PANIMALAR INSTITUTE OF TECHNOLOGY
CHENNAI – 600 123

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS8611-MINI PROJECT

HUMAN AND OBJECT RECOGNITION

USING VIDEO SURVEILLANCE

A MINI PROJECT REPORT

Submitted by

KALLURI VENKATA DATTA SAI MANI **(211518104062)**

PALADUGU CHARAN **(211518104112)**

VAYUGUNDLA VENKATA SIVA DHANUSH **(211518104179)**

In the sixth semester of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

During the academic year 2020 - 2021

AUGUST 2021

PANIMALAR INSTITUTE OF TECHNOLOGY
CHENNAI – 600 123

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS8611-MINI PROJECT

BONAFIDE CERTIFICATE

Certified that this project report "**HUMAN AND OBJECT RECOGNITION USING VIDEO SURVEILLANCE" is the bonafide work of KALLURI VENKATA DATTA SAI MANI (211518104062), PALADUGU CHARAN (211518104112), VAYUGUNDLA VENKATA SIVA DHANUSH (211518104179)** of Third Year B.E - CSE during the academic year 2020 – 2021 who carried out under my supervision.

SIGNATURE

Dr. V. Subedha, M. Tech., Ph.D..,
Professor and Head, Department of CSE, Panimalar Institute of Technology, Poonamalle, Chennai-600 123

SIGNATURE

Mrs.K. Hema Priya, M.Tech
Assistant professor and Supervisor, Department of CSE, Panimalar Institute of Technology, Poonamallee, Chennai-600 123

Certified that the above candidates were examined in the university **CS8611-Mini project** viva-voce held on **05.08.2021** at Panimalar Institute of Technology, Chennai– 600 123.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

A project of this magnitude and nature requires the kind cooperation and support of many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

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

We also express our sincere thanks to Our **Dynamic Directors Mrs.C. VIJAYARAJESWARI, Mr. C. SAKTHIKUMAR, M.E., Ph.D.,** and **Mrs. S. SARANYA SREE SAKTHIKUMAR, B.E., M.B.A,** for providing us with the necessary facilities for the completion of this project.

We also express our gratefulness to our **Principal Dr.T. JAYANTHY, M.E., Ph.D.,** who helped us in the completion of this project.

We wish to convey our thanks and gratitude to our **Head of the Department Dr. V. SUBEDHA, M.Tech., Ph.D.,** Department of Computer Science and Engineering, for her support and providing us ample time to complete our project.

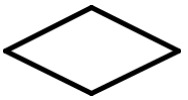
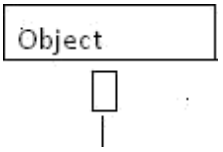
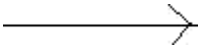
We express our indebtedness and gratitude to our Project supervisor, **Mrs. K. Hema Priya, M. Tech** Assistant Professor**,** Department of Computer Science and Engineering, for her guidance throughout the project.

ABSTRACT

Object recognition is a computer vision technique for identifying objects in images or videos. Object recognition is a key output of deep learning and machine learning algorithms. When humans look at a photograph or watch a video, we can readily spot people, objects, scenes, and visual details.To develop a system which is used to recognize different objects in each frame of the video in video surveillance system and to give an alert when some animals enter into the residential premises.Image processing is used to extract the images from video.Deep learning is used to train the system to identify a different objects and to detect the objects.Big data applications are consuming most of the space in industry and research area. Among the widespread examples of big data, the role of video streams from CCTV cameras is equally important as other sources like social media data, sensor data, agri- culture data, medical data and data evolved from space research. Surveillance videos have a major contribution in unstructured big data. CCTV cameras are implemented in all places where security having much importance. Manual surveillance seems tedious and time consuming. Security can be defined in different terms in different contexts like theft identification, violence detection, chances of explosion etc.

TABLE OF CONTENTS

LIST OF SYMBOLS

S.NO	SYMBOL NAME	NOTATION	DESCRIPTION
1.	Initial Activity		This shows the Starting point or first activity of flow.
2.	Final Activity		The end of the Activity diagram is shown by a bull's eye symbol.
4.	Decision		A logic where a decision is to be made.
5.	Actor		A role that a user plays with respect to system.
6.	Object		A Real time Entity.
7.	Message		To send message between the life of an object.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Deep learning is a branch of machine learning is completely based on artificial neural networks , as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

It provides a systematic and methodical overview of the latest developments in deep learning theory and its applications to computer vision, illustrating them using key topics, including object detection, face analysis, 3D object recognition, and image retrieval. The book offers a rich blend of theory and practice.

For example, image classification is straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition.

Image classification involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all of these problems are referred to as object recognition.

1.1 AIM AND SCOPE OF THEPROJECT:

In our country there is a lack of shortage for surveillance system due to high cost to overcome this shortage we came up with this technique.

The main goal is to detect both persons and objects for clear cut surveillance and ease for recognition of objects.

SYNOPSIS:

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in digital photographs.
Image classification involves predicting the class of one object in an image. Object localization refers to identifying the location of one or more objects in an image and drawing abounding box around their extent. Object detection combines these two tasks and localizes and classifies one or more objects in an image.

As such, we can distinguish between these three computer vision tasks:

- **Image Classification:** Predict the type or class of an object in an image.
  - Input: An image with a single object, such as a photograph.
  - Output: A class label (e.g. one or more integers that are mapped to class labels).
- **Object Localization:** Locate the presence of objects in an image and indicate their location with a bounding box.

Input: An image with one or more objects, such as a photograph.

- Output: One or more bounding boxes (e.g. defined by a point, width, and height).

- **Object Detection:** Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
  - Input: An image with one or more objects, such as a photograph.
  - Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

CHAPTER 2

SYSTEM STUDY

2.1 EXISTING SYSTEM

- The present surveillance system has poor object recognition ability and not much accurate and can be easily manipulated by some experts.
- It cannot differentiate between persons and objects.

2.1.1 LIMITATIONS

- Should monitor in person.
- Low accuracy.
- Low frame capture.

2.2 PROPOSEDSYSTEM

- In a video the images are extracted using the image processing techniques.
- The training to identify the objects is done using deep learning.
- The frames are extracted from the video using deep learning.
- Using Mobile net SSD we detect imaged or objects from the frames extracted frames.

2.2.1 ADVANTAGES

- ➢ Frame detection helps to identify clearly.
- ➢ We can get images with the help of image processing.
- ➢ Objects are detected with their names.

3. LITERATURESURVEY:

2.3.1. Paper Title : Transfer learning based object detection by using convolutional neutral networks.

Author : Ms. Bulbul Bamne, Ms. Neha Shrivatava, MR.

Lokesh Parashar, MR. Upendra Singh

Abstract : Object detection has become an important task for various purposes in our daily lives. Machine learning techniques have been used for this task from earlier but they are used for the classification of image-based species to extract the feature set. This task of deciding the feature set helps to decide the desired object detection. To overcome the object classification problem, this paper proposes a transfer learning-based deep learning method. The different convolutional neural networks (CNN) are studied in this work. Here for the improvement in the result, the majority voting scheme is used. The overall work is carried out on the CUB 200-2011 dataset. The results obtained have shown incredible improvement in the accuracy of the proposed work when compared to the different CNN models.

Drawbacks:

It detects only the images. The fields like Artificial Intelligence, Speech recognition, face recognition.

2.3.2. ProjectTitle : Salient Object Detection

Author : Ali Borji , Ming- Ming Cheng ,Huaizu Jiang and JiaLi.

Abstract : With the rapid development of deep learning techniques, deep convolutional neural networks (DCNNs) have become more important for object detection. Compared with traditional handcrafted feature-based methods, the deep learning-based object detection methods can learn both low-level and high-level image features. The image features learned through deep learning techniques are more representative than the handcrafted features. Therefore, this review paper focuses on the object detection algorithms based on deep convolutional neural networks, while the traditional object detection algorithms will be simply introduced as well. Through the review and analysis of deep learning-based object detection techniques in recent years, this work includes the following parts: backbone networks, loss functions and training strategies, classical object detection architectures, complex problems, datasets and evaluation metrics, applications and future development directions. We hope this review paper will be helpful for researchers in the field of object detection.

Drawbacks:

Salient object detection, saliency, explicit saliency, visual attention, regions of interest, object, segmentation, interestingness, importance, eye movements.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 SOFTWARE SPECIFICATION

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

REQUIREMENTS

- Language : Python
- Platform : OpenCV
- Tool : Deep Leaning toolbox

3.2 HARDWARE SPECIFICATION

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

REQUIREMENTS

- Processor : Intel® core ™ I 3-4030Ucpu@1.90GHZ
- RAM : 4 GB or More
- System type : 64-bit operating system
- Keyboard : Normal or Multimedia
- Mouse : Compatible mouse

CHAPTER 4

SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM:

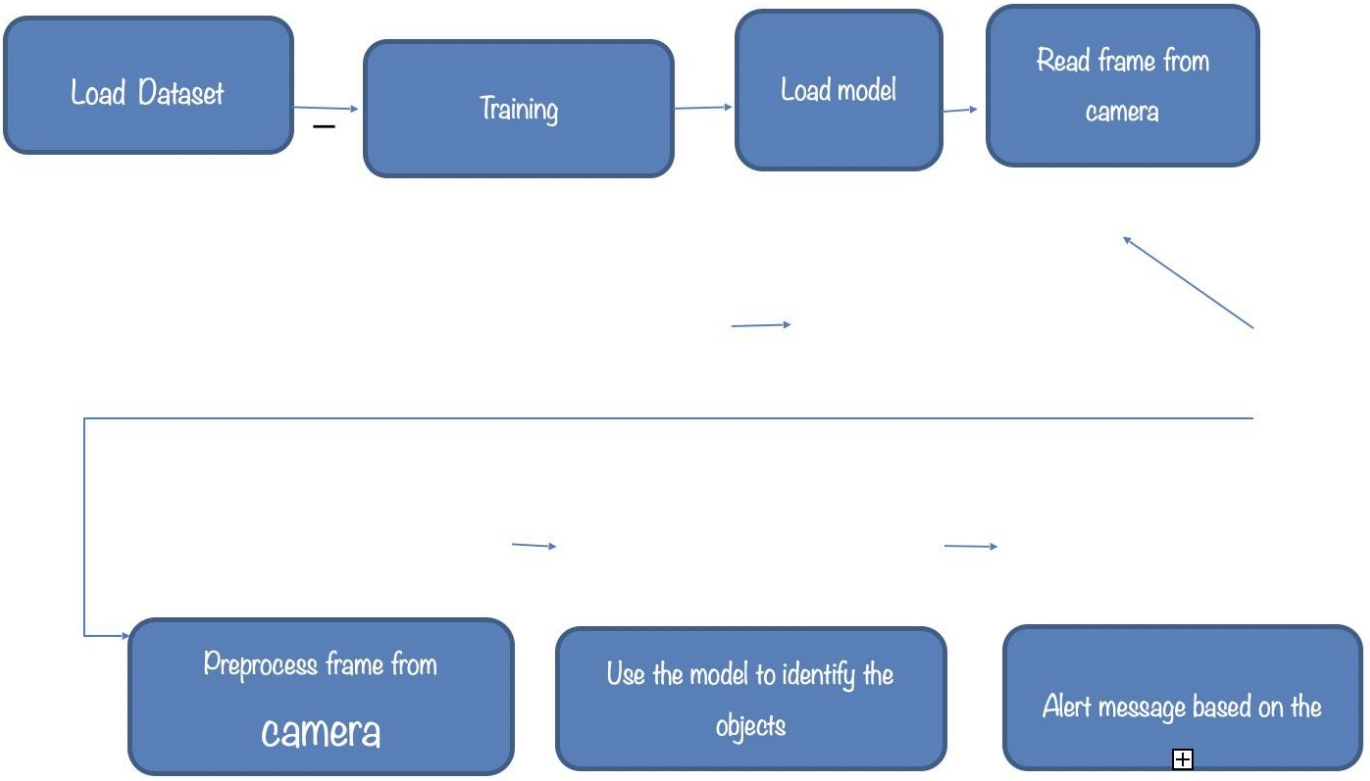


Fig: 4.1 Architecture

4.2 UML DIAGRAMS:

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. It is based on diagrammatic representations of software components.

