HUMAN ACTIVITY RECOGNITION USING GENERATIVE ADVERSARIAL NETWORKS

### A PROJECT REPORT

***Submitted by***

## BALUMURI AMRUTHA [REGISTER NO:211417104013] SANA RUSHITHA REDDY [REGISTER NO:211417104235] USIRIKE SWETHA REDDY [REGISTER NO:211417104285]

***in partial fulfillment for the award of the degree of***

### BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MARCH 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“HUMAN ACTIVITY RECOGNITION USING GENERATIVE ADVERSARIAL NETWORK”** is the bonafide work of “**AMRUTHA.B (REG.NO.211417104013), SANA RUSHITHA REDDY (REG.NO.211417104235),**

**USIRIKE SWETHA REDDY (211417104285)”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Dr.T.JACKULIN,M.E.,(Ph.D)**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**Assistant Professor - CSE Department**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva- Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors

### Tmt.C.VIJAYARAJESWARI, Thiru.C.SAKTHIKUMAR,M.E., and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,**

for the support extended throughout the project.

We would like to thank my **Project Guide T.JACKULIN, M.E., Ph.D** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

### NAME OF THE STUDENTS BALUMURI AMRUTHA SANA RUSHITHA REDDY USIRIKE

**SWETHA REDDY**

**ABSTRACT**

Human pose estimation is an important problem in the field of Computer Vision. Now-a-days, this world is more on automation, we make capturing all the activities in our surroundings using surveillances and cameras. It is difficult for computer to determine their poses for the analysis process. Pose Estimation is predicting the body part or joint positions of a person from an image or a video.

This technology will have huge implications. Applications may include video surveillance, assisted living, Advanced driver assistance systems and sports analysis. Human are flexible they can change their poses frequently. To analysis the human movement positions we use Generative Adversarial Network (GAN) which is an unsupervised machine learning algorithm.

GAN can be trained to generate images from random noises. GAN contains generator and discriminator in which generator generates the fake samples using noises and the discriminator tries to classify the fake and real images. GAN has the objective to produce a complex output from a simple input.

iv

### TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iv |
|  | **LIST OF FIGURES** | vii |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** | viii |
| **1.** | **INTRODUCTION** | 01 |
|  | 1.1 Overview | 01 |
|  | 1.2 Problem Definition | 04 |
| **2.** | **LITERATURE SURVEY** | 05 |
| **3.** | **SYSTEM ANALYSIS** | 07 |
|  | 3.1 Existing System | 07 |
|  | 3.2 Proposed system | 08 |
|  | * 1. Requirement Analysis and specifications      1. Hardware Environment      2. Software Environment   3.3 Software Specification | 10  10  10  11 |
| **4.** | **SYSTEM DESIGN** |  |
|  | * 1. Use Case Diagram   2. Activity Diagram   3. Sequence Diagram | 21  22  23 |
| **5.** | * 1. Data Flow Diagram   2. Collaboration Diagram   **SYSTEM ARCHITECTURE** | 24  25 |
|  | 5.1 Architecture Overview | 26 |
|  | 5.2 Module Design Specification | 26 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO.** | **TITLE** | **PAGE NO.** |
| **6.** | **SYSTEM IMPLEMENTATION** | 31 |
| **7.** | **SYSTEM TESTING**   * 1. Testing Techniques   2. white box testing   3. black box testing | 46  46  46  47 |
| **8.** | * 1. software testing strategies 47   2. integration testing 48   **CONCLUSION** 49 | |
|  | Conclusion | 49 |
|  | **APPENDICES** | 50 |
|  | Sample Screens 50 | |
|  | Publications | 52 |
|  | **REFERENCES** 53  **WEBSITES** 54 | |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure no**  3.6.4.1 | **Title**  Types of Regression model | **Page no**  16 |
| 3.6.4.2 | Types of Regression model | 16 |
| 3.6.5.1 | Clusters | 18 |
| 3.6.5.2 | Grid based cluster | 20 |
| 4.1 | Use Case Diagram | 21 |
| 4.2 | Activity Diagram | 22 |
| 4.3 | Sequence Diagram | 23 |
| 4.4.1 | Data Flow Diagram (Level 0) | 24 |
| 4.4.2 | Data Flow Diagram (Level 1) | 24 |
| 4.4.3 | Data Flow Diagram (Level 2) | 24 |
| 4.5 | Collaboration Diagram | 25 |
| 5.1 | System Architecture | 26 |
| 5.2 | Methodology step by step | 27 |
| 5.2.3.1 | Architecture of GAN | 29 |
| 5.2.3.2 | GAN | 30 |
| 5.2.4 | Example of Detection | 30 |

**LIST OF SYMBOLS, ABBREVIATIONS**

GAN- Generative Adversarial Network CNN- Convolutional Neural Network DNN- Deep Neural Network

HAR- Humanitarian Action Report OCR- Optical character Recognition PAF- Part Affinity Fields

UML- Unified Modelling Language

FLIC- Functional Language Intermediate Code LSP- Language Server Protocol

CPMs- Convolutional Pose Machines HRI- Human Robot Interaction

## OVERVIEW

**CHAPTER 1 INTRODUCTION**

This world is counting on challenges of computer vision for its automation. We face many problems during the detection of human pose. In advance, self-driving car has a system to recognize the human pose which detects through cameras in the front and the rear. It is very difficult for it to recognize the human pose when the car is in motion. Those images get blurred and get poor quality images. In running state this system must be capable to recognize those blurred images and process and tune the images. In cricket, bowling action need to be perfect if not then that particular ball will not be considered to be counted.

The bowler must follow the rule which is the angle between the upper and lower arm during the bowling action as the arm passes above shoulder height, measure again when the ball released, the difference must be no more than 15 degrees. At the point of release the angle is almost always zero degrees. These actions must be processed soon to determine the ball for its validity. For these we need to use computer for faster processing the actions need to be captured using cameras. For a better computer vision for human pose recognition we use Generative Adversarial Network algorithm. The input source for our model is through video surveillances or through any cameras. The captured videos will undergoes following 3 processes:

1. Detection of person on the video feed.
2. Key point generation on the detected human
3. Processing with GAN algorithm module. Using the above process the recognition of the human pose can be done even with the blurred image.

1111

Human body pose recovery, or pose recovery in short, refers to the process of estimating the conﬁguration of the underlying kinematic structure of a person. Vision- based approaches are often used to provide such a solution, using cameras as sensor inputs. Human pose estimation is one of the key problems in computer vision that has been being studied well over 15 years.

The reason for its importance is the abundance of applications that could beneﬁt from such a technology. For instance, human pose estimation would allow for higher level reasoning in the context of human-robot interaction (HRI) and activity recognition. A potential solution for Gan pose tracking is to require the human to wear specialized markers so that the visual sensors can locate the markers on the human body and infer the human’s kinematic pose (i.e. OptiTrack System). However, using markers has issues as well. The markers may be sensitive to lighting and other environmental conditions. Wearing multiple markers can also be cumbersome to the user.

First, learning-based methods which rely on prior probabilities for human poses, and assume therefore limited motions. Second, model-free methods which do not use any a priori knowledge, and recover articulated structures automatically. However, the articulated structure is likely to change in time, when encountering a new articulation for instance, hence making identiﬁcation or tracking diﬃcult. Third, model-based approaches which ﬁt and track a known model using image information. In many applications, only one camera is available. In such cases, either only RGB data is considered when still images are available, or it can be combined with temporal information when input images are provided in a video sequence. Most of pose recovery approaches recover the human body pose in the image plane, since, until

recently, 2d pose estimation was the main focus of investigation.

