POSE ESTIMATION

A project Report

Submitted in the partial fulfillment of the requirements for the award of the degree of

# Bachelor of technology in

Department of Computer Science and Engineering

By

ALLURI YASHWANTH (2010030007)

MANNE TEJASWINI SAI GAYATHRI (2010030481)

KONYALA ESHIKA SANJANA (2010030521)

KOMARAVELLI SAI KAMAL (2010030532)

under the supervision of

# DR. ARPITA GUPTA,

Associate Professor



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

March, 2022

**DECLARATION**

The Project Report entitled “Pose Estimation” is a record of the bonafide work of Alluri Yashwanth (2010030007), Manne Tejaswini Sai Gayathri (2010030481), Konyala Eshika Sanjana (2010030521), Komaravelli Sai Kamal (2010030532), submitted in partial fulfillment for the award of B.Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/universities/institutes.

ALLURI YASHWANTH

(2010030007)

MANNE TEJASWINI SAI GAYATHRI

(2010030481)

KONYALA ESHIKA SANJANA

(2010030521)

KOMARAVELLI SAI KAMAL

(2010030532)

**CERTIFICATE**

This is to certify that the Project Report entitled “Pose Estimation” is being submitted by Alluri Yashwanth (2010030007), Manne Tejaswini Sai Gayathri (2010030481), Konyala Eshika Sanjana (2010030521), Komaravelli Sai Kamal (2010030532) submitted in partial fulfillment for the award of B.Tech in CSE to the K L University, Hyderabad is a record of bonafide work carried out under our guidance and supervision. The results embodied in this report have not been copied from any other departments/universities/institutes.

## Signature of the Supervisor

Dr. Arpita Gupta

Associate Professor

## Signature of the HOD Signature of the External Examiner

**ACKNOWLEDGEMENT**

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take grateful opportunity to thank our beloved **Founder and Chairman** who has given constant encouragement during our course and motivated us to do this project. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgment & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for her exemplary guidance, monitoring and constant encouragement throughout the course of the project. We thank **Dr. Arpita Gupta** of our department who has supported throughout this project holding the position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this project a reality. We would like to extend our sincere thanks especially to our parent, our family members and friends who have supported us to make this project a grand success.

**ABSTRACT**

Position estimation is a technique used in computer vision to track a person or object's movements. Usually, this is done by locating key points for the given objects. Through the comparison of these key points, we can develop insights into diverse movements and postures. Position estimation plays a critical role in augmented reality, animation, games, and robotics. Video sequences or still images containing human activity can be challenging to identify due to many factors, including background clutter, partial occlusion, changes in scale, viewpoint, lighting, and appearance.

In this paper, we propose and compare, two neural networks based on the convolutional long short-term memory unit, namely ConvLSTM, and implement LRCN Approach by combining Convolution and LSTM layers in a single model. Another similar approach can be to use a CNN model and LSTM model trained separately. The CNN model can be used to extract spatial features from the frames in the video, and for this purpose, a pre-trained model can be used, that can be fine-tuned for the problem. And the LSTM model can then use the features extracted by CNN, to predict the action being performed in the video.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Topics** | **Page Number** |
|  | Introduction | 7 |
|  | Related Work | 8-9 |
|  | Proposed Work | 10-15 |
|  | Dataset | 16-17 |
|  | Implementation | 18-21 |
|  | Discussion | 22 |
|  | Results | 23-24 |
|  | GitHub | 25 |
|  | References | 26-27 |

1. **INTRODUCTION**

Human behavior understanding has received great interest in computer vision researchers in the last decades due to the broad variety of possible applications. Recognizing and understanding human activity is essential in applications such as automated video surveillance, health care services, human-computer interaction, autonomous driving or video analysis.[1]

Action recognition was a significant area of research in the last few years due to its enormous applications in human interaction, security, communications, and entertainment. Recognizing an action helps the system summarize an event, unlike static images, video recognition includes spatial as well as temporal features to identify the task.[2]