4.2.1 USE CASE DIAGRAM

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system. Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration. They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

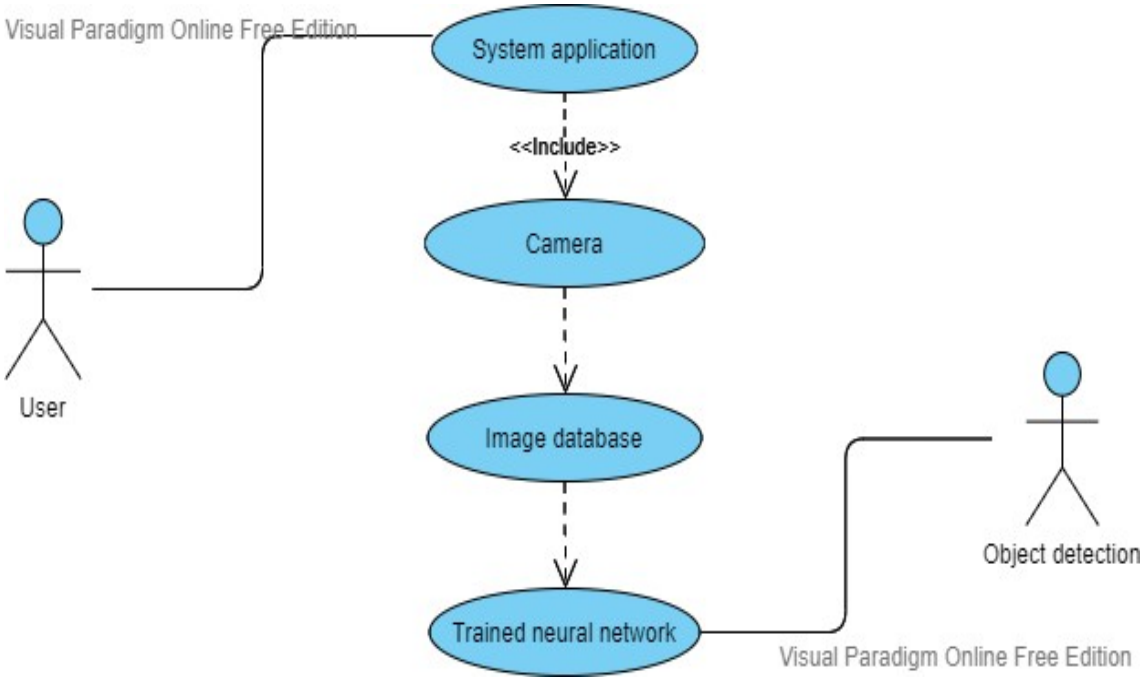


Fig: 4.2.1 Use Case Diagram

4.2.2 CLASS DIAGRAM

A Class diagram is a Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object- oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

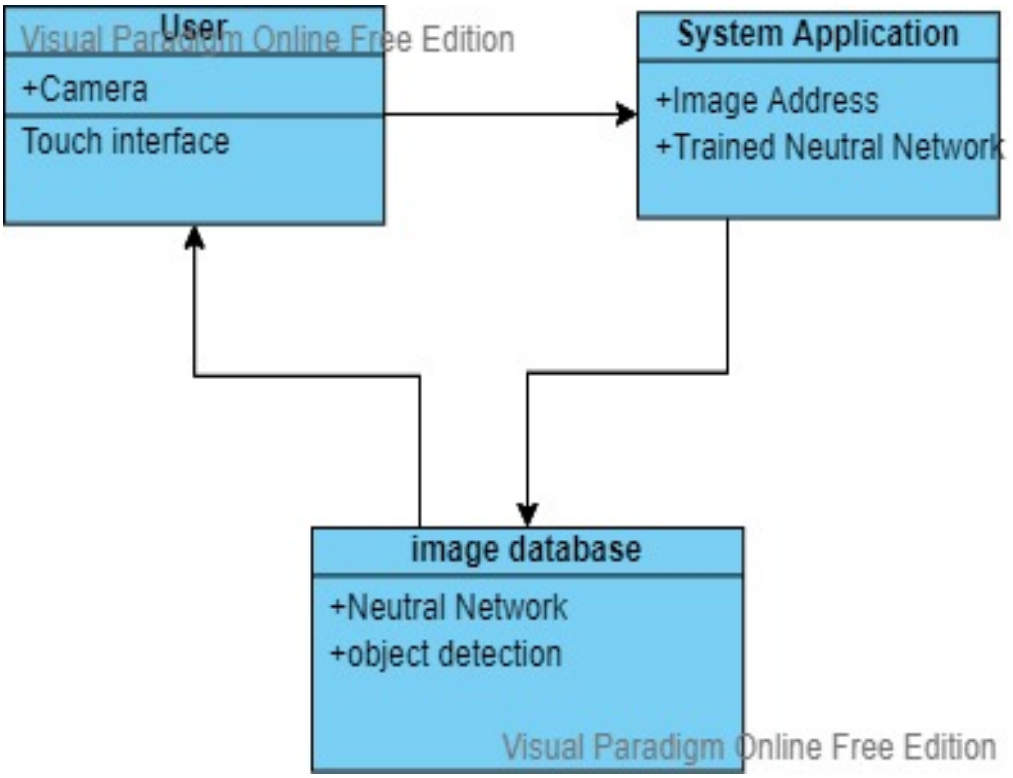


Fig: 4.2.2 Class Diagram

4.2.3 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system.

