Human Movement Detection using Recurrent Convolutional Neural Networks

M. Kusuma, M.Vijaya Lakshmi, K. Sai Krishna

Abstact: Human Movement detection is vital in Tele-presence Robots, Animations, Games and Robotic movements. By using Traditional methods with the help of sensor suits it is difficult to find and interpret the movements. As it includes so much sensor data which is difficult to interpret, find the action and send to long distances. It is also very expensive and bulky too. Image processing and computer vision provides a solution to detect and interpret Human movement based on R-CNN approach. It is cheap, easy and light weight algorithm. It takes the video input and divides it in to frames, then it is Human body is separated for the background image. This paper mainly focused on skeleton, its major points and its relative positions in successive picture frames. A set of frames (Video) is given as input to the model, so that the model compares the coordinates of the successive frames and estimates the movement. First, the human is identified and separated from the rest of the image by drawing a bounding box around the human by using CNN (Convolution neural networks), then by applying R-CNN human is segmented and converted to skeleton. From the shape of the skeleton we can identify whether the skeleton is that of a human or not. Comparing the relative coordinates of skeletons extracted from frames photographed over time gives the movement of the human and its direction.

Keywords: Video, Deep Learning, R-CNN.

I. INTRODUCTION

Human Movement detection is vital in Tele-presence Robots, Animations, Games and Robotic movements. By using Traditional methods with the help of sensor suits it is difficult to find and interpret the movements. As it includes so much sensor data which is difficult to interpret, find the action and send to long distances. The experiments have been conducted under different illumination levels, at different times of day and at different locations. Better results were obtained in all cases. This algorithm is capable of tracking human beings in different environments, under different lighting conditions. This work is mainly useful for robot vision and robot walking dynamics. A set of frames (Video) is given as a input to the model, so that the model compares the coordinates of the successive frames and estimates the movement. First, the human is identified and separated from the rest of the image by drawing a bounding box around the human by using CNN (Convolution neural networks), then by applying R-CNN human is segmented and converted to skeleton. The rest of this paper is explained as follows section 2 deals with literature review, section 3 explained the proposed method, section 4 describes the results and discussion and finally section 5 is conclusion and future scope.

II. LITERATURE REVIEW

Dhriti Sengupta proposed skeleton algorithm for efficient implementation of human shape variation[1,2]. Comaniciu used active camera for detection and tracking of human faces[3]. Human pose estimation is proposed [4,5,6,7]. Deep residual learning methods were proposed for image recognition [8,9,10]. Insafutdinov proposed a method for multi person pose estimation, but this method takes more time[11]. The pairwise representations used are difficult to regress precisely and thus a separate logistic regression is required[12].

III. PROPOSED METHOD

The major step in designing the processing is the Pre-Processing task. It is very time consuming and most important than anything else in the project. The pre-processing cleanses the data well and sends it to the model for main high-end computation. The dataset is collected from various open-source databases like COCO. The images are renamed and formatted to a single .png format for a consistent computation. The images can be of any format, but we went ahead with .png format. Re-Sizing it, allocating images to specific folders by name, and then given to the model for the training.

The overall flow of the proposed method is shown in the figure.1. The input given is the image dataset and Model building program, which in return generates a program that can be deployable in any hardware and software development kit or environment for making it into an end-to-end product. So, the output program is the program where test input image and trained weight file must be given to get the output.

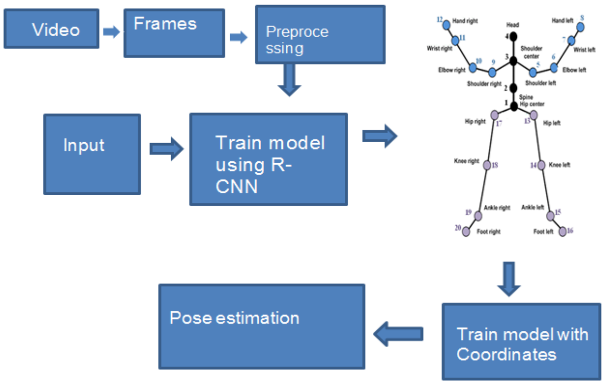


Figure:1. Internal Block Diagram

Revised Manuscript Received on September 14, 2019.