Researchers tend to incline towards action recognition due to its vast domain of application in surveillance, real-time gaming, video description, and human interaction.[3]

A convolutional neural network is a conventional model for image classification but for a [4][5] video each and every frame plays a critical role in the analysis. Long Short-Term Memory networks; commonly termed LSTMs are a certain form of RNN[7], competent in grasping long-term dependency. LSTM also follows a chain-like structure where each cell contains more than one neural network and is interconnected as well as interacting with each other. [6] LSTMs help in rectifying or minimizing the error that could backpropagate through time and layers. By maintaining more constant error, they allow recurrent nets to continue to learn over many time steps.

In the below describes and compares, two neural networks based on the convolutional long short-term memory unit, namely ConvLSTM, and implements LRCN Approach by combining Convolution and LSTM layers in a single model. Another similar approach can be to use a CNN model and LSTM model trained separately. The CNN model can be used to extract spatial features from the frames in the video, and for this purpose, a pre-trained model can be used, that can be fine-tuned for the problem. And the LSTM model can then use the features extracted by CNN, to predict the action being performed in the video.

5

## RELATED WORK

2.1 LITERATURE SURVEY

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.NO** | **TITLE** | **Authors** | **Date** | **Publisher** | **Techniques** | **Datasets** | **Pros** |
| 1. | Omni-sourced Webly-supervised Learning for Video Recognition | Haodong Duan 1 , Yue Zhao 1 , Yuanjun Xiong 2 , Wentao Liu 3 , and Dahua Lin | 29 Mar 2020 | The Chinese University of Hong Kong 2 Amazon AI 3 SenseTime Research | CNN, RGB | Kinetics-400, UCF101, HMDB51. | OmniSource improve Top-1 accuracy of 2D- and 3D-ConvNet baseline models |
| 2. | Learning Transferable Visual Models From Natural Language Supervision | Alec Radford Jong Wook Kim , Chris Hallacy 1 Aditya Ramesh 1 Gabriel Goh 1 Sandhini Agarwal 1 Girish Sastry 1 Amanda Askell 1 Pamela Mishkin 1 Jack Clark | 26 Feb 2021 | Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry et al. "Learning transferable visual models from natural language supervision." In *International Conference on Machine Learning*, pp. 8748-8763. PMLR, 2021. | Cross-linking immunoprecipitation (CLIP), CNN | HMBD51,  UCF101,  YFCC100M | CLIP’s features outperform the features of the best ImageNet model on a wide variety of datasets. |
| 3. | UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild | Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah. | 3 Dec 2012 | arXiv preprint arXiv: |  | Related Datasets  UCF Sports, UCF11, UCF50 and UCF101 | It consists of 101 action classes,  over 13k clips and 27 hours of video data. |
| 4. | Mask R-CNN | Kaiming He  Georgia Gkioxari Piotr Dollar  Ross Girshick | 24 Jan 2018 | Facebook AI Research (FAIR) | R-CNN which is simple and effective | COCO, COCO-WholeBody | It extends faster R-CNN by adding a branch for bounding box recognition.  It efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. |
| 5. | HPRNet: Hierarchical Point Regression for Whole-Body Human Pose Estimation | Nermin Samet, Emre Akbas | 8 june 2021 | Department of Computer Engineering METU | HPRNet as a bottom-up, one-stage method for whole-body keypoint detection. | COCO, COCO-WholeBody | The run time analysis of HPRNet shows that HPRNet runs in constant time, independently of the number of persons in an image. |

1. **PROPOSED WORK**

**3.1 MODEL & TECHNIQUES**

**3.1.1 CONVLSTM:**

**ConvLSTM** is a type of recurrent neural network for the Spatio-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. This can easily be achieved by using a [convolution](https://paperswithcode.com/method/convolution) operator in the state-to-state and input-to-state transitions

Diagram, schematic

Description automatically generated

**Fig 3.1.1**

Data collected over successive periods of time are characterized as a Time Series. In such cases, an interesting approach is to use a model based on **LSTM** (Long Short-Term Memory), a Recurrent Neural Network architecture. In this kind of architecture, the model passes the previous hidden state to the next step of the sequence. Therefore holding information on previous data the network has seen before and using it to make decisions. In other words, the data order is extremely important.