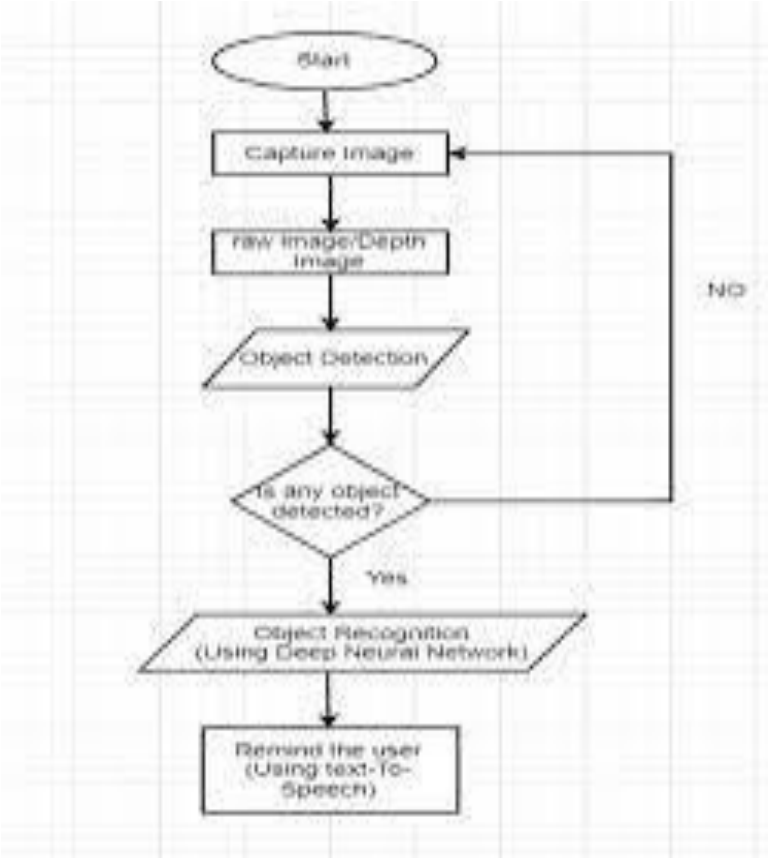


Fig: 4.2.3 Activity Diagram

4.2.4 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

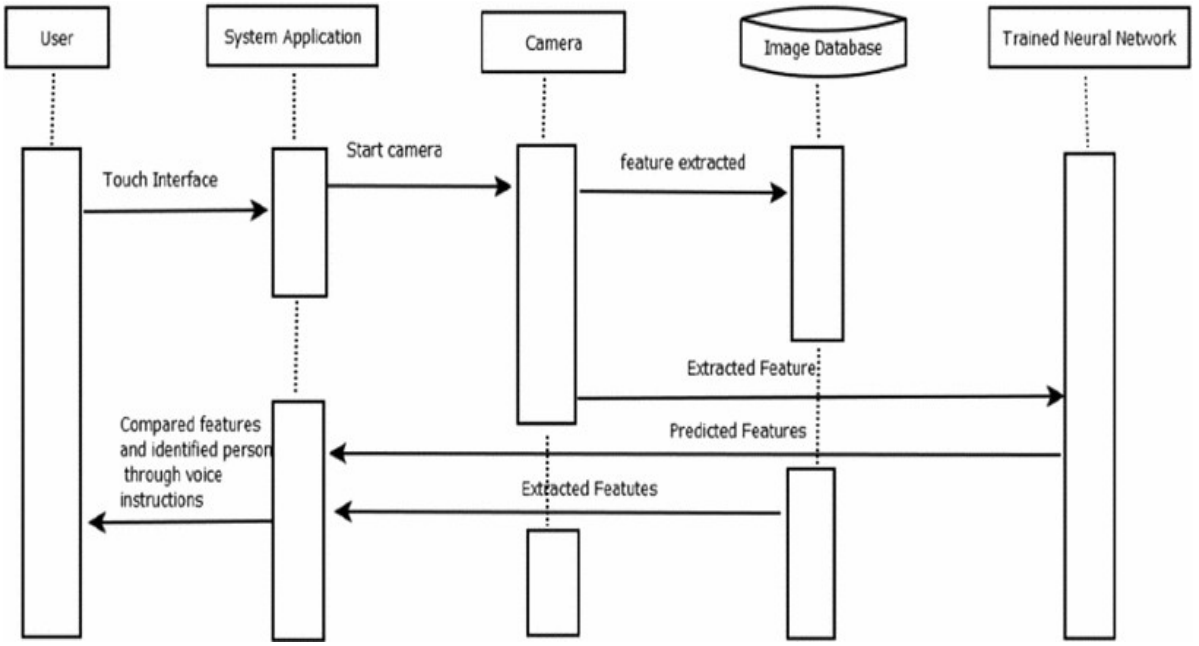


Fig: 4.2.4 Sequence Diagram

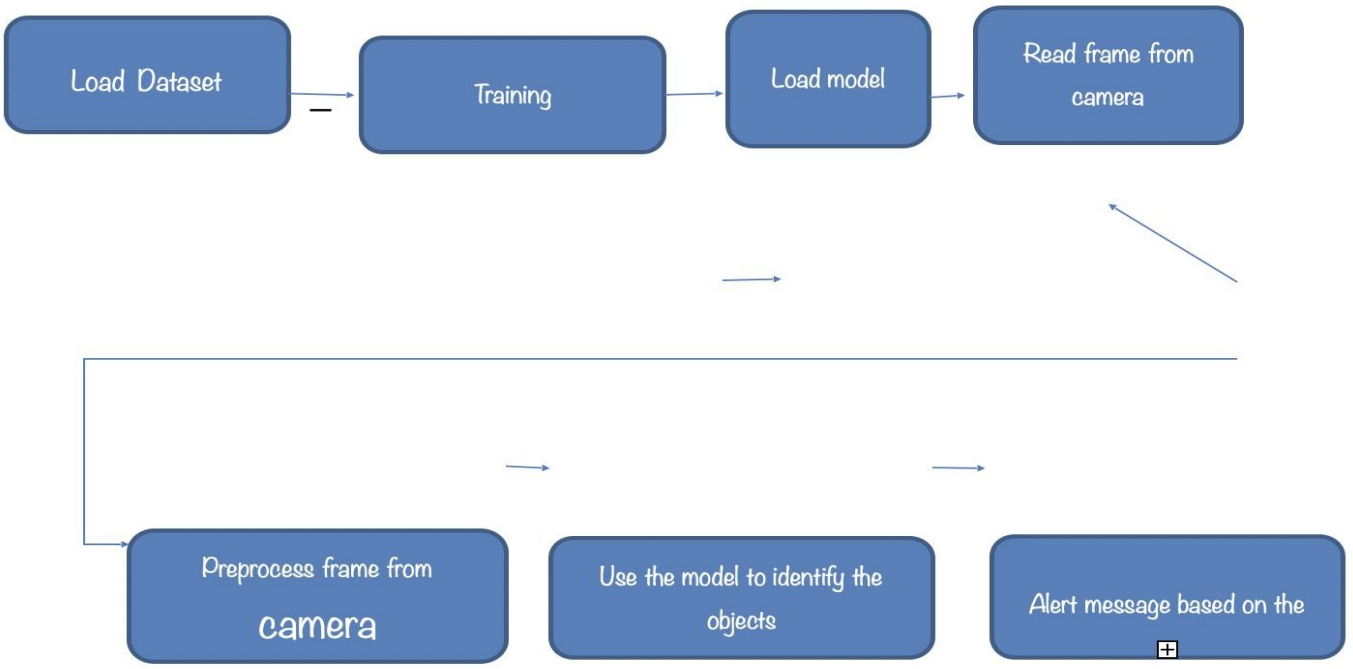
CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES OVERVIEW:

Module is a logical separation of functionality within a project. They are basically used for reusability and better code maintenance. There are four modules used here.

- Dataset Loading
- Training
- Load model
- Read Frame
- Apply the model for identification



Fig; 5.1 Module Diagram

5.2 MODULE DESCRIPTION:

5.2.1 DATASET LOADING

Data is the bread and butter of a Data Scientist, so knowing man approaches to loading data for analysis is crucial. Here, five Python techniques to bring in your data. We might only know a single way to load data (normally in CSV) which is to read it using pandas. read_csv function. It is one of the most mature and strong functions, but other ways are a lot helpful and will definitely come in handy sometimes.

They are:

- Manual function
- loadtxt function
- genfromtxtf unction
- read_csv function
- Pickle

Manual Function

This is the most difficult, as you have to design a custom function, which can load data for you. You have to deal with Python's normal filing concepts and using that you have to read a .csv file.

Numpy.loadtxt function

This is a built-in function in Numpy, a famous numerical library in Python. It is a really simple function to load the data. It is very useful for reading data which is of the same datatype.

When data is more complex, it is hard to read using this function, but when files are easy and simple, this function is really powerful.

```
array([[ 1.80000e+01,  6.20000e+01,  1.93200e+03],
       [ 2.30000e+01,  4.90000e+01, -4.66682e+05],
       [ 4.70000e+01,  5.20000e+01,  2.93158e+05],
       [ 5.10000e+01,  3.90000e+01, -3.27168e+05],
       [ 2.20000e+01,  3.80000e+01, -7.62248e+05]])
```

Fig: 5.2.1 Numpy. loadtxt function

5.2.2 TRAINING:

You should have 2 folders, one for the train set and the other for the test set.In the training set, you will have a .csv file and an image folder:

- The .csv file contains the names of all the training images and their corresponding true labels
- The image folder has all the training images.

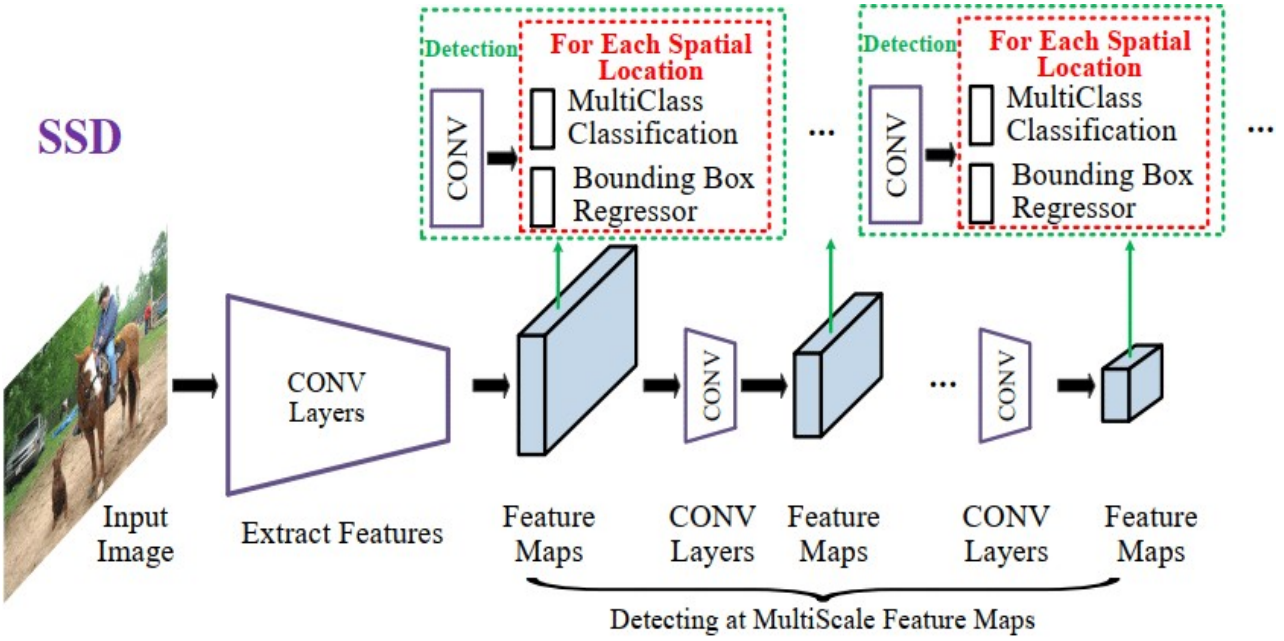
The .csv file in our test set is different from the one present in the training set. This test set .csv file contains the names of all the test images, but they do not have any corresponding labels. Can you guess why? Our model will be trained on the images present in the training set and the label predictions will happen on the testing set images

For training the model, we require:

- Training images and their corresponding true labels
- Validation images and their corresponding true labels (we use these labels only to validate the model and not during the training phase)

We also define the number of epochs in this step. For starters, we will run the model for 10 epochs (you can change the number of epochs later).

Time required for this step: Since training requires the model to learn structures, we need around.



5.2.2 IMAGE TRAINING USING MOBILE NET SSD

5.2.3 LOAD MODEL:

Intel Image classification dataset is already split into train, test, and Val, and we will only use the training dataset to learn how to load the dataset using different libraries.

Typical steps for loading custom dataset for Deep Learning Models

- **Open the image file**. The format of the file can be JPEG, PNG, BMP, etc.
- Resize the image to match the input size for the Input layer of the Deep Learning model.
- Convert the image pixels to float datatype.
- **Normalize the image** to have pixel values scaled down between 0 and 1 from 0 to 255.

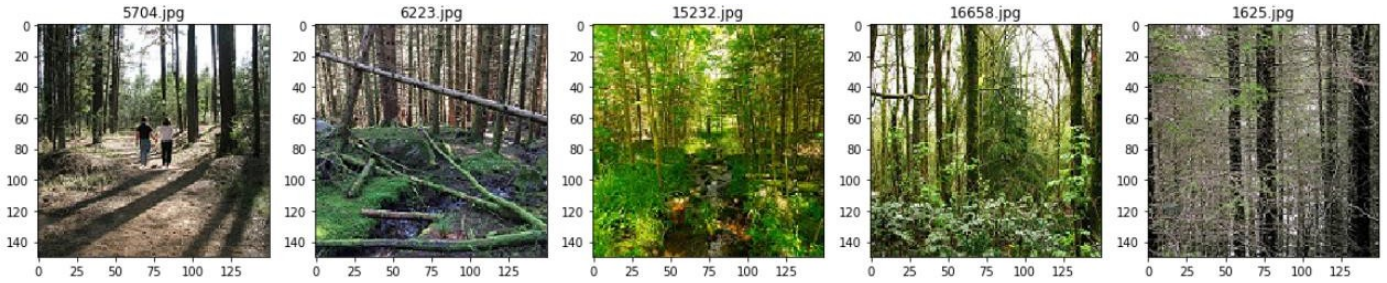


Fig: 5.2.3 Setting the Image dimension and source folder for loading the dataset

5.2.4 READ FRAME :

- Open the Video file or camera using **cv2.VideoCapture()**
- Read frame by frame
- Save each frame using **cv2.imwrite()**
- Release the VideoCapture and destroy all windows

If we don't have any video, no need to worry. Open the camera instead of the file using cv2.VideoCapture(0) and start extracting frames.

We can perform a number of operations on these frames like crop, flip, reverse etc. save them into a list and iterate over them to get cropped/flipped/reversed video.