M.Kusuma sri, Asst.prof, Anurag Group of Institutions Hyderabad, Telangana, India.(Email: kusumasri.personal2@gmail.com)

M.Vijaya Lakshmi, Asst.prof, AGI, Hyderabad, Telangana, India.(Email: vijayalakshmiece@cvsr.ac.in)

K.Sai Krishna, Asst.prof, AGI, Hyderabad, Telangana, India.(Email: saikrishnaece@cvsr.ac.in)

Retrieval Number: L109010812S19/2019©BEIESP
DOI: 10.35940/ijitee.L1090.10812S19

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The image dataset is given to the model after pre-processing irrespective of the Deep Learning framework. Whether it is CNN or Transfer Learning, the pre-processing will be the same and the outer block diagram will be the same. The features extracted will be the same in each and every model. Both CNN and Transfer Learning uses Global View parameters. Here in this paper we used pre trained coco dataset model weights, so as to speed up the process of training.

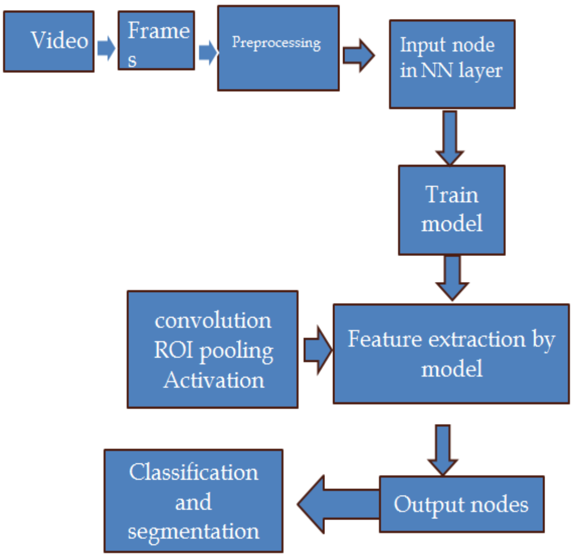

Video
Frame s
Preprocessing
Input node in NN layer
Train model
convolution ROI pooling Activation
Feature extraction by model
Classification and segmentation
Output nodes

Figure: 2. Algorithmic flowchart of R-CNN

Nearly 2 million images from open sourced dataset COCO is taken and a model is trained using Convolution neural networks to identify different objects in a image and to classify them. Objects from the test input are first identified and classified in to different objects. From these objects human (required object is selected) and a bounding box is drawn in order to separate human from the image which is called Region of Interest (ROI). To this ROI, RCN (Region convolved networks) is applied to segment the human from the background.

The first step is the Convolution operation which extracts the features from the image. Convolution learns the image features to represent the relationship between pixels using small squares of input data. It's a mathematical operation that takes two inputs such as image matrix and a filter, this is called "Feature Map".

2.3.1 Padding

Padding operation is followed after Convolution. It allows us to use a CONV layer without necessarily shrinking the height and width of the volumes. Height/width is shrinking as we go into deeper layers without padding. It helps us keep more information at the border of an image. A 2-Stride operation is used in order to avoid the overlapping and reduce the computation.

2.3.2 Pooling

Pooling Layers helps to reduce the number of parameters when the images are too large. Pooling performs the nonlinear down-sampling so it reduces the number of parameters needs to learn by the network. Spatial pooling or subsampling reduces the dimensionality of each map but retains the important information. Max Pooling operation is performed which takes maximum value from the rectified feature map. This avoids the data corruption and hence, guarantees all features are rightly chosen and taken into account.

Finally in flattening matrix converted into vector and connected to fully connected layer like neural network. Almost 10 billion+ parameters are extracted from the training set which gives highest accuracy possible in classifying the human and parameters as well from a test image.

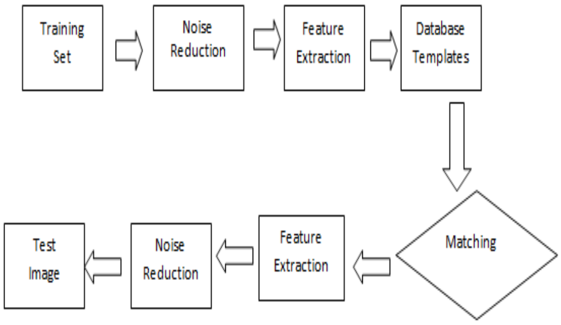


figure.3. Flow chart for Identifying Human

- Inception v4 has specialized "Reduction Blocks" which are used to change the width and height of the grid.
- In Inception-ResNet, for residual addition to work, the input and output after convolution must have the same dimensions. Hence, 1x1 convolutions were used after original convolutions, to match the depth sizes.