Diagram

Description automatically generated

**Fig 3.1.2 (A LSTM cell)**

When working with images, the best approach is a **CNN**(Convolutional Neural Network) architecture. The image passes through Convolutional Layers, in which several filters extract important features. After passing some convolutional layers in sequence, the output is connected to a fully-connected Dense network.

Diagram

Description automatically generated

**Fig 3.1.3(Convolution of an image with one filter)**

**3.1.2 Code:**

def create\_convlstm\_model():

    '''

    This function will construct the required convlstm model.

    Returns:

        model: It is the required constructed convlstm model.

    '''

    # We will use a Sequential model for model construction

    model = Sequential()

    # Define the Model Architecture.

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

    model.add(ConvLSTM2D(filters = 4, kernel\_size = (3, 3), activation = 'tanh',data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True, input\_shape = (SEQUENCE\_LENGTH,

                                                                                      IMAGE\_HEIGHT, IMAGE\_WIDTH, 3)))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 8, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 14, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 16, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    #model.add(TimeDistributed(Dropout(0.2)))

    model.add(Flatten())

    model.add(Dense(len(CLASSES\_LIST), activation = "softmax"))

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

    # Display the models summary.

    model.summary()

    # Return the constructed convlstm model.

    return model

## 3.2 Long-term Recurrent Convolutional Network (LRCN):

## The main idea is to use a combination of CNNs to learn visual features from video frames and LSTMs to transform a sequence of image embeddings into a class label, sentence, probabilities, or whatever you need. Thus, raw visual input is processed with a CNN, whose outputs are fed into a stack of recurrent sequence models.

## A picture containing diagram Description automatically generated

## Fig 3.2.1

The above figure is an example architecture of LRCN. As it is described in the figure, LRCN processes the variable-length visual input with a CNN. And their outputs are fed into a stack of recurrent sequence models which is LSTM in the figure. The final output from the sequence models is a variable-length prediction. This makes LRCN is a proper model to handle tasks with time-varying inputs and output, such as activity recognition, image captioning, and video description The below figure is task-specific instantiations of the LRCN model for each task.

## Diagram, engineering drawing, timeline Description automatically generated

## Fig 3.2.2

**3.2.1 Code:**

def create\_LRCN\_model():

    '''

    This function will construct the required LRCN model.

    Returns:

        model: It is the required constructed LRCN model.

    '''