Despite many years of research, however, pose estimation remains a very diﬃcult and still largely unsolved problem. Among the most signiﬁcant challenges are variability of human visual appearance in images,

* + Variability in lighting conditions,
  + Variability in human physique,
  + Partial occlusions due to self articulation and layering of objects in the scene,
  + Complexity of human skeletal structure,
  + High dimensionality of the pose,
  + The loss of 3d information that results from observing the pose from 2d plan a image projections.

In conclusion, several software packages have been developed for human pose estimation from RGB images or depth images, being the ones based on CNNs the most promising ones. For the incorporation of these software packages in robotic applications in real time it is fundamental to use them from ROS. In general, the modules for 2d pose estimation from single images are the most developed and there are even some quite eﬀective open source packages that have reached a certain popularity. In particular, OpenPose (based on CNN architecture) is an eﬃcient open-source software for 2d real-time multi-person pose estimation.

On the other hand, considering the availability of hardware to carry out this work (we have a Logitech C300 webcam and a Kinect One camera that will be described in the next chapter), we considered the implementation of a package ROS that allows 2d pose estimation from simple RGB images.

Additionally, a ROS node that can obtain 3d pose estimation from the initial 2d pose estimation and a depth image synchronized with the RGB image (as in the Kinect camera), by means of projecting pose 2d pose estimation onto the point-cloud of the depth image, has been developed . In this way we can have a software package adaptable to the available hardware, that will allow us to obtain 2d pose estimation or 3d pose estimation depending on the type of available images.

## PROBLEM STATEMENT

**Generative adversarial networks** (**GANs**) are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework. These are also able to capture and copy the variations within a data set.

Activity detection is a major problem in smart videos surveillance. It is a fundamental problem in computer vision, i.e. to detect the activity of human in surveillance videos. These applicants need real-time detection performance, but it is generally vey time consuming to detect the actual activity. Human Activity Recognition or HAR for short, is the problem of predicting what a person is doing based on a trace of their movement using sensors. It is a challenging problem because there is no clear analytical way to relate the sensor data to specific actions in a general way.

## CHAPTER 2

**LITERATURE SURVEY**

* + 1. M. Andriluka, L et al. (2014) designed a methodology for the Face aging with conditional generative adversarial network in which it has been recently shown that Generative Adversarial Networks (GANs) can produce synthetic images of exceptional visual fidelity. In this work, we propose the GAN-based method for automatic face aging. Contrary to previous work employing GANs for altering of facial attributes, we make a particular emphasize on preserving the original person’s identity in the aged version of his/her face. To this end, we introduce a novel approach for “Identity- Preserving” optimization of GAN’s latent vectors. The objective evaluation of the resulting aged and rejuvenated face images by the state-of-threat face recognition and age estimation solutions demonstrate the high potential of the proposed method.
    2. S. Belongie, et al. (2017) designed a methodology named “Do as I do” motion transfer in which given a source video of a person dancing, we can transfer that performance to a novel (amateur) target after only a few minutes of the target subject performing standard moves. We approach this problem as video-to-video translation using pose as an intermediate representation. To transfer the motion, we extract poses from the source subject and apply the learned pose-to-appearance mapping to generate the target subject. We predict two consecutive frames for temporally coherent video results and introduce a separate pipeline for realistic face synthesis. Although our method is quite simple, it produces surprisingly compelling results.

This motivates us to also provide a forensics tool for reliable synthetic content detection, which is able to distinguish videos synthesized by our system from real data. In addition, we release a first of- its-kind open-source dataset of videos that can be legally used for training and motion transfer.

* + 1. B. Sapp and B. Taskar designed a methodology in the name of Learning to discover cross-domain relations with generative adversarial networks in which humans easily recognize relations between data from different domains without any supervision; learning to automatically discover them is in general very challenging and needs many ground-truth pairs that illustrate the relations. To avoid costly pairing, the task of discovering cross-domain relations given unpaired data is addressed. A method based on generative adversarial networks was developed that learn to discover relations between different domains.

## CHAPTER 3 SYSTEM ANALYSIS

* 1. **EXISTING SYSTEM**

Optical Character Recognition (OCR) is a type of document image analysis where a scanned digital image that contains either machine printed or handwritten script is input into an OCR software engine and translating it into an editable machine readable digital text format (like ASCII text). Exiting system provide less accuracy. Less number of input taken.

Convolutional Neural Networks (CNNs)can be in corporated into the pose machine framework for learning image features and image-dependent spatial models for the task of pose estimation. At least in the very near future, CNNs are the most promising in creating general detection, tracking, and recognition modules for human-aware robots. A sequential architecture composed of convolutional networks directly operate on belief maps from previous stages, producing increasingly reﬁned estimates for part locations, without the need for explicit graphical model-style inference.

Convolutional Pose Machines (CPMs) consist of a sequence of convolutional networks that repeatedly produce 2d belief maps for the location of each part. At each stage, image features and belief maps produced by the previous stage are used as input. The belief maps provide the subsequent stage an expressive non-parametric encoding of the spatial uncertainty of location for each part, allowing the CPM to learn rich image dependent spatial models of the relationships between parts.

7

The overall proposed multi-stage architecture is fully diﬀerentiable and therefore can be trained in an end-to end fashion using back propagation. At a particular stage in the CPM, the spatial context of part beliefs provide strong disambiguating cues to a subsequent stage. As a result, each stage of a CPM produces belief maps with increasingly reﬁned estimates for the locations of each part. In order to capture long- range interactions between parts, the design of the network in each stage of our sequential prediction framework is motivated by the goal of achieving a large receptive ﬁeld on both the image and the belief maps. Based on CPM architecture, OpenPose is an eﬃciente method for multi-person pose estimation (Figure 19) what uses a nonparametric representacion of association scores via Part Aﬃnity Fields (PAFs), a set of 2d vectors ﬁelds that encode the location and orientation of limbs over the image domain.

## PROPOSED SYSTEM

Human Activity Recognition using Generative Adversarial Network is based on Deep learning, it provides a lot of data set as an Input to the software tool which will be recognized by the machine and similar pattern will be taken out from them. It can be used as an Octave as a building tool for this product but Octave is recommended in initial state as it’s free and easy to use. The Implementation of such a tool depends on two factors – Feature extraction and classification algorithm. So you can use various classifiers available online and also read about basic feature extraction algorithm. The basic version of product can be implemented in Octave with limited training data set and simple component analysis.

8

The overall architecture of our proposed system is used in a manner in which the Capturing of the real world using surveillances cameras, using these videos we need to detect the human pose. The process 1Detection of person, we need to distinguish the object and person on the video feed. Since they only need personon the video we try to omit the objects. For each frame of the video we need to detect for the person. The process-2 key point generation, in this process we need to detect the coordinate points or key points on the person, these coordinate points are the joints of the human body. Every action of the human depends and varies with and every coordinate points. Determining the coordinate point should be done for each person on the video feed. Process-3 Processing with GAN.

This algorithm has 2 components generator and discriminator, which are used to determine with more efficiency. The Generator is used to generate all possible fake images using noises for ach frame of the video. Then the discriminator is used to distinguish the fake and the real images. By creating all possible fake images of the human pose detected on the video frame we can possibly generate some blurred images which would be more useful during the process of detecting the human pose at the real time. By comparing the faked images along with the real image the process of algorithm can make a fixed decision to react. Which make this process more efficient than the other system.

## Requirement Analysis and Specifications

These are the requirements for doing the project. Without using these tools and software, the project can’t be done. So we have two requirements to do the project.

