb. 2019.

[2] S. Oniga and J. Suto, "Human activity recognition using neural networks", Proceedings of the 2014 15th International Carpathian Control Conference (ICCC), pp. 403-406, May 2014.

[3] T. Zebin, M. Sperrin, N. Peek and A. J. Casson, "Human activity recognition from inertial sensor time-series using batch normalized deep LSTM recurrent networks", 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1-4, Jul. 2018.

[4] L. B. Marinho, A. H. de Souza Junior and P. P. Rebouças Filho, "A New Approach to Human Activity Recognition Using Machine Learning Techniques" in Intelligent Systems Design and Applications, Cham:Springer International Publishing, vol. 557, pp. 529-538, 2017.

[5] Y. Chen, K. Zhong, J. Zhang, Q. Sun and X. Zhao, "LSTM Networks for Mobile Human Activity Recognition", presented at the 2016 International Conference on Artificial Intelligence: Technologies and Applications, 2016.

[6] C. Jobanputra, J. Bavishi and N. Doshi, "Human Activity Recognition: A Survey", Procedia Computer Science, vol. 155, pp. 698-703, 2019.

[7] A. Murad and J.-Y. Pyun, "Deep Recurrent Neural Networks for Human Activity Recognition", Sensors, vol. 17, no. 11, pp. 2556, Nov. 2017.

[8] J. R. Kwapisz, G. M. Weiss and S. A. Moore, "Activity recognition using cell phone accelerometers", SIGKDD Explor. Newsl., vol. 12, no. 2, pp. 74-82, Mar. 2011.

[9] https://www.researchgate.net/publication/340073805-LSTM-CNN-Architecture-for-Human-Activity-Recognition

[10] https://ieeexplore.ieee.org/abstract/document/9065078