    # 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

1. **DATASETS**

## 

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Characteristics | Titles | Technique |
| UCF101 – Action Recognition | UCF101 is an action recognition data set of realistic action videos, collected from YouTube, having 101 action categories.  This data set is an extension of the UCF50 data set which has 50 action categories. UCF101 gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, and illumination conditions. | 1. Omni-sourced Webly-supervised Learning for Video Recognition. 2. Learning Transferable Visual Models From Natural Language Supervision. | CNN, RGBCross-linking immunoprecipitation (CLIP), CNN |
| COCO  (Microsoft Common Objects in Context) | The MS COCO dataset is large-scale object detection, segmentation, key-point detection, and captioning dataset.  The dataset consists of 328K images.  80 object categories, the "COCO classes", which include things for which  individual instances may be easily labeled and 91 stuff categories, where "COCO stuff" includes materials and objects with no clear boundaries (sky, street, grass, etc.) that provide significant contextual information. | 3. OpenPose: Realtime Multi-Person 2D Pose  4. Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose  5. EvoPose2D: Pushing the Boundaries of 2D Human Pose Estimation using Accelerated Neuroevolution with Weight Transfer | Part Affinity Fields (PAFs), CNN, confidence  Maps  CNN, R-CNN  Accelerated Neuroevolution with Weight Transfer,CNN, neural architecture search (NAS)  , Cascaded Pyramid Network (CPN) |
| MPII (MPII Human Pose) | The MPII Human Pose Dataset for single-person pose estimation.  It consists of 25K images of which 15K are training samples, 3K are validation samples and 7K are testing samples.  The images are taken from YouTube videos covering 410 different human activities and the poses are manually annotated with up to 16 body joints. | 6. Unified Human Pose Estimation in Single Images and Videos  7. 3D Human Pose Estimation = 2D Pose Estimation + Matching  8. 2D/3D Pose Estimation and Action Recognition using Multitask Deep Learning | CNN, Fully Convolutional Networks (FCN)  CNN, NN model, MPJPE  CNN |
| PoseTrack | The PoseTrack dataset is a large-scale benchmark for multi-person pose estimation and tracking in videos.  It requires not only pose estimation in single frames, but also temporal tracking across frames.  It contains 514 videos including 66,374 frames in total, split into 300, 50 and 208 videos for training, validation and test set respectively. | 9.Recent Advances in Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective  10. PoseTrack: A Benchmark for Human Pose Estimation and Tracking.  11. Deep Learning-Based Human Pose Estimation: A Survey | (DCNNs)deep convolutional neural networks,CNet, Rnet  ML-LAB, SOPT-PT, Mask-RCNN, PAF , DeeperCut  CNNs, human-computer interaction, motion analysis, augmented reality (AR), virtual reality (VR) |

1. **IMPLEMENTATION**

**5.1 Import the Libraries**

# Import the required libraries.

import os

import cv2

import pafy

import math

import random

import numpy as np

import datetime as dt

import tensorflow as tf

from collections import deque

import matplotlib.pyplot as plt

from moviepy.editor import \*

%matplotlib inline

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.layers import \*

from tensorflow.keras.models import Sequential

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.utils import plot\_model

**5.2 Visualize the Data with its Labels**

# Create a Matplotlib figure and specify the size of the figure.

plt.figure(figsize = (20, 20))

# Get the names of all classes/categories in UCF50.

all\_classes\_names = os.listdir('/content/UCF50')

# Generate a list of 20 random values. The values will be between 0-50,

# where 50 is the total number of class in the dataset.

random\_range = random.sample(range(len(all\_classes\_names)), 20)

# Iterating through all the generated random values.

for counter, random\_index in enumerate(random\_range, 1):

    # Retrieve a Class Name using the Random Index.

    selected\_class\_Name = all\_classes\_names[random\_index]

    # Retrieve the list of all the video files present in the randomly selected Class Directory.

    video\_files\_names\_list = os.listdir(f'/content/UCF50/{selected\_class\_Name}')

    # Randomly select a video file from the list retrieved from the randomly selected Class Directory.

    selected\_video\_file\_name = random.choice(video\_files\_names\_list)

    # Initialize a VideoCapture object to read from the video File.

    video\_reader = cv2.VideoCapture(f'/content/UCF50/{selected\_class\_Name}/{selected\_video\_file\_name}')

    # Read the first frame of the video file.

    \_, bgr\_frame = video\_reader.read()

    # Release the VideoCapture object.

    video\_reader.release()

    # Convert the frame from BGR into RGB format.

    rgb\_frame = cv2.cvtColor(bgr\_frame, cv2.COLOR\_BGR2RGB)

    # Write the class name on the video frame.

    cv2.putText(rgb\_frame, selected\_class\_Name, (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2)

    # Display the frame.

    plt.subplot(5, 4, counter);plt.imshow(rgb\_frame);plt.axis('off')

**5.3 Create a Function to Extract, Resize & Normalize Frames**

def frames\_extraction(video\_path):

    '''

    This function will extract the required frames from a video after resizing and normalizing them.

    Args:

        video\_path: The path of the video in the disk, whose frames are to be extracted.