They are

* + 1. Hardware Requirements
    2. Software Requirements

### HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

* + - * Processor : Intel i5
      * Hard disk : 1 TB
      * RAM :4 GB (minimum)

### SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* Windows 7 and above(64-bit)
* Language: Python (python libraries)
* Software Interface: Anaconda Navigator

## SOFTWARE SPECIFICATION

### MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers. Machine learning is a continuously developing field. Because of this, there are some

considerations to keep in mind as you work with machine learning methodologies or analyze the impact of machine learning processes.

Here in this thesis, we are providing basic info of the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning

### SUPERVISED LEARNING

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data.

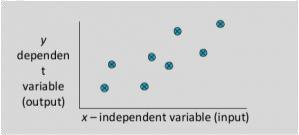
### CLASSIFICATION

As the name suggests, Classification is the task of “classifying things” into sub- categories. But by a machine. If that doesn’t sound like much, imagine your computer being able to differentiate between you and a stranger. Between a potato and a tomato. Between an A grade and a F.In Machine Learning and Statistics, Classification is the problem of identifying to which of a set of categories (sub populations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

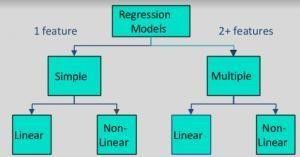
### Type of Classification:

Classification is of two types, They are Binary classification and Multiclass Classification.

### REGRESSION

A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used; the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

**Fig:3.6.4.1** Types of Regression Model



**Fig:3.6.4.2** Types of Regression Model

### UNSUPERVISED LEARNING

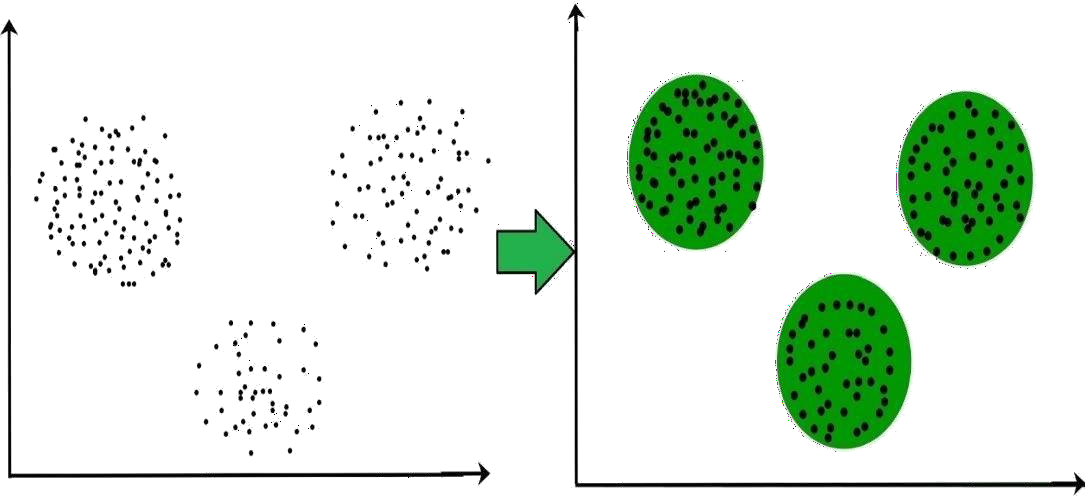
Unsupervised learning is where we only have input data (X) and no corresponding

output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answer and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems.

### CLUSTERING

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them. For example, the data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



**Fig:3.6.5.1** Clusters

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for calculation of the outliers. Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

### Clustering Methods:

1. **Density-Based Methods:** These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters.Example DBSCAN (Density-Based Spatial Clustering of Applications with Noise) , OPTICS (Ordering

Points to Identify Clustering Structure) etc.

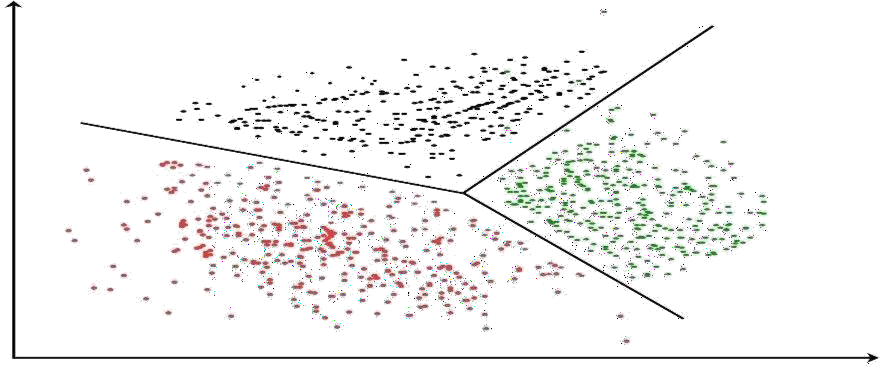
1. **Hierarchical Based Methods :** The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category

* Agglomerative (bottom up approach)
* Divisive (top down approach).

1. **Partitioning Methods :** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon randomized Search) etc.
2. **Grid-based Methods:** In this method the data space are formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example STING (Statistical Information Grid), wave cluster, CLIQUE (CLusteringIn Quest) etc.

Clustering Algorithms:

* + K-Means Clustering.
  + Mean-Shift Clustering for a single sliding window.
  + The entire process of Mean-Shift Clustering.
  + DBSCAN Smiley Face Clustering.
  + EM Clustering using GMMs.
  + Agglomerative Hierarchical Clustering.



**Fig:3.6.5.2** Grid Based cluster.

## CHAPTER 4 SYSTEM DESIGN

* 1. **USE CASE DIAGRAM:**

USER INPUT



DATA SET

PERSON DETECTION

ACTOR (PERSON) CO-ORDINATE POINTS

INTERFACE

KEY POINT IDENTIFICATION

MODEL TRAINER

DETECTION

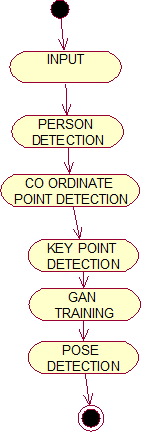
**Fig 4.1** USE CASE DIAGRAM

The above figure represents use case diagram, in which user upload dataset The functional modules such as person detection, co-ordinate points and model trainer are

shown. Dataset are analyzed for fake or real.

17

## ACTIVITY DIAGRAM:



**Fig 4.2** Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities actions with support for choice, iteration and concurrency. The above figure shows Activity diagrams for fake review detection process.

18

## SEQUENCE DIAGRAM:

POSE DETECTION

PERSON DETECTION

GAN TRAINING

KEY POINT DETECTION

USER INPUT



POSE DETECTION

GANTRAINING

KEY POINT DETECTION

PERSON DETECTION

USER INPUT



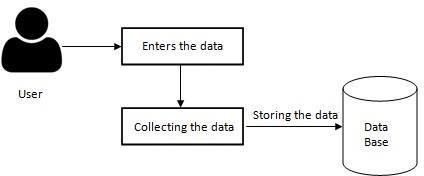
: ACTOR

**Fig 4.3** Sequence Diagram

A sequence diagram shows a parallel vertical lines, different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in order in which they occur. The above figure represents sequence diagram, the proposed system’s sequence of data flow is represented.

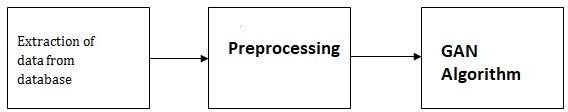
## DATA FLOW DIAGRAM:

### LEVEL 0:



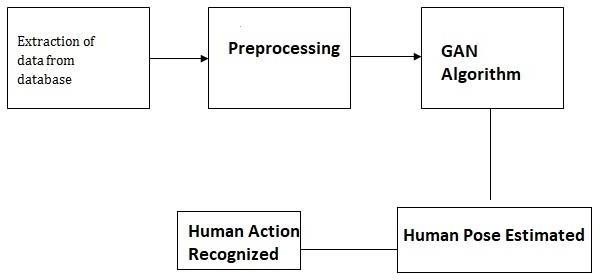
**Fig 4.4.1** Data flow diagram (level 0)

### LEVEL 1:



**LEVEL 2:**

**Fig 4.4.2** Data flow diagram (level 1)

**Fig 4.4.3** Data flow diagram (level 2)

## COLLABORATION DIAGRAM:

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object.