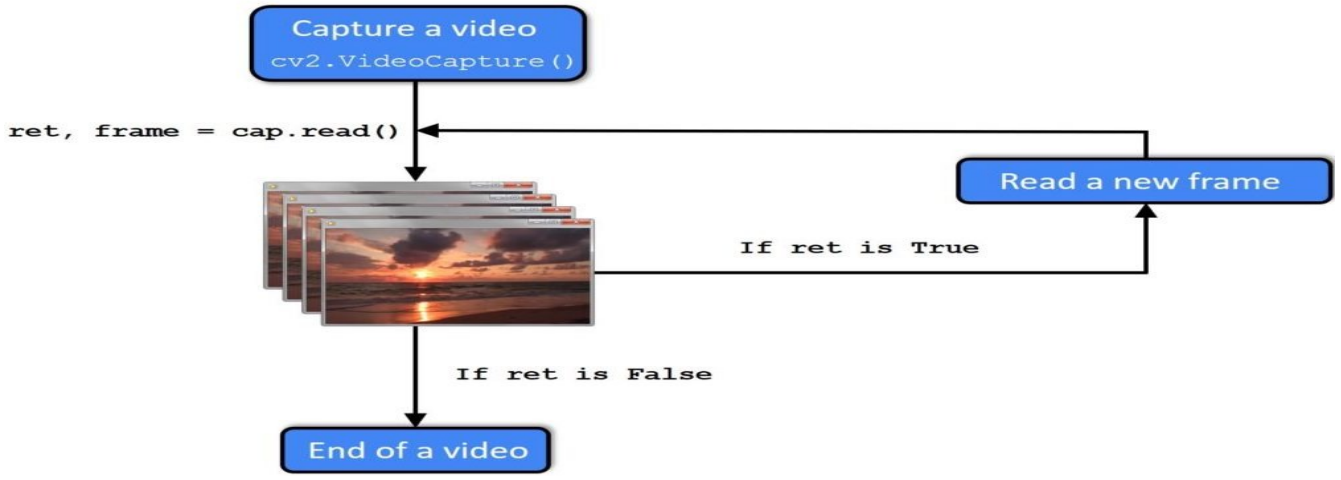


Fig: 5.2.4 CAPTURING FRAMES FROM VIDEO

5.2.5 APPLY THE MODEL FOR IDENTIFICATION

- **Image classification:** Algorithms produce a list of object categories present in the image.
- **Single-object localization:** Algorithms produce a list of object categories present in the image, along with an axis-aligned bounding box indicating the position and scale of one instance of each object category.
- **Object detection**: Algorithms produce a list of object categories present in the image along with an axis-aligned bounding box indicating the position and scale of every instance of each object category.

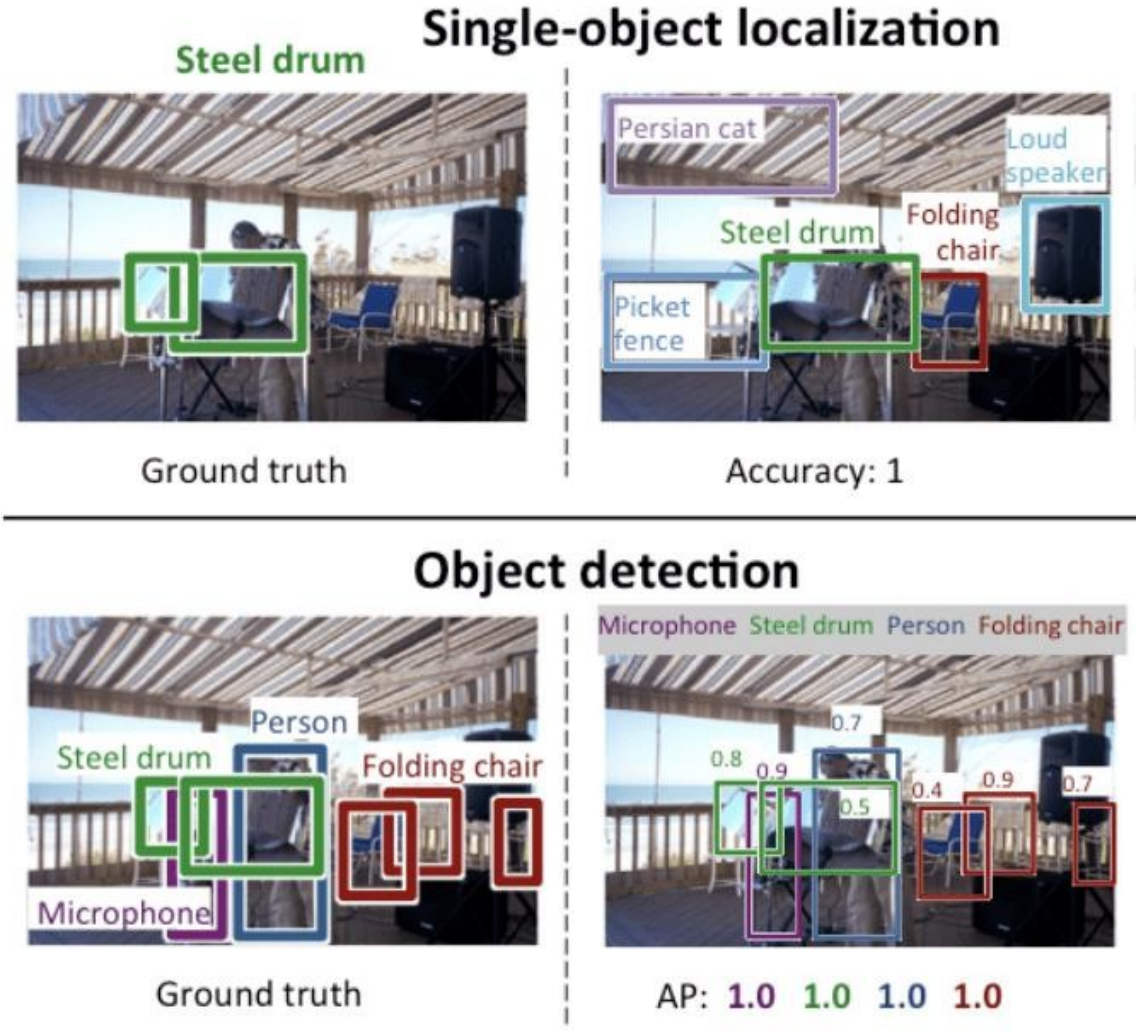


Fig: 5.2.5 OBJECT DETECTION

5.3 DIGITAL IMAGE PROCESSING:

Digital Image Processing means processing digital image by means of a digital computer. We can also say that it is a use of computer algorithms, in order to get enhanced image either to extract some useful information. The identification of objects in an image would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures. The clever bit is to interpret collections of these shapes as single objects. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skillful programming and lots of processing power to approach human performance. Manipulating data in the form of an image through several possible techniques. An image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value

of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

These operations, taken singly or in combination, are the means by which the image is enhanced, restored, or compressed. An image is enhanced when it is modified so that the information it contains is more clearly evident, but enhancement can also include making the image more visually appealing.

User interface:

Detecting Graphical consumer Interface (GUI) factors in GUI photos is a website-specific item detection challenge. It supports many software program engineering obligations, along with GUI animation and testing, GUI seek and code technology. present studies for GUI element detection at once borrow the mature methods from pc imaginative and prescient (CV) area, which includes old school ones that depend on traditional image processing capabilities (e.g., canny edge, contours), and deep mastering fashions that learn how to detect from huge-scale GUI information. lamentably, those CV methods aren't in the beginning designed with the awareness of the unique traits of GUIs and GUI factors and the excessive localization accuracy of the GUI element detection project. We behaviour the primary huge-scale empirical observe of 7 consultant GUI detail detection methods on over 50k GUI snap shots to understand the abilities, limitations and powerful designs of these techniques. This examine now not best sheds the light on the technical challenges to be addressed but additionally informs the design of latest GUI element detection methods. We as a consequence layout a new GUI-particular old-fashioned technique for non-textual content GUI detail detection which adopts a singular top-down coarse-to-high-quality method, and include it with the mature deep learning model for GUI text detection. Our assessment on 25,000 GUI photographs suggests that our method extensively advances the begin-of-the-art overall performance in GUI element detection.

CHAPTER 6

TESTING

6.1 OVERVIEW ABOUT TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risk of software implementation. Test techniques include the process executing the program or application with the intent of finding software bugs(errors or other defects), and verifying that the software product is fit for use.

6.2 TYPES OF SOFTWARE TESTING:

6.2.1 White box testing

6.2.2 Black box testing

6.2.3 Unit Testing

6.2.4 Functional Testing

6.2.5 Performance Testing

6.2.6 Integration Testing

6.2.7 Validation Testing

6.2.8 System Testing

6.2.9 Structure Testing

6.2.10 Output Testing

6.2.11 User Acceptance Testing

6.2.1 WHITE BOX TESTING

White Box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white box testing an internal perspective of the system as well as programming skills are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. White box testing can be applied at the unit, integration and system levels of the testing process. Although traditional testers tended to think of white box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration and between subsystems during a system level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

6.2.2 BLACK BOX TESTING

Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing like unit, integration, software, system and acceptance. It is sometimes referred to as specification based testing. Black box testing, also known as behavioural testing is a software testing method in which the internal structure design implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

6.2.3 UNIT TESTING

- Unit testing, also known as Module testing, focuses verification efforts on the module. The module is tested separately and this is carried out at the programming stage itself.
- Unit test comprises of the set of tests performed by an individual programmer before integration of the unit into the system.
- Unit test focuses on the smallest unit of software design the software component or module.
- Using component level design, important control paths are tested to uncover errors within the boundary of the module.
- Unit test is white box oriented and the step can be conducted in parallel for multiple components.

6.2.4 FUNCTIONAL TESTING

Functional testing is a type of software testing whereby the system is tested against the functional requirements specifications. Functions or features are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs but rather with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black box testing technique is used in which the internal logic of the system being tested is not known to the tester. Functional testing is normally performed during the levels of system testing and acceptance testing.

Typically Functional testing involves the following steps:

Identify functions that the software is expected to perform.

Create input data based on the function’s specification.

Determine the output based on the function's specification.

Execute the test case.

Compare the actual and expected outputs.

6.2.5 PERFORMANCE TESTING

Performance testing is done to provide stakeholders with information about their application regarding speed, stability and scalability. More importantly, Performance testing uncovers what needs to be improved before the product goes to market. Without performance testing, software is likely to suffer from issues such as: running slow while several users use it simultaneously, inconsistencies across different operating systems and poor usability. Performance testing determines the amount of execution tine spent in various parts of the unit, program throughput and response time and device utilization of the program unit. It occurs throughout all steps in the testing process.

6.2.6 INTEGRATION TESTING

- It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface.
- It takes the unit tested modules and builds a program structure.
- All the modules are combined and tested as a whole.
- Integration of all the components to form the entire system and a overall testing is executed.

6.2.7 VALIDATION TESTING

- Validation test succeeds when the software functions in a manner that can be reasonably by the client.
- Software validation is achieved through a series of black box testing which confirms to the requirements. The test is designed to uncover interface errors, is also used to demonstrate that software functions are operational, input is properly accepted, output are produced and the integrity of external information is maintained.

6.2.8 SYSTEM TESTING

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. Tests to find the discrepancies between the system and its original objective, current specification and system documentation.