    Returns:

        frames\_list: A list containing the resized and normalized frames of the video.

    '''

    # Declare a list to store video frames.

    frames\_list = []

    # Read the Video File using the VideoCapture object.

    video\_reader = cv2.VideoCapture(video\_path)

    # Get the total number of frames in the video.

    video\_frames\_count = int(video\_reader.get(cv2.CAP\_PROP\_FRAME\_COUNT))

    # Calculate the the interval after which frames will be added to the list.

    skip\_frames\_window = max(int(video\_frames\_count/SEQUENCE\_LENGTH), 1)

    # Iterate through the Video Frames.

    for frame\_counter in range(SEQUENCE\_LENGTH):

        # Set the current frame position of the video.

        video\_reader.set(cv2.CAP\_PROP\_POS\_FRAMES, frame\_counter \* skip\_frames\_window)

        # Reading the frame from the video.

        success, frame = video\_reader.read()

        # Check if Video frame is not successfully read then break the loop

        if not success:

            break

        # Resize the Frame to fixed height and width.

        resized\_frame = cv2.resize(frame, (IMAGE\_HEIGHT, IMAGE\_WIDTH))

        # Normalize the resized frame by dividing it with 255 so that each pixel value then lies between 0 and 1

        normalized\_frame = resized\_frame / 255

        # Append the normalized frame into the frames list

        frames\_list.append(normalized\_frame)

    # Release the VideoCapture object.

    video\_reader.release()

    # Return the frames list.

    return frames\_list

**5.4 Split the Data into Train and Test Set**

# Split the Data into Train ( 75% ) and Test Set ( 25% ).

features\_train, features\_test, labels\_train, labels\_test = train\_test\_split(features, one\_hot\_encoded\_labels,

                                                                            test\_size = 0.25, shuffle = True,

                                                                            random\_state = seed\_constant)

**5.5 Implement the ConvLSTM Approach**

def create\_convlstm\_model():

    '''

    This function will construct the required convlstm model.

    Returns:

        model: It is the required constructed convlstm model.

    '''

    # We will use a Sequential model for model construction

    model = Sequential()

    # Define the Model Architecture.

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

    model.add(ConvLSTM2D(filters = 4, kernel\_size = (3, 3), activation = 'tanh',data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True, input\_shape = (SEQUENCE\_LENGTH,

                                                                                      IMAGE\_HEIGHT, IMAGE\_WIDTH, 3)))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 8, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 14, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 16, kernel\_size = (3, 3), activation = 'tanh', data\_format = "channels\_last",

                         recurrent\_dropout=0.2, return\_sequences=True))

    model.add(MaxPooling3D(pool\_size=(1, 2, 2), padding='same', data\_format='channels\_last'))

    #model.add(TimeDistributed(Dropout(0.2)))

    model.add(Flatten())

    model.add(Dense(len(CLASSES\_LIST), activation = "softmax"))

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

    # Display the models summary.

    model.summary()

    # Return the constructed convlstm model.

    return model

# Construct the required convlstm model.

convlstm\_model = create\_convlstm\_model()

# Display the success message.

print("Model Created Successfully!")

**6. DISCUSSION**

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.

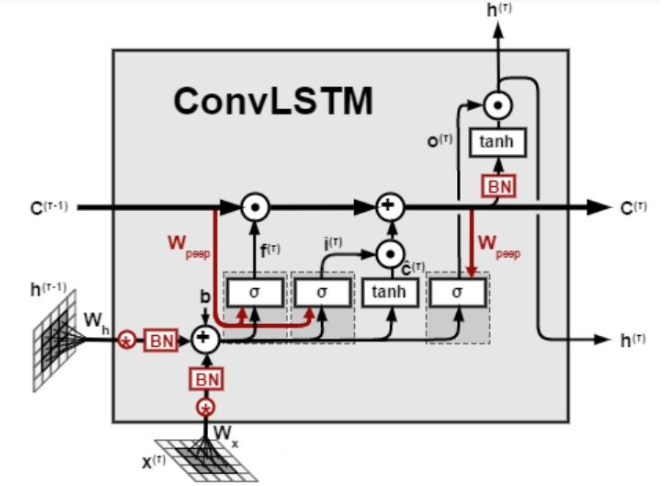


Fig: 6.1

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.