# 1: INPUT 2: IDENTIFICATION



3: DETECTION

7: MATCHING

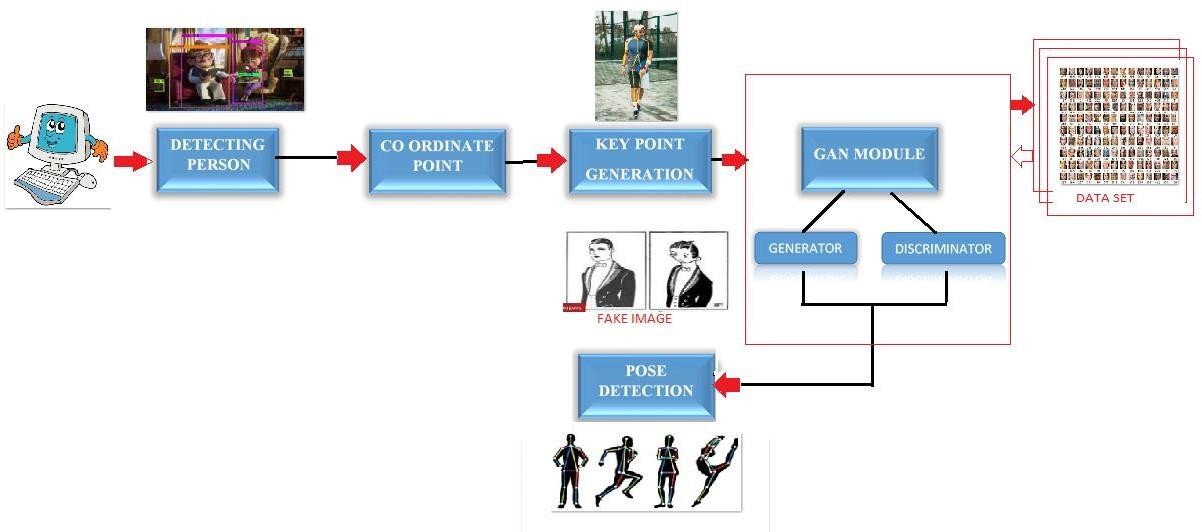
4: ANALYSIS

**Fig 4.5** Collaboration Diagram

## CHAPTER 5 SYSTEM ARCHITECTURE

* 1. **ARCHITECTURE OVERVIEW:**

System architecture is the conceptual model that defines the structure, behaviour, and more views of a system .An architecture description is a formal ascription and representation of a system, organized in a way that supports reasoning about the structures and behaviour of the system.



**Fig 5.1** System Architecture

### Steps:

* + 1. **DETECTING A PERSON**:

To detect a person on the frame, using cluster grouping algorithm on a set of detection areas which define a set of features based on spatial, color and temporal information for each detection. Then using these features, we cluster the detections. We finally define a measure to calculate the actual number of people within each cluster to infer the final estimation of the number of people in the scene.

### COORDINATE POINT or KEY POINT:

To determine the pose of the human and to classify the pose structure, we need to determine the coordinate points such as elbow, shoulder, neck, knee, hip, toe, etc., which might help us to determine exact pose of the human.

### GENERATIVE ADVERSARIAL NETWORK (GAN) ALGORITHM:

GAN has two parts generator and discriminator the generator generates fake samples of data(be it an image, audio, etc.) and tries to fool the Discriminator. The Discriminator, on the other hand, tries to distinguish between the real and fake samples. The Generator and the Discriminator are both Neural Networks and they both run in competition with each other in the training phase. The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition.

### POSE DETECTION:

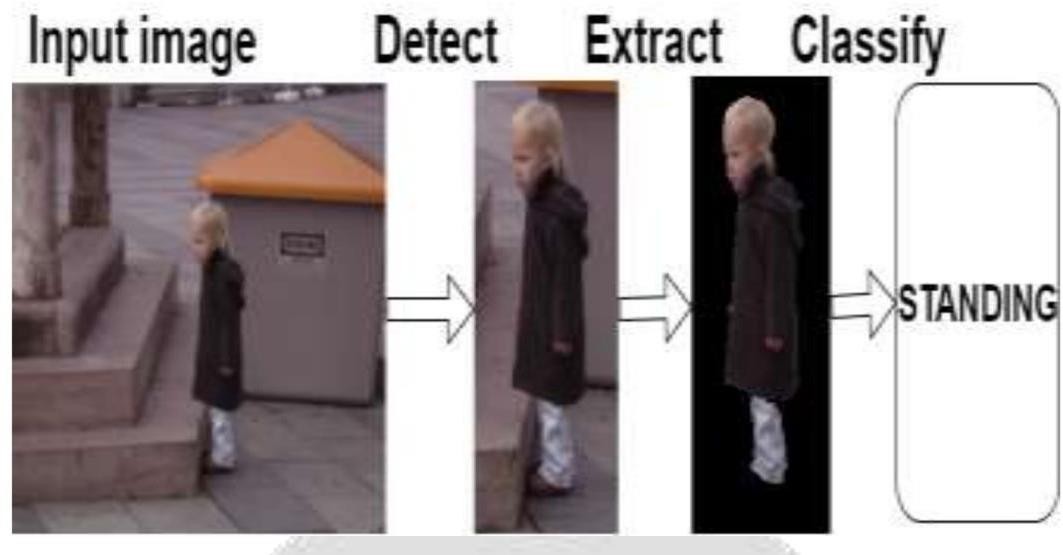
After determining all the possible position of the human image using generator and discriminator, now it would make us very simple to determine exact position of the human and to understand the human gesture.

### MODULE DESIGN SPECIFICATION:

Now-a-days, Recognising the Human pose is the important problem in the field of computer science, As humans are more flexible to change their poses and its being more difficult to estimate the poses.

To analyze this human movement the following modules are used.

* Detecting the person
* Coordinate separation or key point generation
* GENERATIVE ADVERSARIAL NETWORK (GAN) algorithm Pose detection



**Fig 5.2** Methodology step by step

### DETECTING THE PERSON:

* + - * Detecting an object is a primary task in computer vision. It involves identification of presence, location, and type of one or more objects in each photographic frame.
      * It is a challenging problem that involves building upon various methods for object recognition, object localization, and classifying the objects.
      * In this process we feed the input as video from the live and it is important to detect the person on the real world.
      * we need to detect multiple person on the video which can be processed using Haar Cascade classifier.

A Haar Cascade basically a classifier which is used to detect objects from the input.