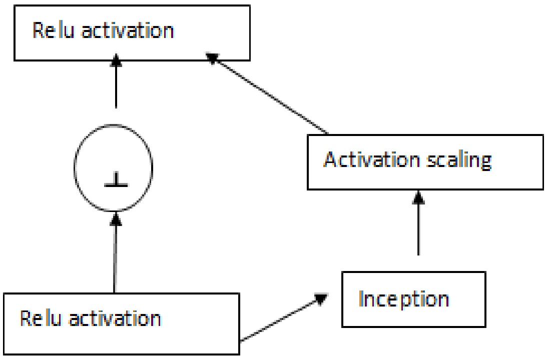


Figure.4. Activations are scaled by a constant to prevent the network from dying.

IV. RESULTS & DISCUSSIONS

Python with Tensorflow and Open cv library is used to achieve the results. We pass the input such as an image or an video to the program to perform the analysis. We can also process the live webcam feed. Proposed method used Anaconda IDE environment to test the code.To analyze performance of proposed method, we collect videos with number of people in moving. The original size of frame is 1080×1920, those are resized into 368×654 during testing to fit in GPU memory. The runtime analysis is performed on a laptop with NVIDIA GeForce GTX-1080 GPU.

Fig Input Frame of a video.

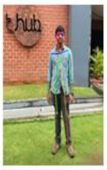


Fig Skeletonised human of input frame.

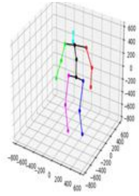


Fig Segmented and skeletonised plotted image.

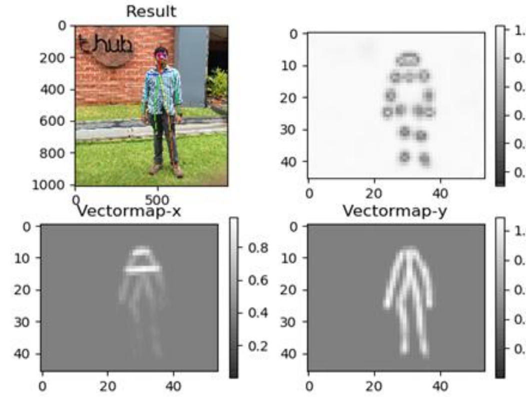
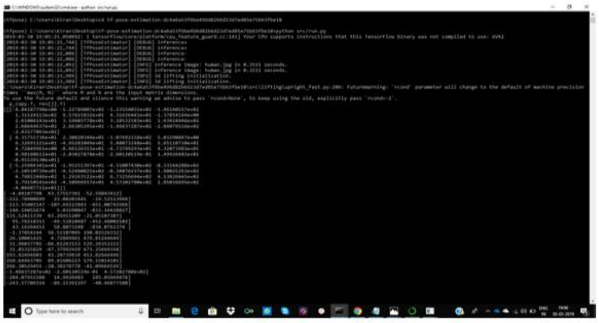


Figure. 5 vector plots



```
(tfpose) C:\Users\Kiran\Desktop>cd tf-pose-estimation-dc4a6a53f6be496d82b6d23d7ed85e75b63fbe10

(tfpose) C:\Users\Kiran\Desktop\tf-pose-estimation-dc4a6a53f6be496d82b6d23d7ed85e75b63fbe10>python src/run.py
2019-03-30 19:05:21.050092: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
[2019-03-30 19:05:21,768] [TfPoseEstimator] [DEBUG] inference+
[2019-03-30 19:05:21,768] [TfPoseEstimator] [DEBUG] inference+
[2019-03-30 19:05:22,086] [TfPoseEstimator] [DEBUG] inference-
[2019-03-30 19:05:22,086] [TfPoseEstimator] [DEBUG] inference-
[2019-03-30 19:05:22,092] [TfPoseEstimator] [INFO] inference image: human.jpg in 0.3511 seconds.
[2019-03-30 19:05:22,092] [TfPoseEstimator] [INFO] inference image: human.jpg in 0.3511 seconds.
[2019-03-30 19:05:23,989] [TfPoseEstimator] [INFO] 3d lifting initialization.
[2019-03-30 19:05:23,989] [TfPoseEstimator] [INFO] 3d lifting initialization.
C:\Users\Kiran\Desktop\tf-pose-estimation-dc4a6a53f6be496d82b6d23d7ed85e75b63fbe10\src\lifting\upright_fast.py:206: FutureWarning: `rcond` parameter will change to the default of machine precision
times ``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  p_copy.T, res[j].T)
[illegible]
```