**7. RESULTS & CONCLUSION**

**Output:**



**Fig: 7.1**



**Fig: 7.2**



**Fig: 7.3**

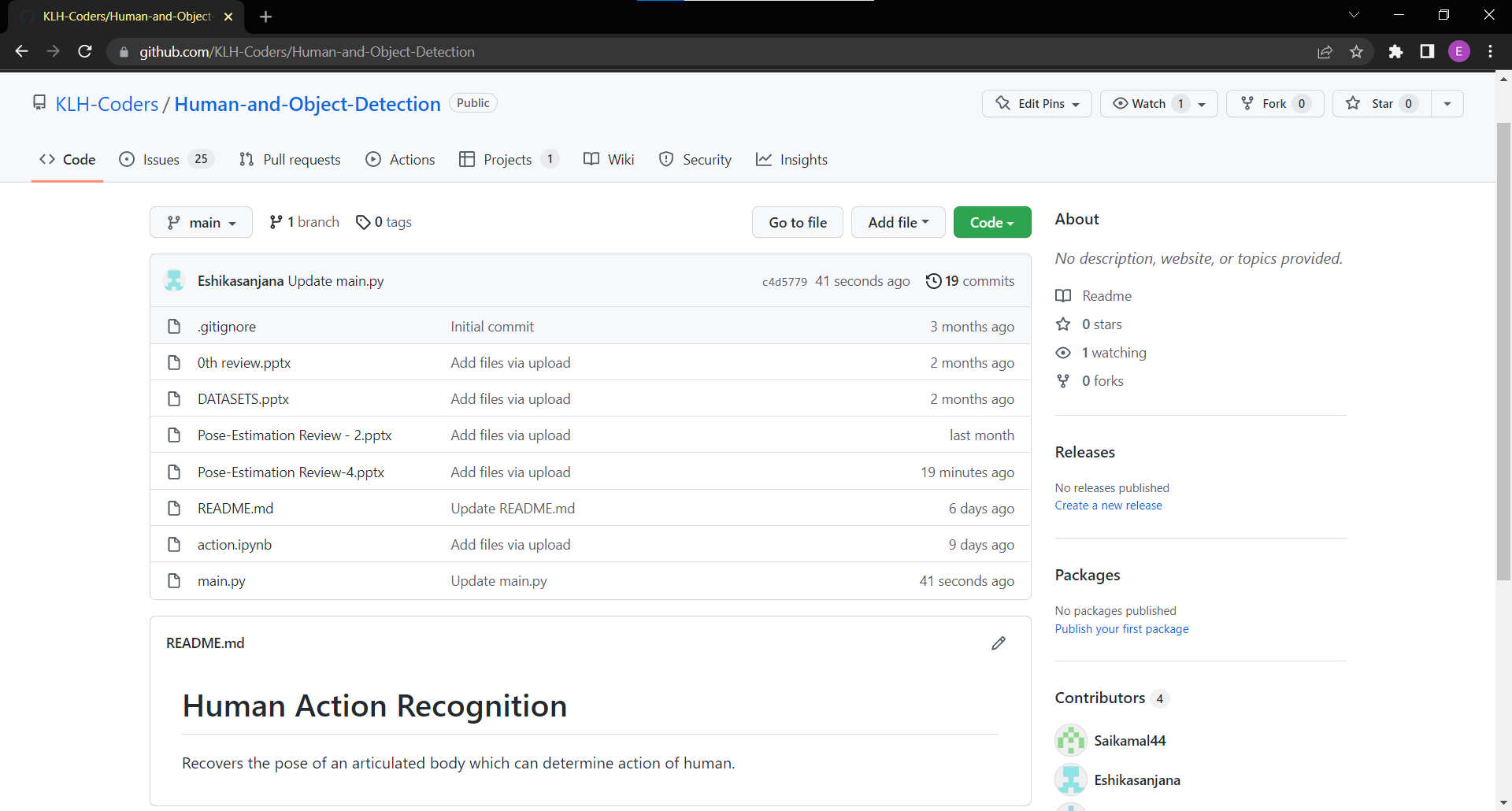


**Fig: 7.4**

**8. GITHUB**

**8.1 GITHUB Repository:**

https://github.com/KLH-Coders/Human-and-Object-Detection

****

1. **REFERENCES**

[1] Sanchez-Caballero, Adrian, David Fuentes-Jimenez, and Cristina Losada-Gutiérrez. "Exploiting the convlstm: Human action recognition using raw depth video-based recurrent neural networks."

[2] Pandya, Meet, Abhishek Pillai, and Himanshu Rupani. "Segregating and Recognizing Human Actions from Video Footages Using LRCN Technique." In International Conference on Advanced Machine Learning Technologies and Applications, pp. 3-13. Springer, Singapore, 2020.

[3] Alireza Tavakkoli, Richard Kelley, Christopher King, Mircea Nicolescu,Monica Nicolescu, and George Bebis ,A Visual Tracking Framework for Intent Recognition in Videos

[4] M. Bregonzio, S. Gong, T. Xiang, et al. Recognising action as clouds of spacetime interest points. In CVPR, volume 9, pages 1948–1955, 2009.

[5] T. M. Breuel. The effects of hyperparameters on sgd training of neural networks. arXiv preprint arXiv:1508.02788, 2015.

[6]. Sepp Hochreiter, Jurgen Schmidhuber, LONG SHORT-TERM MEMORY, 1997.

[7]. Alex Sherstinsky, Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network, August 2018.

[8]Duan, Haodong, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. "Omni-sourced webly-supervised learning for video recognition." In European Conference on Computer Vision, pp. 670-688. Springer, Cham, 2020.

[9] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry et al. "Learning transferable visual models from natural language supervision." In International Conference on Machine Learning, pp. 8748-8763. PMLR, 2021.