* + - * The haarcascade\_frontalface\_default.xml is a haar cascade designed by OpenCV to detect the frontal face.A Haar Cascade classifier trains by cascading on thousands of

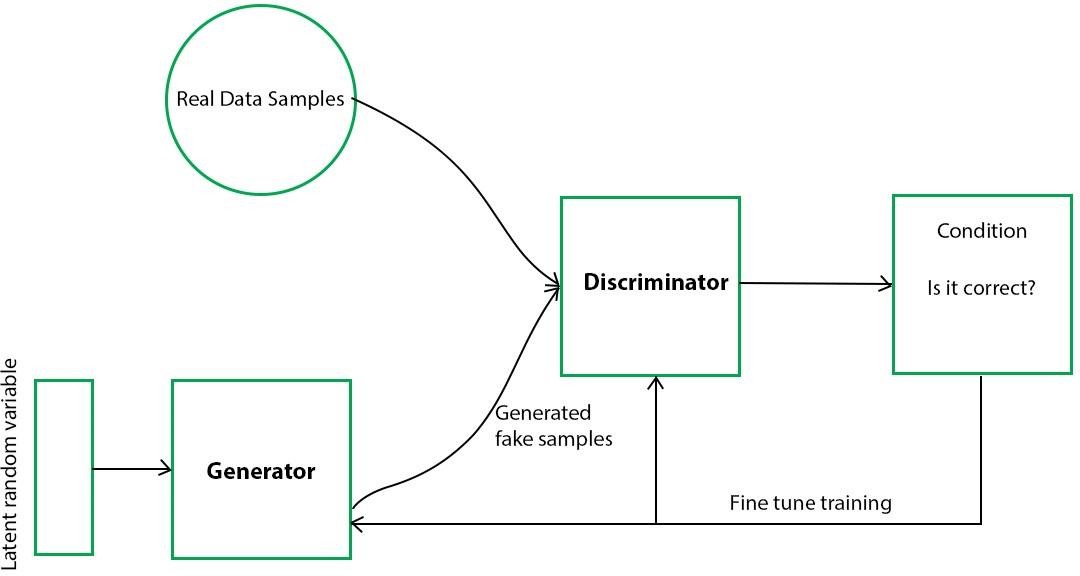
negative images with the positive image superimposed on it.

### COORDINATING POINTS OR KEY POINTS:

* All prior detection systems repurpose classifiers or localizers to perform detection.

They apply the model to an image at multiple locations and scales.

* High scoring regions of the image are considered detections.
* We use a totally different approach. We apply a single neural network to the full image.
* This network divides the image into regions and predicts bounding boxes and probabilities for each region.
* These bounding boxes are weighted by the predicted probabilities.
* These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.
  + 1. **GENERATIVE ADVERSARIAL NETWORKS(GAN):**



**Fig 5.2.3.1** Architecture of GAN

A generative adversarial network (GAN) has two parts:

* The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator.
* The discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results. The discriminator's training data comes from two sources:
* Real data instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
* Fake data instances created by the generator. The discriminator uses these instances as negative examples during training.

### Training the Discriminator:

The discriminator connects to two loss functions. During discriminator training, the

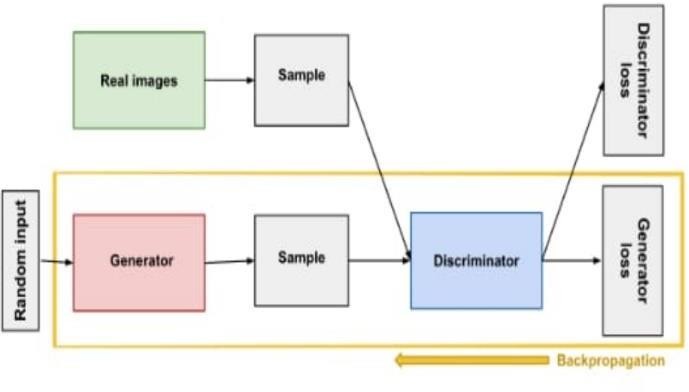
discriminator ignores the generator loss and just uses the discriminator loss. We use the generator loss during generator training, as described in the following. During discriminator training:

1. The discriminator classifies both real data and fake data from the generator.
2. The discriminator loss penalizes the discriminator for misclassifying areal instance as fake or a fake instance as real.
3. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

### The Generator:

The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real. Generator training requires tighter integration between the generator and the discriminator than discriminator training requires. The portion of the GAN that trains the generator includes:

* random input
* generator network, which transforms the random input into a data instance
* discriminator network, which classifies the generated data
* discriminator output
* generator loss, which penalizes the generator for failing to fool the discriminator



**Fig 5.2.3.2** GAN

### POSE DETECTION :

After determining all the possible position of the human image using generator and discriminator, now it would make us very simple to determine exact position of the human and to understand the human gesture.



**Fig 5.2.4** Example of walking human detected and highlighted with green rectangle

## CHAPTER 6

**CODING:**

import os

## SYSYTEM IMPLEMENTATION

from os.path import exists, join, basename, splitext git\_repo\_url = 'alpha.git'

project\_name = splitext(basename(git\_repo\_url))[0] if not exists(project\_name):

# clone and install dependencies

get\_ipython().system('git clone -q -b pytorch --depth 1 $git\_repo\_url') get\_ipython().system('cd $project\_name && pip install -q -r requirements.txt') get\_ipython().system('pip install -q youtube-dl visdom')

import sys sys.path.append(project\_name) import time

import matplotlib

import matplotlib.pylab as plt plt.rcParams["axes.grid"] = False

from IPython.display import YouTubeVideo # ##

Download pretrained models # In[12]:

def download\_from\_google\_drive(file\_id, file\_name): #

get\_ipython().system('rm -f ./cookie')

get\_ipython().system('curl -c ./cookie -s -L "https://drive.google.com/uc?export=download&id=$file\_id" > /dev/null') confirm\_text

= get\_ipython().getoutput("awk '/download/ {print $NF}' ./cookie") confirm\_text = confirm\_text[0]

get\_ipython().system('curl -Lb ./cookie "https://drive.google.com/uc?export=download&confirm=$confirm\_text&id=$file

\_id" -o $file\_name')

pretrained\_model\_path = join(project\_name, 'models/sppe/duc\_se.pth') if not exists(pretrained\_model\_path):

# download the pretrained model download\_from\_google\_drive('1OPORTWB2cwd5YTVBX-NE8fsauZJWsrtW', pretrained\_model\_path)

yolo\_pretrained\_model\_path = join(project\_name, 'models/yolo/yolov3- spp.weights')

if not exists(yolo\_pretrained\_model\_path):

# download the YOLO weights download\_from\_google\_drive('1D47msNOOiJKvPOXlnpyzdKA3k6E97NTC', yolo\_pretrained\_model\_path)

# ## Detect poses on a test video

# We are going to detect poses on the following youtube video:

# In[13]:

YOUTUBE\_ID = 'wKH09Lh6CIk'

YouTubeVideo(YOUTUBE\_ID)

# Download the above youtube video, cut the first 5 seconds and do the pose detection on that 5 seconds:

# In[14]:

get\_ipython().system('rm -df youtube.mp4') # download the youtube with the given ID

get\_ipython().system('youtube-dl -f \'bestvideo[ext=mp4]\' --output "youtube.%(ext)s" https://[www.youtube.com/watch?v=$YOUTUBE\_ID')](http://www.youtube.com/watch?v=%24YOUTUBE_ID%27))

# cut the first 5 seconds

get\_ipython().system('ffmpeg -y -loglevel info -i youtube.mp4 -t 5 video.mp4') # run AlpaPose on these 5 seconds video

get\_ipython().system('rm -rf AlphaPose\_video.avi')

get\_ipython().system('cd $project\_name && python3 video\_demo.py --sp --video

../video.mp4 --outdir .. --save\_video') # convert the result into MP4

get\_ipython().system('ffmpeg -y -loglevel info -i AlphaPose\_video.avi AlphaPose\_video.mp4')