Figure 7 Input Frame of a video

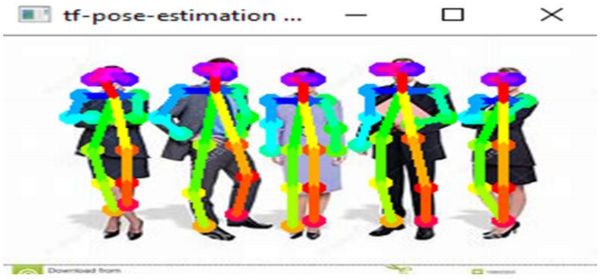


Figure 8 Segmented and skeletonised plotted image

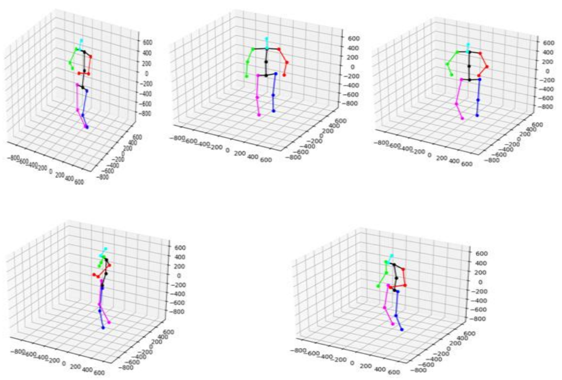
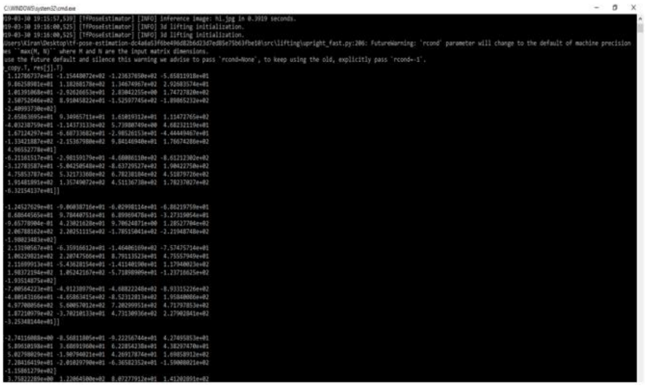


Figure 9 Individual 3d plots of humans in the frame



```
19-03-30 19:15:57,539] [TfPoseEstimator] [INFO] inference image: hi.jpg in 0.3919 seconds.
19-03-30 19:16:00,525] [TfPoseEstimator] [INFO] 3d lifting initialization.
19-03-30 19:16:00,525] [TfPoseEstimator] [INFO] 3d lifting initialization.
Users\Kiran\Desktop\tf-pose-estimation-dc4a6a53f6be496d82b6d23d7ed85e75b63fbe10\src\lifting\upright_fast.py:206: FutureWarning: `rcond` parameter will change to the default of machine precision
es ``max(M, N)`` where M and N are the input matrix dimensions.
use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
_copy.T, res[j].T)
 1.12786737e+01 -1.15448072e+02 -1.23637050e+02 -5.65811918e+01
 9.86258981e+01  1.18268178e+02  1.34674967e+02  2.92683574e+01
 1.01391068e+01 -2.92626653e+01  2.83042255e+00  1.74727820e+02
 2.50752646e+02  8.91045822e+01 -1.52597745e+02 -1.89865232e+02
-2.40993730e+02]
 2.65863695e+01  9.34965711e+01  1.61019312e+01  1.11472765e+02
-4.03238759e+01 -1.14373133e+02  5.73980749e+00  4.68232119e+01
 1.67124297e+01 -6.60733682e+01 -2.98526153e+01 -4.44449467e+01
-1.33421887e+02 -2.15367980e+02  9.84146940e+01  1.76674286e+02
 4.96552778e+01]
-6.21161517e+01 -2.98159179e+01 -4.68086110e+02 -8.61212302e+02
-3.12783587e+01 -5.04250548e+02 -8.63729527e+02  1.90422750e+02
 4.75853787e+02  5.32173368e+02  6.78238184e+02  4.51879726e+02
 1.91481891e+02  1.35749072e+02  4.51136738e+02  1.78237027e+02
-6.32154137e+01]]

-1.24527629e+01 -9.06038716e+01 -6.02998114e+01 -6.88219759e+01
 8.68644565e+01  9.78440751e+01  6.89969478e+01 -3.27319054e+01
-9.65778904e-01  4.23021628e+01  9.70624871e+00  1.28527704e+02
 2.06788162e+02  2.20251115e+02 -1.78515041e+02 -2.21948748e+02
-1.98823483e+02]
 2.13190567e+01 -6.35916612e+01 -1.46406169e+02 -7.57475714e+01
 1.06229821e+02  2.20747566e+01  8.79113523e+01  4.75557949e+01
 2.11699913e+01 -5.43628154e+01 -1.41140190e+01  1.17940023e+02
 1.98372194e+02  1.05242167e+02 -5.71898909e+01 -1.23716625e+02
-1.93514875e+02]
-7.00564223e+01 -4.91238979e+01 -4.68822248e+02 -8.93315226e+02
-4.80143166e+01 -4.65863415e+02 -8.52312813e+02  1.95840086e+02
 4.97700856e+02  5.60057012e+02  7.20299951e+02  4.71797853e+02
 1.87210979e+02 -3.70210133e+01  4.73130936e+02  2.27902841e+02
-3.25348144e+01]]

-2.74116088e+00 -8.56811805e+01 -9.22256744e+01  4.27495853e+01
 5.89610198e+01  3.60691960e+01  6.22854238e+01  4.38297470e+01
 5.02798029e+01 -1.90794021e+01  4.26917874e+01  1.09858912e+02
 7.28416419e+01 -2.01029790e+01 -6.36582352e+01 -1.59080021e+02
-1.15861279e+02]
 3.75822289e+00  1.22064580e+02  8.07277912e+01  1.41202891e+02
```