[10] Zhe Cao, Student Member, IEEE, Gines Hidalgo, Student Member, IEEE,

Tomas Simon, Shih-En Wei, and Yaser Sheikh , 18 dec 2018

[11] Osokin, Daniil. "Real-time 2d multi-person pose estimation on CPU: Lightweight OpenPose." arXiv preprint arXiv:1811.12004 (2018).

[12] McNally, William, Kanav Vats, Alexander Wong, and John McPhee. "EvoPose2D: Pushing the boundaries of 2d human pose estimation using accelerated neuroevolution with weight transfer." IEEE Access 9 (2021): 139403-139414.

[13] Artacho, Bruno, and Andreas Savakis. "Unipose: Unified human pose estimation in single images and videos." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7035-7044. 2020.

[14] Chen, Ching-Hang, and Deva Ramanan. "3d human pose estimation= 2d pose estimation+ matching." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7035-7043. 2017.

[15] Luvizon, Diogo C., David Picard, and Hedi Tabia. "2d/3d pose estimation and action recognition using multitask deep learning." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5137-5146. 2018.

[16] Liu, Wu, Qian Bao, Yu Sun, and Tao Mei. "Recent Advances in Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective." arXiv preprint arXiv:2104.11536 (2021)

[17] Andriluka, Mykhaylo, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. "Posetrack: A benchmark for human pose estimation and tracking." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5167-5176. 2018.

[18] Zheng, Ce, Wenhan Wu, Taojiannan Yang, Sijie Zhu, Chen Chen, Ruixu Liu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. "Deep learning-based human pose estimation: A survey." arXiv preprint arXiv:2012.13392 (2020).