6.2.9 STRUCTURAL TESTING

Structural testing is the type of testing carried out to test the structure of code. It is also known as testing or Glass box testing. This type of testing requires knowledge of the code, so it is mostly done by the developers. It is more concerned with how system does it rather than the functionality of the system. It provides more coverage to the testing. For example, to test certain error message in application, we need to test the trigger condition for it, but there must be many trigger for it. It is possible to miss out one while testing the requirements drafted in SRS. But using this testing, the trigger is most likely to be covered since structural testing aims to cover all the nodes and paths in the

structure of code. It is concerned with exercising the internal logic of a program and traversing particular execution paths.

6.2.10 OUTPUT TESTING

- Output of test cases compared with the expected results created during design of test cases.
- Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
- Here, the output format is considered into two was, one is on screen and another one is printed format.
- The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.
- The output comes out as the specified requirements as the user's hard copy.

6.2.11 USER ACCEPTANCE TESTING

- Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
- The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.

It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.

Two set of acceptance test to be run:

1. Those developed by quality assurance group.
2. Those developed by customer.

CHAPTER 7

SCREENSHOTS

Fig 7.1 Reading frame

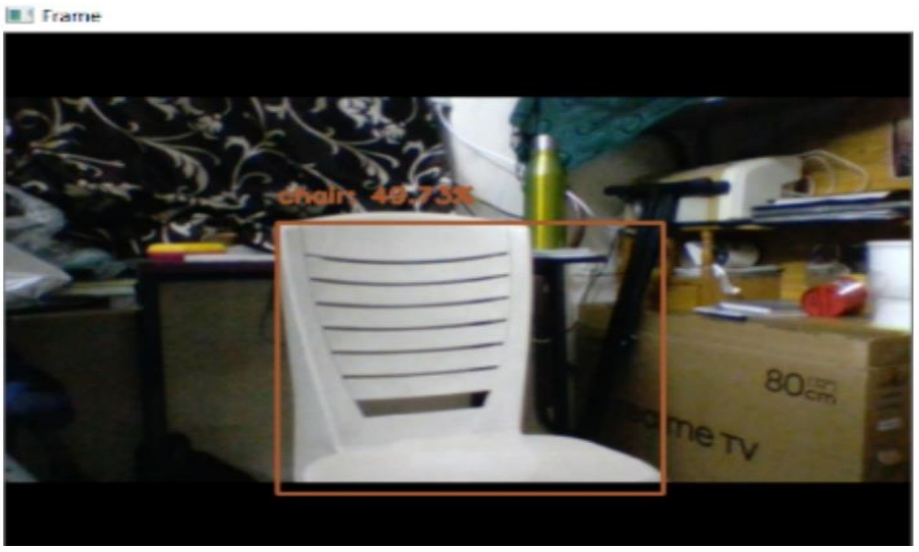


Fig 7.2 Detecting the objects

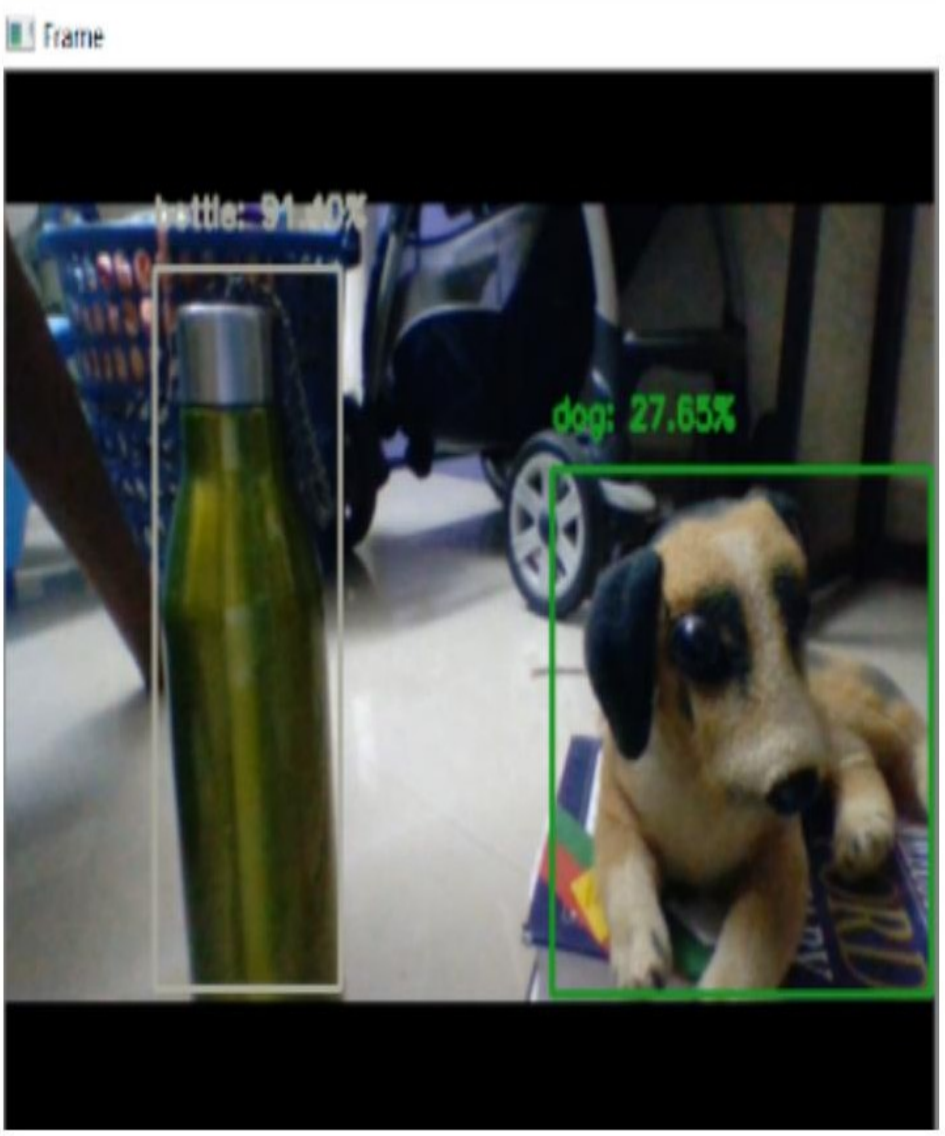


Fig 7.3 Displaying the objects

CHAPTER 8

CONCLUSION

This project involves the detection of both human and objects that are being displayed in the video and the objects are pre-trained using mobile net SSD. Then the identification of different objects are done by using the fames obtained from the video. This kind of object recognition in video surveillance systems is used in many applications.

FUTURE ENHANCEMENT

Object recognition system can be applied in the area of surveillance system, face recognition, fault detection, character recognition etc. The objective of this thesis is to develop an object recognition system to recognize the 2D and 3D objects in the image. The performance of the object recognition system depends on the features used and the classifier employed for recognition. This research work attempts to propose a novel feature extraction method for extracting global features and obtaining local features from the region of interest. Also the research work attempts to hybrid the traditional classifiers to recognize the object. The object recognition system developed in this research was tested with the benchmark datasets like COIL100, Caltech 101, ETH80 and MNIST. The object recognition system is implemented in MATLAB 7.5It is important to mention the difficulties observed during the experimentation of the object recognition system due to several features present in the image. The research work suggests that the image is to be preprocessed and reduced to a size of 128 x 128. The proposed feature extraction method helps to select the important feature. To improve the efficiency of the classifier, the number of features should be less in number.

CHAPTER 9

REFERENCES:

[1] Implementation of an Automated Single Camera Object Tracking System Using Frame Differencing and Dynamic Template Matching'Karan Gupta1, Anjali V. Kulkarni2 Indian Institute of Technology, Kanpur, India

[2] Image Processing Procedures for the Thermal Measurements Vladimir Sz´ekely and M´artaRencz, Member, IEEE.

[3] Practical and Advanced Image Processing for Security and Recognition by Thermal Distributed Image Features Osamu Ono Dept. of Electronics & Bioinformatics, School of Science and Engineering.

[4] Vision based moving object detection and tracking 1 Kalpesh R Jadav, 2Prof.M.A.Lokhandwala,3Prof.A.P.Gharge EC Dept, GTU University Parul Institute Engg& Tech Limda,vadodara,India.

[5] Shadow Detection And Removal In Colour Images Using Matlab' Sanjeev Kumar et. al. / International Journal of Engineering Science and Technology

[6] Detection of Foreign Bodies in Food by Thermal Image Processing GiaimeGinesu, Student Member, IEEE, Daniele D. Giusto, Senior Member, IEEE, Volker Märgner, Member, IEEE, and Peter Meinlschmidt.

CHAPTER 10

APPENDIX

APPENDIX A - SOURCE CODE:

```
import numpy as np
import imutils
import cv2
import time

prototxt = "MobileNetSSD_deploy.prototxt.txt"
model = "MobileNetSSD_deploy.caffemodel"
confThresh = 0.2

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

print("Loading model...")
net = cv2.dnn.readNetFromCaffe(prototxt, model)
print("Model Loaded")
print("Starting Camera Feed...")
```

```
vs = cv2.VideoCapture(0)
time.sleep(2.0)

while True:
    _,frame = vs.read()
    frame = imutils.resize(frame, width=500)

    (h, w) = frame.shape[:2]
    imResizeBlob = cv2.resize(frame, (300, 300))
    blob = cv2.dnn.blobFromImage(imResizeBlob,
        0.007843, (300, 300), 127.5)

    net.setInput(blob)
    detections = net.forward()
    detShape = detections.shape[2]
    for i in np.arange(0,detShape):
        confidence = detections[0, 0, i, 2]
        if confidence > confThresh:
            idx = int(detections[0, 0, i, 1])
            print("ClassID:",detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            label = "{}: {:.2f}%".format(CLASSES[idx],
                confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                COLORS[idx], 2)
            if startY - 15 > 15:
```

```
            else:
                startY + 15
                cv2.putText(frame, label, (startX, startY),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
         cv2.imshow("Frame", frame)
        key = cv2.waitKey(1)
        if key == 27:
              break
vs.release()
cv2.destroyAllWindows()
```

Detecting and Recognizing Human-Object Interactions

Georgia Gkioxari Ross Girshick Piotr Dollár Kaiming He
Facebook AI Research (FAIR)

Abstract

To understand the visual world, a machine must not only recognize individual object instances but also how they interact. *Humans are often at the center of such interactions and detecting human-object interactions is an important practical and scientific problem. In this paper, we address the task of detecting* ⟨`human`, `verb`, `object`⟩ *triplets in challenging everyday photos. We propose a novel model that is driven by a* human-centric *approach. Our hypothesis is that the appearance of a person – their pose, clothing, action – is a powerful cue for localizing the objects they are interacting with. To exploit this cue, our model learns to predict an action-specific density over target object locations based on the appearance of a detected person. Our model also jointly learns to detect people and objects, and by fusing these predictions it efficiently infers interaction triplets in a clean, jointly trained end-to-end system we call InteractNet. We validate our approach on the recently introduced Verbs in COCO (V-COCO) and HICO-DET datasets, where we show quantitatively compelling results.*

1. Introduction

Visual recognition of individual instances, *e.g.*, detecting objects [10, 9, 27] and estimating human actions/poses [12, 32, 2], has witnessed significant improvements thanks to deep learning visual representations [18, 30, 31, 17]. However, recognizing individual objects is just a first step for machines to comprehend the visual world. To understand what is *happening* in images, it is necessary to also recognize relationships between individual instances. In this work, we focus on *human-object interactions*.