Figure 10 coordinates of human individually

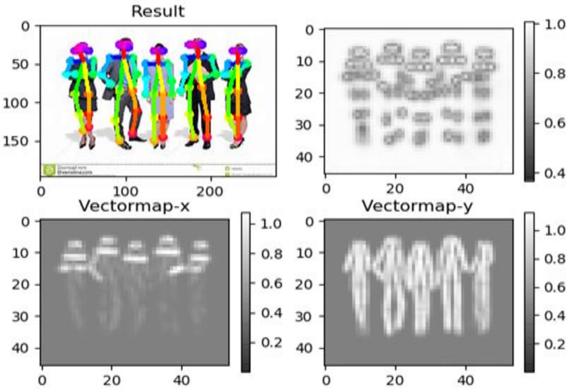


Figure 11 Vector plots of human

A video is given as input to the system, which feeds the frames of the video to the trained model as in Figures 7. The model then performs the convolution operation followed by pooling, padding, activation and applies RPN(Region propagation networks) and gives a skeleton structure to the humans in the frame as shown in the figures 8. To the coordinates of skeleton of human, pyopengl library draws the 3d plots and also gives the coordinates of human as shown in figures 9. using Mat plot lib library vector plots are plotted as shown in the Figures 10 and 11. These coordinates obtained while applying the algorithm to the datasets are used to train the model which recognizes the pose of the human. By comparing pose of human in successive frames movement can be detected.

V. CONCLUSION

Video frames are analysed by applying R-CNN algorithm and transformed in to 3d space by which coordinates are obtained. These co-ordinates should be given to the trained model of data sets having people performing different actions and movements there by actions or movements of human are identified. This is made by using pre-trained weights of coco dataset, as training a neural network requires heavy use of computational power by GPUs.

VI. FUTURE SCOPE

Human detection, Segmentation, Skeletenization is performed and coordinates are obtained. A model can be trained with the coordinates obtained by the frames of different poses and movements of human. To this Model, a video can be given, so that it can identify the pose and movement of the human in the video. As the human body is segmented from the background and the skeleton, human skeleton coordinates were obtained. These coordinates can be calibrated and used to program in multimedia to make animated characters to move like the human. The model can be effectively trained and can be employed in Automatic surveillance of CC T.V feeds to detect any suspicious activities. The coordinates obtained as a result can be calibrated to move a Tele-presence Robot. With the advent of increasing of computing power day by day it is possible to increase speeds algorithm.

REFERENCES

1. Dhriti Sengupta, Merina Kundu, Jayati Ghosh Dastidar, 2014. "Human Shape Variation - Efficient Implementation using Skeleton", IJACR, Vol-4, No-1, Issue-14, pg-145-150, March-2014.
2. Blum, H. 1973. "Biological shape and Visual Science", J. Theoretical Biology, Vol 38, pp 205-287.
3. Comaniciu, D. and Ramesh, V. 2000. "Robust detection and tracking of human faces with an active camera.", IEEE International Workshop on Visual Surveillance.
4. M. Andriluka, S. Roth, and B. Schiele. Monocular 3D pose estimation and tracking by detection. In CVPR, 2010.
5. V. Belagiannis and A. Zisserman. Recurrent human pose estimation. In 12th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2017.
6. A. Bulat and G. Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In ECCV, 2016.
7. X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In NIPS, 2014.
8. P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. In IJCV, 2005.
9. G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. In CVPR, 2014.
10. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
11. E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A deeper, stronger, and faster multiperson pose estimation model. In ECCV, 2016.
12. H. W. Kuhn. The hungarian method for the assignment problem. In Naval research logistics quarterly. Wiley Online Library, 1955.