# Finally, visualize the result: #

In[9]:

def show\_local\_mp4\_video(file\_name, width=640, height=480): import io

import base64

from IPython.display import HTML

video\_encoded = base64.b64encode(io.open(file\_name, 'rb').read())

return HTML(data='''<video width="{0}" height="{1}" alt="test" controls>

<source src="data:video/mp4;base64,{2}" type="video/mp4" />

</video>'''.format(width, height, video\_encoded.decode('ascii'))) show\_local\_mp4\_video('AlphaPose\_video.mp4', width=960, height=720) 0Openpose Image

import csv import json import os import warnings

import numpy as np from PIL import Image

from .generator import Generator

from ..utils.image import read\_image\_bgr

def load\_hierarchy(metadata\_dir, version='v4'): hierarchy = None

if version == 'challenge2018':

hierarchy = 'bbox\_labels\_500\_hierarchy.json'

elif version == 'v4':

hierarchy = 'bbox\_labels\_600\_hierarchy.json' elif version == 'v3':

hierarchy = 'bbox\_labels\_600\_hierarchy.json'

hierarchy\_json = os.path.join(metadata\_dir, hierarchy) with open(hierarchy\_json) as f:

hierarchy\_data = json.loads(f.read()) return hierarchy\_data

def load\_hierarchy\_children(hierarchy): res

= [hierarchy['LabelName']] if 'Subcategory' in hierarchy:

for subcategory in hierarchy['Subcategory']: children = load\_hierarchy\_children(subcategory) for c in children:

res.append(c) return res

def find\_hierarchy\_parent(hierarchy, parent\_cls): if hierarchy['LabelName'] == parent\_cls:

return hierarchy

elif 'Subcategory' in hierarchy:

for child in hierarchy['Subcategory']:

res = find\_hierarchy\_parent(child, parent\_cls) if res is not None:

return res return None

def get\_labels(metadata\_dir, version='v4'):

if version == 'v4' or version == 'challenge2018':

csv\_file = 'class-descriptions-boxable.csv' if version == 'v4' else 'challenge- 2018- class-descriptions-500.csv'

boxable\_classes\_descriptions = os.path.join(metadata\_dir, csv\_file) id\_to\_labels = {}

cls\_index = {} i = 0

with open(boxable\_classes\_descriptions) as f: for row in csv.reader(f):

# make sure the csv row is not empty (usually the last one) if len(row):

label = row[0]

description = row[1].replace("\"", "").replace("'", "").replace('`', '') id\_to\_labels[i] = description

cls\_index[label] = i i += 1

else:

trainable\_classes\_path = os.path.join(metadata\_dir, 'classes-bbox- trainable.txt')

description\_path = os.path.join(metadata\_dir, 'class-descriptions.csv') description\_table = {}

with open(description\_path) as f: for row in csv.reader(f):

# make sure the csv row is not empty (usually the last one) if len(row):

description\_table[row[0]] = row[1].replace("\"", "").replace("'", "").replace('`', '')

with open(trainable\_classes\_path, 'rb') as f: trainable\_classes = f.read().split('\n')

id\_to\_labels = dict([(i, description\_table[c]) for i, c in enumerate(trainable\_classes)])

cls\_index = dict([(c, i) for i, c in enumerate(trainable\_classes)]) return id\_to\_labels, cls\_index

def generate\_images\_annotations\_json(main\_dir, metadata\_dir, subset, cls\_index, version='v4'):

validation\_image\_ids = {} if version == 'v4':

annotations\_path = os.path.join(metadata\_dir, subset, '{}-annotations-

bbox.csv'.format(subset))

elif version == 'challenge2018':

validation\_image\_ids\_path = os.path.join(metadata\_dir, 'challenge-2018- image-ids-valset-od.csv')

with open(validation\_image\_ids\_path, 'r') as csv\_file:

reader = csv.DictReader(csv\_file, fieldnames=['ImageID']) next(reader)

for line, row in enumerate(reader): image\_id = row['ImageID'] validation\_image\_ids[image\_id] = True

annotations\_path = os.path.join(metadata\_dir, 'challenge-2018-train- annotations-bbox.csv')

else:

annotations\_path = os.path.join(metadata\_dir, subset, 'annotations-human- bbox.csv')

fieldnames = ['ImageID', 'Source', 'LabelName', 'Confidence', 'XMin', 'XMax', 'YMin', 'YMax',

'IsOccluded', 'IsTruncated', 'IsGroupOf', 'IsDepiction', 'IsInside'] id\_annotations = dict()

with open(annotations\_path, 'r') as csv\_file:

reader = csv.DictReader(csv\_file, fieldnames=fieldnames) next(reader)

images\_sizes = {}

for line, row in enumerate(reader): frame = row['ImageID']

if version == 'challenge2018': if subset == 'train':

if frame in validation\_image\_ids: continue

elif subset == 'validation':

if frame not in validation\_image\_ids: continue

else:

raise NotImplementedError('This generator handles only the train and validation

subsets')

class\_name = row['LabelName'] if class\_name not in cls\_index:

continue

cls\_id = cls\_index[class\_name] if version == 'challenge2018':

# We recommend participants to use the provided subset of the training set as a validation set.

# This is preferable over using the V4 val/test sets, as the training set is more densely annotated.

img\_path = os.path.join(main\_dir, 'images', 'train', frame + '.jpg')

else:

img\_path = os.path.join(main\_dir, 'images', subset, frame + '.jpg') if frame in images\_sizes:

width, height = images\_sizes[frame] else:

try:

with Image.open(img\_path) as img: width, height = img.width, img.height images\_sizes[frame] = (width, height)

except Exception as ex:

if version == 'challenge2018': raise ex

continue

x1 = float(row['XMin']) x2 = float(row['XMax']) y1 = float(row['YMin']) y2 = float(row['YMax'])

x1\_int = int(round(x1 \* width)) x2\_int = int(round(x2 \* width)) y1\_int = int(round(y1 \* height)) y2\_int = int(round(y2 \* height))

# Check that the bounding box is valid. if x2 <= x1:

raise ValueError('line {}: x2 ({}) must be higher than x1 ({})'.format(line,

x2, x1))

if y2 <= y1:

raise ValueError('line {}: y2 ({}) must be higher than y1 ({})'.format(line,

y2, y1))

if y2\_int == y1\_int:

warnings.warn('filtering line {}: rounding y2 ({}) and y1 ({}) makes them equal'.format(line, y2, y1))

continue

if x2\_int == x1\_int:

warnings.warn('filtering line {}: rounding x2 ({}) and x1 ({}) makes them equal'.format(line, x2, x1))

continue

img\_id = row['ImageID']

annotation = {'cls\_id': cls\_id, 'x1': x1, 'x2': x2, 'y1': y1, 'y2': y2} if img\_id in id\_annotations:

annotations = id\_annotations[img\_id] annotations['boxes'].append(annotation)

else:

id\_annotations[img\_id] = {'w': width, 'h': height, 'boxes': [annotation]} return id\_annotations

class OpenImagesGenerator(Generator): def

init (

self, main\_dir, subset, version='v4', labels\_filter=None, annotation\_cache\_dir='.', parent\_label=None,

\*\*kwargs

):

if version == 'challenge2018': metadata = 'challenge2018'

elif version == 'v4': metadata = '2018\_04'

elif version == 'v3': metadata = '2017\_11'

else:

raise NotImplementedError('There is currently no implementation for versions older than v3')

if version == 'challenge2018':

self.base\_dir = os.path.join(main\_dir, 'images', 'train') else:

self.base\_dir = os.path.join(main\_dir, 'images', subset)

metadata\_dir = os.path.join(main\_dir, metadata) annotation\_cache\_json = os.path.join(annotation\_cache\_dir, subset + '.json') self.hierarchy = load\_hierarchy(metadata\_dir, version=version) id\_to\_labels, cls\_index = get\_labels(metadata\_dir, version=version)

if os.path.exists(annotation\_cache\_json): with open(annotation\_cache\_json, 'r') as f:

self.annotations = json.loads(f.read())

else:

self.annotations = generate\_images\_annotations\_json(main\_dir, metadata\_dir, subset, cls\_index, version=version)

json.dump(self.annotations, open(annotation\_cache\_json, "w")) if labels\_filter is not None or parent\_label is not None:

self.id\_to\_labels, self.annotations = self. filter\_data(id\_to\_labels,

cls\_index, labels\_filter, parent\_label) else:

self.id\_to\_labels = id\_to\_labels

self.id\_to\_image\_id = dict([(i, k) for i, k in enumerate(self.annotations)]) super(OpenImagesGenerator, self). init (\*\*kwargs)

def filter\_data(self, id\_to\_labels, cls\_index, labels\_filter=None,

parent\_label=None): """