The task of recognizing human-object interactions [13, 33, 6, 14, 5] can be represented as detecting ⟨`human`, `verb`, `object`⟩ triplets and is of particular interest in applications and in research. From a practical perspective, photos containing people contribute a considerable portion of daily uploads to internet and social networking sites, and thus human-centric understanding has significant demand in practice. From a research perspective, the person category involves a rich set of actions/verbs, most of which are rarely taken by other subjects (*e.g.*, *to talk*, *throw*, *work*). The fine granularity of human actions and their interactions with a wide array of object types presents a new challenge compared to recognition of entry-level object categories.

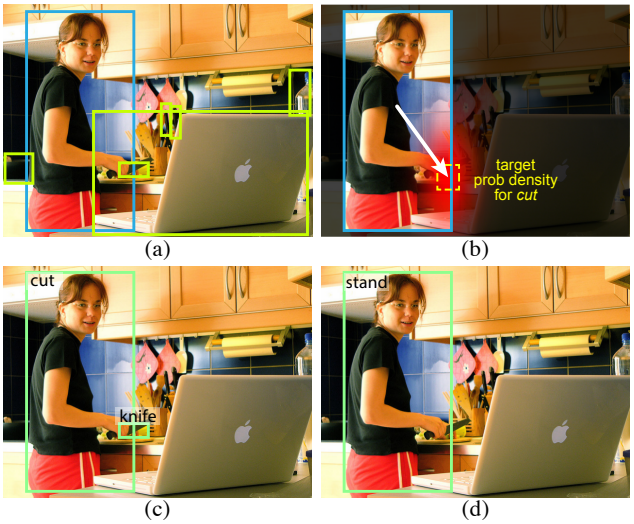


Figure 1. **Detecting and recognizing human-object interactions. (a)** There can be many possible objects (green boxes) interacting with a detected person (blue box). **(b)** Our method estimates an action-type specific *density* over target object locations from the person's appearance, which is represented by features extracted from the detected person's box. **(c)** A ⟨`human`, `verb`, `object`⟩ triplet detected by our method, showing the person box, action (*cut*), and target object box and category (*knife*). **(d)** Another predicted action (*stand*), noting that a person can simultaneously take multiple actions and an action may not involve any objects.

In this paper, we present a human-centric model for recognizing human-object interaction. Our central observation is that a person's appearance, which reveals their action and pose, is highly informative for inferring where the target object of the interaction may be located (Figure 1(b)). The search space for the target object can thus be narrowed by conditioning on this estimation. Although there are often many objects detected (Figure 1(a)), the inferred target location can help the model to quickly pick the correct object associated with a specific action (Figure 1(c)).

We implement this idea as a *human-centric* recognition

Figure 2. Human-object interactions detected by our method. Each image shows one detected ⟨`human`, `verb`, `object`⟩ triplet.

branch in the Faster R-CNN framework [27]. Specifically, on a region of interest (RoI) associated with a person, this branch performs *action* classification and *density estimation* for the action's target object location. The density estimator predicts a 4-d Gaussian distribution, for each action type, that models the likely *relative* position of the target object to the person. The prediction is based purely on the human appearance. This human-centric recognition branch, along with a standard object detection branch [9] and a simple pairwise interaction branch (described later), form a multi-task learning system that can be jointly optimized.

We evaluate our method, *InteractNet*, on the challenging V-COCO (*Verbs in COCO*) dataset [14] for detecting human-object interactions. Our human-centric model improves accuracy by 26% (relative) from 31.8 to 40.0 AP (evaluated by Average Precision on a triplet, called 'role AP' [14]), with the gain mainly due to inferring the target object's relative position from the human appearance. In addition, we prove the effectiveness of InteractNet by reporting a 27% relative improvement on the newly released HICO-DET dataset [3]. Finally, our method can run at about 135ms / image for this complex task, showing good potential for practical usage.

2. Related Work

Object Detection. Bounding-box based object detectors have improved steadily in the past few years. R-CNN, a particularly successful family of methods [10, 9, 27], is a two-stage approach in which the first stage proposes candidate RoIs and the second stage performs object classification. Region-wise features can be rapidly extracted [16, 9] from shared feature maps by an RoI pooling operation. Feature sharing speeds up instance-level detection and enables recognizing higher-order interactions, which would be computationally infeasible otherwise. Our method is based on the Fast/Faster R-CNN frameworks [9, 27].

Human Action & Pose Recognition. The action and pose of humans is indicative of their interactions with objects or other people in the scene. There has been great progress in understanding human actions [12] and poses [32, 2, 15] from images. These methods focus on the human instances and do not predict interactions with other objects. We rely on action and pose appearance cues in order to predict the interactions with objects in the scene.

Visual Relationships. Research on visual relationship modeling [29, 14, 23, 34] has attracted increasing attention. Recently, Lu *et al*. [23] proposed to recognize visual relationships derived from an open-world vocabulary. The set of relationships include verbs (*e.g*., *wear*), spatial (*e.g*., *next to*), actions (*e.g*., *ride*) or a preposition phrase (*e.g*., *drive on*). Our focus is related, but different. First, we aim to understand *human-centric* interactions, which take place in particularly diverse and interesting ways. These relationships involve direct interaction with objects (*e.g*., *person cutting cake*), unlike spatial or prepositional phrases (*e.g*., *dog next to dog*). Second, we aim to build detectors that recognize interactions in images with high precision, which is a requirement for practical applications. In contrast, in an open-world recognition setting, evaluating precision is not feasible, resulting in recall-based evaluation, as in [23].

Human-Object Interactions. Human-object interactions [13, 33, 6] are related to visual relationships, but present different challenges. Human actions are more fine-grained (*e.g*., *walking*, *running*, *surfing*, *snowboarding*) than the actions of general subjects, and an individual person can simultaneously take multiple actions (*e.g*., *drinking tea* and *reading a newspaper* while *sitting in a chair*). These issues require a deeper understanding of human actions and the objects around them and in much richer ways than just the presence of the objects in the vicinity of a person in an image. Accurate recognition of human-object interaction can benefit numerous tasks in computer vision, such as action-specific image retrieval [26], caption generation [35], and question answering [35, 24].

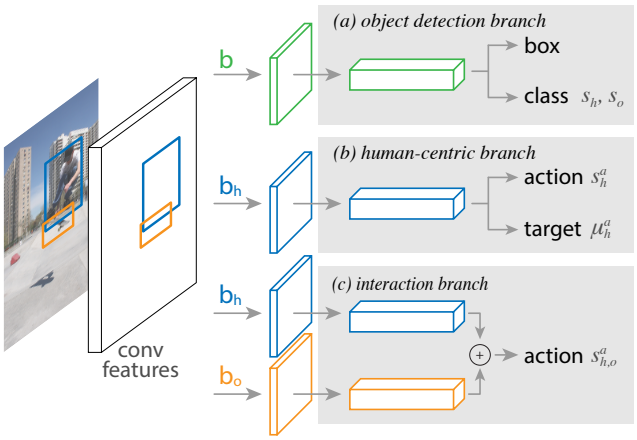


Figure 3. **Model Architecture.** Our model consists of (a) an *object detection* branch, (b) a *human-centric* branch, and (c) an optional *interaction* branch. The person features and their layers are shared between the human-centric and interaction branches (blue boxes).

3. Method

We now describe our method for detecting human-object interactions. Our goal is to detect and recognize triplets of the form ⟨`human`, `verb`, `object`⟩. To detect an interaction triplet, we have to accurately localize the box containing a `human` and the box for the associated `object` of interaction (denoted by b_h and b_o, respectively), as well as identify the action a being performed (selected from among A actions).

Our proposed solution decomposes this complex and multifaceted problem into a simple and manageable form. We extend the Fast R-CNN [9] object detection framework with an additional *human-centric* branch that classifies actions and estimates a probability density over the *target* object location for each action. The human-centric branch reuses features extracted by Fast R-CNN for object detection so its marginal computation is lightweight.

Specifically, given a set of candidate boxes, Fast R-CNN outputs a set of object boxes and a class label for each box. Our model extends this by assigning a triplet score $S^a_{h,o}$ to pairs of candidate human/object boxes b_h, b_o and an action a. To do so, we decompose the triplet score into four terms:

$$S^a_{h,o} = s_h \cdot s_o \cdot s^a_h \cdot g^a_{h,o} \quad (1)$$

While the model has multiple components, the basic idea is straightforward. s_h and s_o are the class scores from Fast R-CNN of b_h and b_o containing a human and object. Our human-centric branch outputs two extra terms. First, s^a_h is the score assigned to action a for the person at b_h. Second, μ^a_h is the predicted location of the target of interaction for a given human/action pair, computed based on the appearance of the human. This, in turn, is used to compute $g^a_{h,o}$, the likelihood that an object with box b_o is the actual target of interaction. We give details shortly and show that this *target localization* term is key for obtaining good results.

We discuss each component next, followed by an extension that replaces the action classification output s^a_h with a dedicated *interaction* branch that outputs a score $s^a_{h,o}$ for an action a based on both the human and object appearances. Finally we give details for training and inference. Figure 3 illustrates each component in our full framework.

3.1. Model Components

Object Detection. The object detection branch of our network, shown in Figure 3(a), is identical to that of Faster R-CNN [27]. First, a Region Proposal Network (RPN) is used to generate object proposals [27]. Then, for each proposal box b, we extract features with RoiAlign [15], and perform object classification and bounding-box regression to obtain a new set of boxes, each of which has an associated score s_o (or s_h if the box is assigned to the person category). These new boxes are only used during inference; during training all branches are trained with RPN proposal boxes.

Action Classification. The first role of the *human-centric* branch is to assign an action classification score s^a_h to each human box b_h and action a. Just like in the object classification branch, we extract features from b_h with RoiAlign and predict a score for each action a. Since a human can simultaneously perform multiple actions (*e.g.*, *sit* and *drink*), our output layer consists of *binary* sigmoid classifiers for *multi-label* action classification (*i.e.* the predicted action classes do not compete). The training objective is to minimize the binary cross entropy losses between the ground-truth action labels and the scores s^a_h predicted by the model.

Target Localization. The second role of the *human-centric* branch is to predict the target object location based on a person's appearance (again represented as features pooled from b_h). However, predicting the *precise* target object location based only on features from b_h is challenging. Instead, our approach is to predict a *density* over possible locations, and use this output together with the location of actual detected objects to precisely localize the target.