If you want to work with a subset of the labels just set a list with trainable labels

:param labels\_filter: Ex: labels\_filter = ['Helmet', 'Hat', 'Analog television']

:param parent\_label: If parent\_label is set this will bring you the parent label but also its children in the semantic hierarchy as defined in OID, ex: Animal hierarchical tree

:return:

"""

children\_id\_to\_labels = {} if parent\_label is None:

# there is/are no other sublabel(s) other than the labels itself for label in labels\_filter:

for i, lb in id\_to\_labels.items(): if lb

== label:

children\_id\_to\_labels[i] = label break

else:

parent\_cls = None

for i, lb in id\_to\_labels.items(): if lb

== parent\_label: parent\_id = i

for c, index in cls\_index.items(): if index == parent\_id:

parent\_cls = c

break

if parent\_cls is None:

raise Exception('Couldnt find label {}'.format(parent\_label)) parent\_tree = find\_hierarchy\_parent(self.hierarchy, parent\_cls) if parent\_tree is None:

raise Exception('Couldnt find parent {} in the semantic hierarchical tree'.format(parent\_label))

children = load\_hierarchy\_children(parent\_tree) for cls in children:

index = cls\_index[cls] label = id\_to\_labels[index] children\_id\_to\_labels[index] = label

id\_map = dict([(ind, i) for i, ind in enumerate(children\_id\_to\_labels.keys())]) filtered\_annotations = {}

for k in self.annotations: img\_ann = self.annotations[k] filtered\_boxes

= []

for ann in img\_ann['boxes']: cls\_id = ann['cls\_id']

if cls\_id in children\_id\_to\_labels: ann['cls\_id'] = id\_map[cls\_id]

filtered\_boxes.append(ann)

if len(filtered\_boxes) > 0:

filtered\_annotations[k] = {'w': img\_ann['w'], 'h': img\_ann['h'], 'boxes': filtered\_boxes} children\_id\_to\_labels = dict([(id\_map[i], l) for (i, l) in children\_id\_to\_labels.items()])

return children\_id\_to\_labels, filtered\_annotations def size(self):

return len(self.annotations) def num\_classes(self):

return len(self.id\_to\_labels) def has\_label(self, label):

""" Return True if label is a known label. """ return label in self.id\_to\_labels def

has\_name(self, name):

""" Returns True if name is a known class. """ raise NotImplementedError() def name\_to\_label(self, name): raise

NotImplementedError() def label\_to\_name(self, label): return

self.id\_to\_labels[label]

def image\_aspect\_ratio(self, image\_index):

img\_annotations = self.annotations[self.id\_to\_image\_id[image\_index]] height, width = img\_annotations['h'], img\_annotations['w']

return float(width) / float(height) def image\_path(self, image\_index):

path = os.path.join(self.base\_dir, self.id\_to\_image\_id[image\_index] + '.jpg') return path

def load\_image(self, image\_index):

return read\_image\_bgr(self.image\_path(image\_index)) def load\_annotations(self, image\_index):

image\_annotations = self.annotations[self.id\_to\_image\_id[image\_index]] labels = image\_annotations['boxes']

height, width = image\_annotations['h'], image\_annotations['w']

annotations = {'labels': np.empty((len(labels),)), 'bboxes': np.empty((len(labels),

4))}

for idx, ann in enumerate(labels): cls\_id = ann['cls\_id']

x1 = ann['x1'] \* width x2 = ann['x2'] \* width y1 = ann['y1'] \* height y2 = ann['y2'] \* height

annotations['bboxes'][idx, 0] = x1

|  |  |  |  |
| --- | --- | --- | --- |
| annotations['bboxes'][idx, | 1] | = | y1 |
| annotations['bboxes'][idx, | 2] | = | x2 |
| annotations['bboxes'][idx, | 3] | = | y2 |

annotations['labels'][idx] = cls\_id return annotations

## CHAPTER 7 SYSTEM TESTING

* 1. **TESTING TECHNIQUES:**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system.

This is the last chance to detect and correct errors before the system is installed for user acceptance testing.The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

### WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* + - Flow graph notation
    - Kilometric complexity
    - Deriving test cases
    - Graph matrices Control

### BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

### SOFTWARE TESTING STRATEGIES

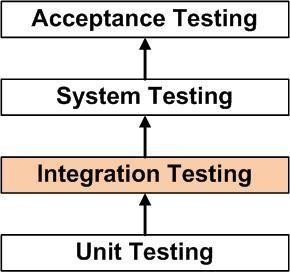
A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

Testing begins at the module level and works “outward” toward the integration.

### INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g components in a software system or – one step up – software applications at the company level – interact without error.

Integration testing is the process of testing the interface between two software units or module. It’s focus on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.



**Fig 7.5.1**

## CHAPTER 8 CONCLUSION

From the above observations in this paper we address that proposed system endures the difficulties in capturing the images with GAN based machine learning model and optimizes the blurred images. This proposed system provides novelty about performance measures related to the quality of human pose detections. This model concise about small variations that capable of increase the amount of real data applications involving live human datasets

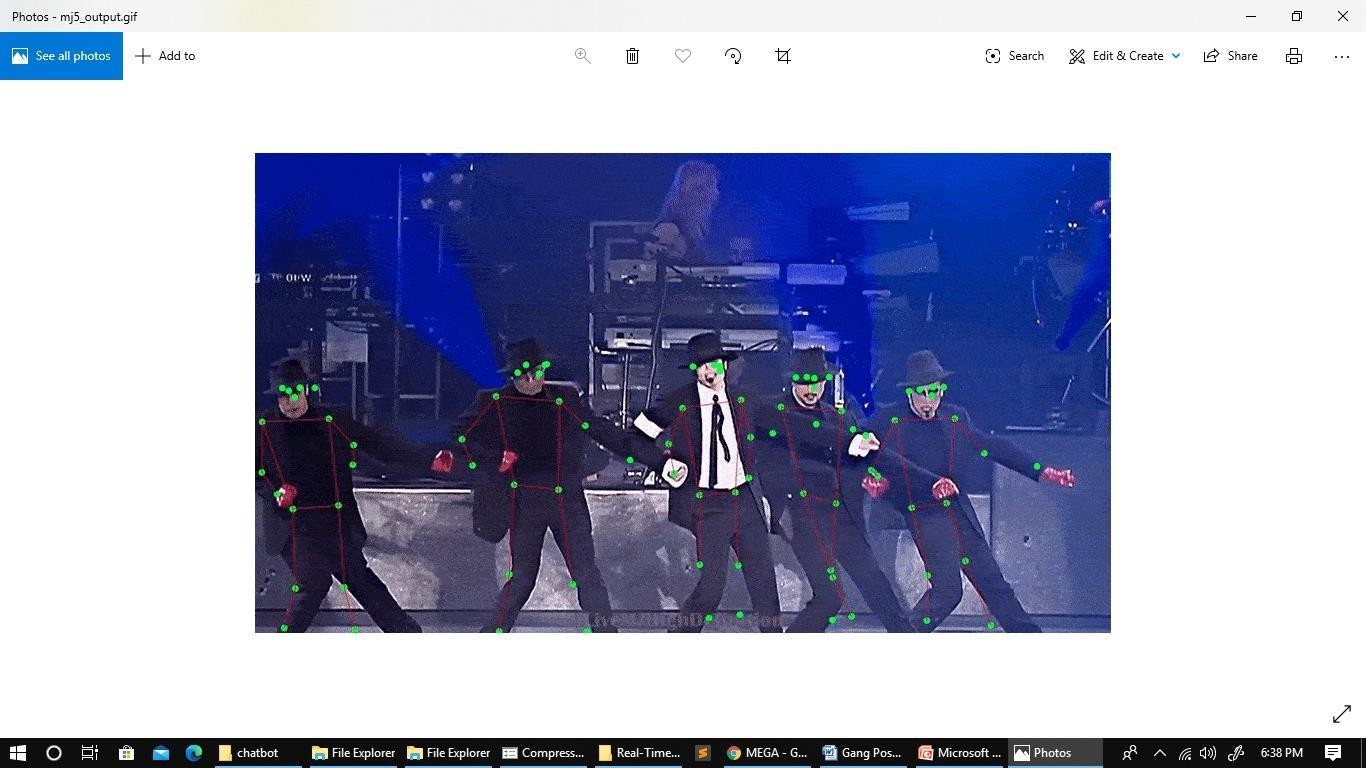
## SAMPLE SCREENS

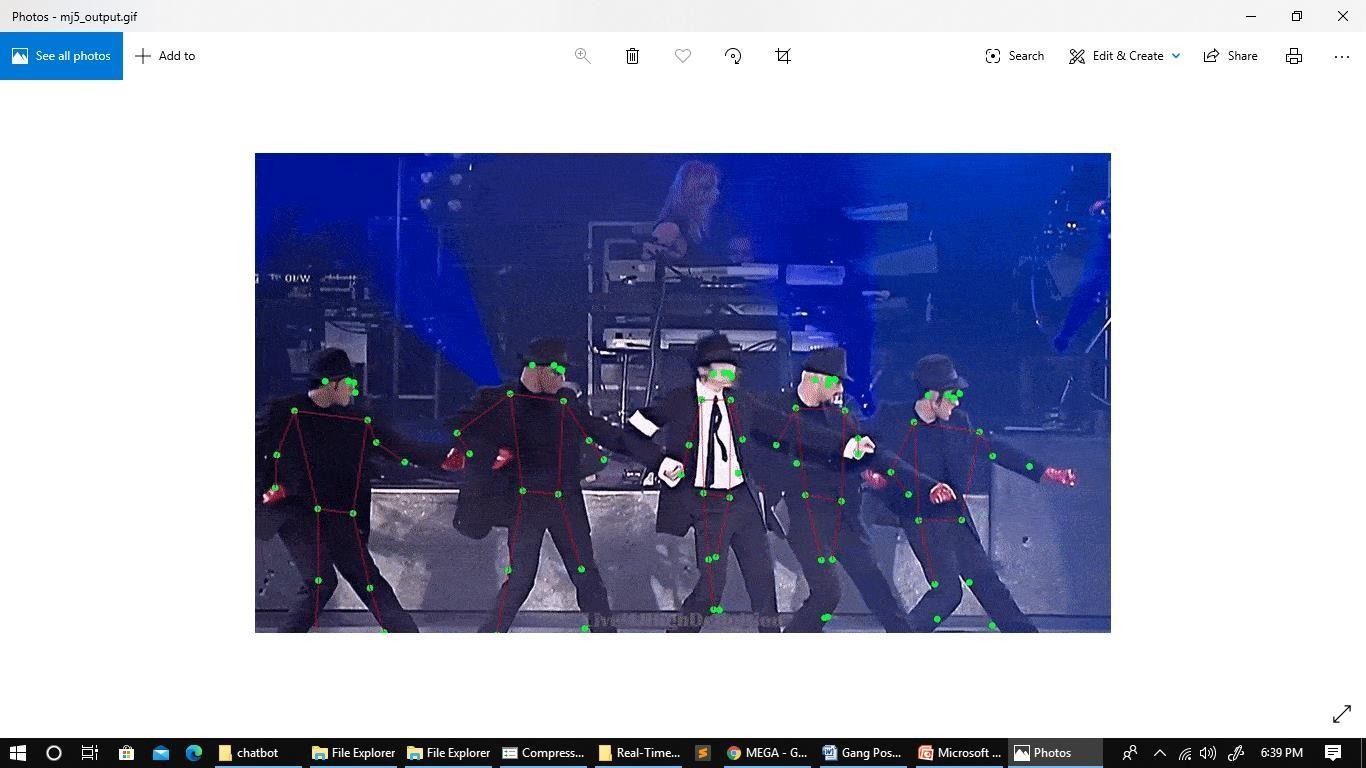
**APPENDICES**

### Co-ordinating points

1. **key generation**



* 1. **Co-ordinating points**
  2. **key generation**



**PUBLICATIONS**

This project was presented as a paper at the Fourth International Conference on Intelligent Computing held at Panimalar Engineering College on 27.03.2021, by Amrutha B, Sana Rushitha Reddy, Usirike Swetha Reddy and N.Indira.

## REFERENCES

### BOOKS REFERENCES:

[1] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, “How far are we from solving pedestrian detection?” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1259–1267.

[2]P. Dollar, Z. Tu, P. Perona, and S. Belongie, “Integral channel features,” ´ in Proc.

Brit. Mach. Vis. Conf., 2009, pp. 91.1–91.11.

1. W. Nam, P. Dollar, and J. H. Han, “Local decorrelation for improved ´ pedestrian detection,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 424– 432.
2. S. Zhang, R. Benenson, and B. Schiele, “Filtered channel features for pedestrian detection.” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1751– 1760. Y. Yang, Z. Wang, and F. Wu, “Exploring prior knowledge for pedestrian detection.” in Proc. Brit. Mach. Vis. Conf., 2015, pp. 176.1–176.12.
3. J. Cao, Y. Pang, and X. Li, “Pedestrian detection inspired by appearance constancy and shape symmetry,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1316–1324.
4. S. Zhang, C. Bauckhage, and A. B. Cremers, “Informed haar-like features improve pedestrian detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 947–954.
5. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, pp. 1627–1645, 2010.

### WEBSITES

* https://towardsdatascience.com/generative-deep-learning-lets-seek-how-ai- extending-not-replacing-creative-process-fded15b0561b
* [https://missinglink.ai/guides/convolutional-neural-networks/generative- adversarial-networks/](https://missinglink.ai/guides/convolutional-neural-networks/generative-adversarial-networks/)
* https://[www.researchgate.net/profile/Yu\_Qiao8/publication/308496682/figu](http://www.researchgate.net/profile/Yu_Qiao8/publication/308496682/figu) re/tbl2/AS:668252325810201@1536335178894/Accuracy-rate-of-CNN- s.png.
* [https://medium.com/activating-robotic-minds/understanding-generative- adversarial-networks-4dafc963f2ef](https://medium.com/activating-robotic-minds/understanding-generative-adversarial-networks-4dafc963f2ef)
* [https://searchenterpriseai.techtarget.com/definition/generative-adversarial- network-GAN](https://searchenterpriseai.techtarget.com/definition/generative-adversarial-network-GAN)
* <https://www.geeksforgeeks.org/generative-adversarial-network-gan/>
* <https://blog.paperspace.com/implementing-gans-in-tensorflow/>