We model the density over the target object's location as a Gaussian function whose mean is predicted based on the human appearance and action being performed. Formally, the human-centric branch predicts μ^a_h, the target object's 4-d mean location given the human box b_h and action a. We then write our target localization term as:

$$g^a_{h,o} = \exp(\|b_{o|h} - \mu^a_h\|^2 / 2\sigma^2) \quad (2)$$

We can use g to test the compatibility of an object box b_o and the predicted target location μ^a_h. In the above, $b_{o|h}$ is the encoding of b_o in coordinates relative to b_h, that is:

$$b_{o|h} = \{\frac{x_o - x_h}{w_h}, \frac{y_o - y_h}{h_h}, \log\frac{w_o}{w_h}, \log\frac{h_o}{h_h}\} \quad (3)$$

This is a similar encoding as used in Fast R-CNN [9] for bounding box regression. However, in our case b_h and b_o

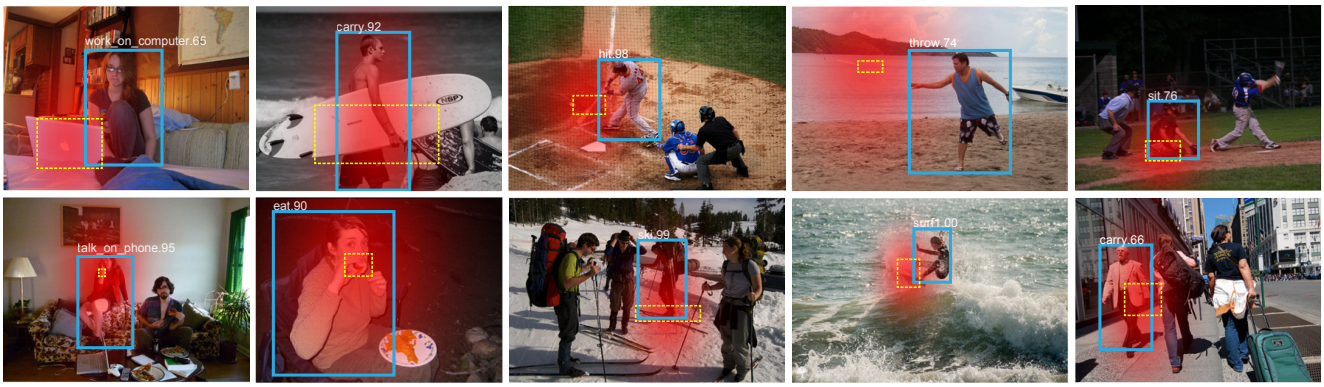


Figure 4. **Estimating target object density from the person features**. We estimate a 4-d Gaussian density whose mean μ_h^a represents a 4-d offset for the target object of action a (illustrated as yellow boxes); the variance of the density is illustrated in red for the 2-d translation offsets of (x, y) (the scaling offsets' variance is not visualized). These target locations will be combined with the object detections b_o to detect human-object interaction triplets. This figure also shows the predicted actions and their scores from the person RoIs. **The rightmost column** shows two intriguing examples: even though there are no target objects, our model predicts reasonably densities from the human pose (these predictions will be rejected by the object detection module, which will not find an object in the high density regions).

are *two different objects* and moreover b_o is not necessarily near or of the same size as b_h. The training objective is to minimize the smooth L_1 loss [9] between μ_h^a and $b_{o|h}$, where b_o is the location of the ground truth object for the interaction. We treat σ as a hyperparameter that we empirically set to $\sigma = 0.3$ using the validation set.

Figure 4 visualizes the predicted distribution over the target object's location for example human/action pairs. As we can see, a *carrying* appearance suggests an object in the person's hand, a *throwing* appearance suggests an object in front of the person, and a *sitting* appearance implies an object below the person. We note that the yellow dashed boxes depicting μ_h^a shown in Figure 4 are inferred from b_h and a and did not have direct access to the objects.

Intuitively, our formulation is predicated on the hypothesis that the features computed from b_h contain a strong signal pointing to the target of an action, even if that target object is outside of b_h. We argue that such 'outside-the-box' regression is possible because the person's appearance provides a strong clue for the target location. Moreover, as this prediction is action-specific and instance-specific, our formulation is effective even though we model the target location using a uni-modal distribution. In Section 5 we discuss a variant of our approach which allows us to handle conditionally multi-modal distributions and predict multiple targets for a single action.

Interaction Recognition. Our human-centric model scores actions based on the human appearance. While effective, this does not take into account the appearance of the target object. To improve the discriminative power of our model, and to demonstrate the flexibility of our framework, we can replace s_h^a in (1) with an *interaction* branch that scores an action based on the the appearance of both the human and target object. We use $s_{h,o}^a$ to denote this alternative term.

The computation of $s_{h,o}^a$ reuses the computation from s_h^a and additionally in parallel performs a similar computation based on features extracted from b_o. The outputs from the two action classification heads, which are A-dimensional vectors of logits, are summed and passed through a sigmoid activation to yield A scores. This process is illustrated in Figure 3(c). As before, the training objective is to minimize the binary cross entropy losses between the ground-truth action labels and the predicted action scores $s_{h,o}^a$.

3.2. Multi-task Training

We approach learning human-object interaction as a *multi-task* learning problem: all three branches shown in Figure 3 are trained jointly. Our overall loss is the sum of all losses in our model including: (1) the classification and regression loss for the object detection branch, (2) the action classification and target localization loss for the human-centric branch, and (3) the action classification loss of the interaction branch. This is in contrast to our cascaded inference described in §3.3, where the output of the object detection branch is used as input for the human-centric branch.

We adopt image-centric training [9]. All losses are computed over both RPN proposal and ground truth boxes as in Faster R-CNN [27]. As in [9], we sample at most 64 boxes from each image for the object detection branch, with a ratio of 1:3 of positive to negative boxes. The human-centric branch is computed over at most 16 boxes b_h that are associated with the human category (*i.e.*, their IoU overlap with a ground-truth person box is ≥ 0.5). The loss for the interaction branch is only computed on positive example triplets (*i.e.*, $\langle b_h, a, b_o \rangle$ must be associated with a ground truth interaction triplet). All loss terms have a weight of one, except the action classification term in the human-centric branch has a weight of two, which we found performs better.

3.3. Cascaded Inference

At inference, our goal is to find high-scoring triplets according to $S^a_{h,o}$ in (1). While in principle this has $O(n^2)$ complexity as it requires scoring every pair of candidate boxes, we present a simple cascaded inference algorithm whose dominant computation has $O(n)$ complexity.

Object Detection Branch: We first detect all objects (including the *person* class) in the image. We apply non-maximum suppression (NMS) with an IoU threshold of 0.3 [9] on boxes with scores higher than 0.05 (set conservatively to retain most objects). This step yields a new *smaller* set of n boxes b with scores s_h and s_o. Unlike in training, these new boxes are used as input to the remaining two branches.

Human-Centric Branch: Next, we apply the human-centric branch to all detected objects that were classified as *human*. For each action a and detected human box b_h, we compute s^a_h, the score assigned to a, as well as μ^a_h, the predicted mean offset of the target object location relative to b_h. This step has a complexity of $O(n)$.

Interaction Branch: If using the optional interaction branch, we must compute $s^a_{h,o}$ for each action a and pair of boxes b_h and b_o. To do so we first compute the logits for the two action classification heads independently for each box b_h and b_o, which is $O(n)$. Then, to get scores $s^a_{h,o}$, these logits are summed and passed through a sigmoid for each pair. Although this last step is $O(n^2)$, in practice its computational time is negligible.

Once all individual terms have been computed, the computation of (1) is fast. However, rather than scoring every potential triplet, for each human/action pair we find the object box that maximizes $S^a_{h,o}$. That is we compute:

$$b_{o^*} = \arg\max_{b_o} \; s_o \cdot s^a_{h,o} \cdot g^a_{h,o} \tag{4}$$

Recall that $g^a_{h,o}$ is computed according to (2) and measures the compatibility between b_o and the expected target location μ^a_h. Intuitively, (4) encourages selecting a high-confidence object near the predicted target location of a high-scoring action. With b_o selected for each b_h and action a, we have a triplet of $\langle$`human`, `verb`, `object`$\rangle$ = $\langle b_h, a, b_o\rangle$. These triplets, along with the scores $S^a_{h,o}$, are the final outputs of our model. For actions that that do not interact with any object (*e.g.*, *smile*, *run*), we rely on s^a_h and the interaction output $s^a_{h,o}$ is not used, even if present. The score of such a predicted $\langle$`human`, `verb`$\rangle$ pair is simply $s_h \cdot s^a_h$.

The above cascaded inference has a dominant complexity of $O(n)$, which involves extracting features for each of the n boxes and forwarding through a small network. The pairwise $O(n^2)$ operations require negligible computation. In addition, for the entire system, a portion of computation is spent on computing the full-image shared convolutional feature maps. Altogether, our system takes ~135ms on a typical image running on a single Nvidia M40 GPU.

4. Datasets and Metrics

There exist a number of datasets for human-object interactions [19, 4, 28, 14, 3]. The most relevant for this work are V-COCO (*Verbs in COCO*) [14] and HICO-DET [3]. V-COCO serves as the primary testbed on which we demonstrate the effectiveness of InteractNet and analyze its various components. The newly released HICO-DET [3] contains ~48k images and 600 types of interactions and serves to further demonstrate the efficacy of our approach. The older TUHOI [19] and HICO [4] datasets only have image-level labels and thus do not allow for grounding interactions in a *detection* setting, while COCO-a [28] is promising but only a small beta-version is currently available.

V-COCO is a subset of COCO [22] and has ~5k images in the `trainval` set and ~5k images in the `test` set.[1] The `trainval` set includes ~8k person instances and on average 2.9 actions/person. V-COCO is annotated with 26 common action classes (listed in Table 2). Of note, there are three actions (*cut*, *hit*, *eat*) that are annotated with two types of targets: *instrument* and *direct object*. For example, *cut* + *knife* involves the instrument (meaning 'cut with a knife'), and *cut* + *cake* involves the direct object (meaning 'cut a cake'). In [14], accuracy is evaluated separately for the two types of targets. To address this, for the target estimation, we train and infer two types of targets for these three actions (*i.e.*, they are treated like six actions for target estimation).

Following [14], we evaluate two Average Precision (AP) metrics. We note that this is a *detection* task, and both AP metrics measure *both* recall and precision. This is in contrast to metrics of Recall@N that ignore *precision*.

The AP of central interest in the *human-object interaction* task is the AP of the *triplet* $\langle$`human`, `verb`, `object`$\rangle$, called '*role* AP' (AP_{role}) in [14]. Formally, a triplet is considered as a *true positive* if: **(i)** the predicted human box b_h has IoU of 0.5 or higher with the ground-truth human box, **(ii)** the predicted object box b_o has IoU of 0.5 or higher with the ground-truth target object, *and* **(iii)** the predicted and ground-truth actions match. With this definition of a true positive, the computation of AP is analogous to standard object detection (*e.g.*, PASCAL [8]). Note that this metric does not consider the correctness of the target object category (but only the target object box location). Nevertheless, our method can predict the object categories, as shown in the visualized results (Figure 2 and Figure 5).

We also evaluate the AP of the pair $\langle$`human`, `verb`$\rangle$, called '*agent* AP' (AP_{agent}) in [14], computed using the above criteria of (i) and (iii). AP_{agent} is applicable when the action has no object. We note that AP_{agent} does note require localizing the target, and is thus of secondary interest.

[1] V-COCO's `trainval` set is a subset of COCO's `train` set, and its `test` set is a subset of COCO's `val` set. See [14] for more details. In this work, COCO's `val` images are not used during training in any way.

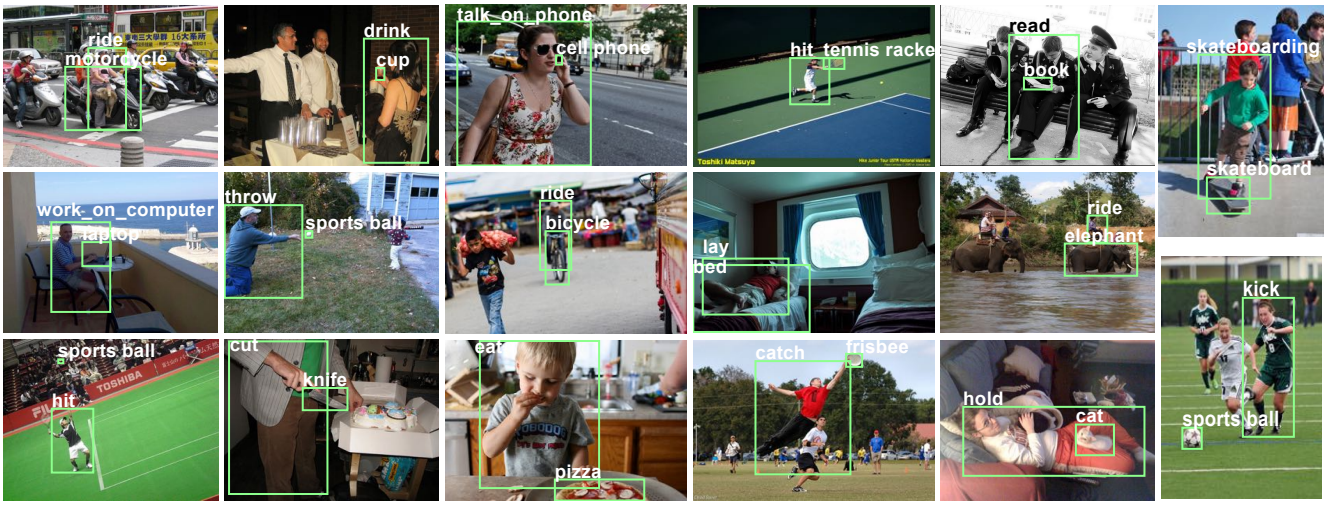


Figure 5. Our results on some V-COCO test images. Each image shows one detected ⟨`human`, `verb`, `object`⟩ triplet.

5. Experiments

Implementation Details. Our implementation is based on Faster R-CNN [27] with a Feature Pyramid Network (FPN) [21] backbone built on ResNet-50 [17]; we also evaluate a non-FPN version in ablation experiments. We train the Region Proposal Network (RPN) [27] of Faster R-CNN following [21]. For convenient ablation, RPN is frozen and does not share features with our network (we note that feature sharing is possible [27]). We extract 7×7 features from regions by RoiAlign [15], and each of the three model branches (see Figure 3) consist of two 1024-d fully-connected layers (with ReLU [25]) followed by specific output layers for each output type (box, class, action, target).

Given a model pre-trained on ImageNet [7], we first train the object detection branch on the COCO `train` set (excluding the V-COCO `val` images). This model, which is in essence Faster R-CNN, has 33.8 object detection AP on the COCO `val` set. Our full model is initialized by this object detection network. We prototype our human-object interaction models on the V-COCO `train` split and perform hyperparameter selection on the V-COCO `val` split. After fixing these parameters, we train on V-COCO `trainval` (5k images) and report results on the 5k V-COCO `test` set.

We fine-tune our human-object interaction models for 10k iterations on the V-COCO `trainval` set with a learning rate of 0.001 and an additional 3k iterations with a rate of 0.0001. We use a weight decay of 0.0001 and a momentum of 0.9. We use synchronized SGD [20] on 8 GPUs, with each GPU hosting 2 images (so the effective mini-batch size per iteration is 16 images). The fine-tuning time is ~2.5 hours on the V-COCO `trainval` set on 8 GPUs.

Baselines. To have a fair comparison with Gupta & Malik [14], which used VGG-16 [30], we reimplement their best-performing model ('model C' in [14]) using the same ResNet-50-FPN backbone as ours. In addition, [14] only reported AP_{role} on a subset of 19 actions, but we are interested in *all* actions (listed in Table 2). We therefore report comparisons in both the 19-action and all-action cases.

The baselines from [14] are shown in Table 1. Our reimplementation of [14] is solid: it has 37.5 AP_{role} on the 19 action classes tested on the `val` set, 11 points higher than the 26.4 reported in [14]. We believe that this is mainly due to ResNet-50 and FPN. This baseline model, when trained on the `trainval` set, has 31.8 AP_{role} on all action classes tested on the `test` set. This is a strong baseline (31.8 AP_{role}) to which we will compare our method.

Our method, InteractNet, has an AP_{role} of **40.0** evaluated on all action classes on the V-COCO `test` set. This is an absolute gain of 8.2 points over the strong baseline's 31.8, which is a relative improvement of 26%. This result quantitatively shows the effectiveness of our approach.

Qualitative Results. We show our human-object interaction detection results in Figure 2 and Figure 5. Each subplot illustrates one detected ⟨`human`, `verb`, `object`⟩ triplet, showing the location of the detected person, the action taken by this person, and the location (and category) of the detected target object for this person/action. Our method can successfully detect the object outside of the person bounding box and associate it to the person and action.

Figure 7 shows our correctly detected triplets of one person taking multiple actions on multiple objects. We note that in this task, one person can take multiple actions and affect multiple objects. This is part of the ground-truth and evaluation and is unlike traditional object detection tasks [8] in which one object has only one ground-truth class.

Moreover, InteractNet can detect multiple interaction instances in an image. Figure 6 shows two test images with *all detected triplets* shown. Our method detects multiple persons taking different actions on different target objects.

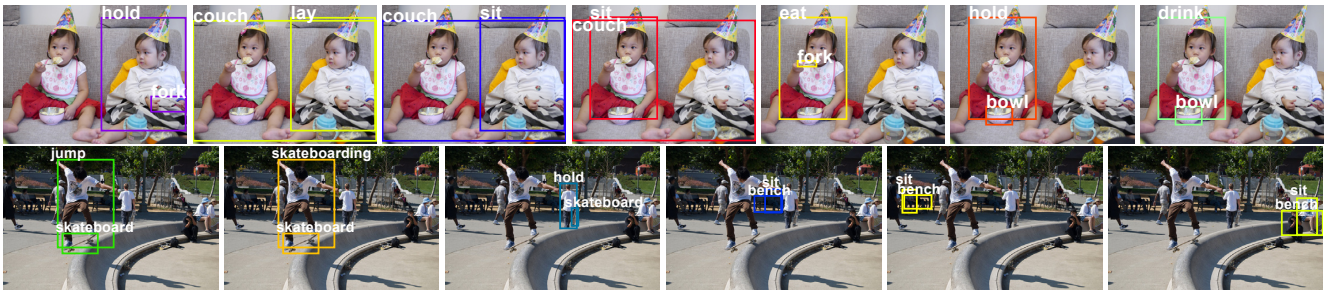


Figure 6. **All detected triplets** on two V-COCO test images. We show all triplets whose scores (1) are higher than 0.01.

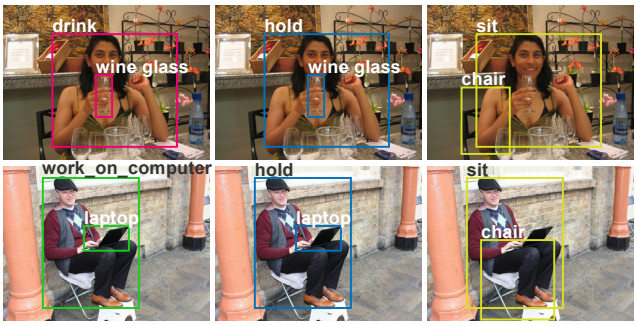


Figure 7. Results of InteractNet on test images. An individual person can take multiple actions and affect multiple objects.

mean AP_{role} evaluated on	19 actions (on `val`)	all actions (on `test`)
model B of [14] [VGG-16]	7.9	N/A
model C of [14] [VGG-16]	26.4	N/A
model C of [14] [ResNet-50-FPN] **ours**	37.5	31.8
InteractNet [ResNet-50-FPN]	**41.9**	**40.0**

Table 1. Comparisons with Gupta & Malik [14]. To have an apples-to-apples comparisons, we reimplement [14]'s 'model C' using ResNet-50-FPN. In addition, [14] reported AP_{role} on a subset consisting of 19 actions, and only on the `val` set. As we evaluate on *all* actions (more details in Table 2), for fair comparison, we also report the mean AP_{role} on these 19 actions of `val`, with models trained on `train`. Our reimplemented baseline of [14] is solid, and InteractNet is considerably better than this baseline.

The multi-instance, multi-action, and multi-target results in Figure 6 and Figure 7 are all detected by one forward pass in our method, running at about 135ms per image on a GPU.

Ablation Studies. In Table 3–5 we evaluate the contributions of different factors in our system to the results.

With vs. *without target localization.* Target localization, performed by the human-centric branch, is the key component of our system. To evaluate its impact, we implement a variant *without* target localization. Specifically, for each type of action, we perform k-means clustering on the offsets between the target RoIs and person RoIs (via cross-validation we found $k = 2$ clusters performs best). This plays a role similar to density estimation, but is *not* aware of the person appearance and thus is not instance-dependent. Aside from this, the variant is the same as our full approach.

	baseline [14] our impl.		InteractNet w/o target loc.		**InteractNet**	
	AP_{agent}	**AP_{role}**	AP_{agent}	**AP_{role}**	AP_{agent}	**AP_{role}**
carry	62.2	8.1	63.9	14.4	64.8	**33.1**
catch	47.0	37.0	53.4	38.5	57.1	**42.5**
drink	11.9	18.1	37.5	25.4	46.0	**33.8**
hold	79.4	4.0	77.3	10.6	80.1	**26.4**
jump	75.5	40.6	75.6	39.3	74.7	**45.1**
kick	60.9	67.9	68.6	**70.6**	77.5	69.4
lay	50.1	17.8	51.1	18.6	47.6	**21.0**
look	68.8	2.8	61.0	2.7	59.4	**20.2**
read	34.9	23.3	43.2	22.0	41.6	**23.9**
ride	73.2	**55.3**	76.2	55.0	74.2	55.2
sit	76.8	15.6	75.6	15.1	76.1	**19.9**
skateboard	89.9	74.0	90.9	71.7	90.0	**75.5**
ski	84.0	29.7	83.9	28.2	84.7	**36.5**
snowboard	81.3	52.8	81.1	50.6	81.1	**63.9**
surf	94.6	50.2	94.5	50.3	93.5	**65.7**
talk-on-phone	63.3	23.0	74.7	23.8	82.0	**31.8**
throw	54.0	36.0	53.9	35.7	58.1	**40.4**
work-on-computer	70.2	46.1	72.6	46.9	75.7	**57.3**
cut (*object*)	61.2	16.5	69.1	17.7	73.6	**23.0**
cut (*instrument*)		15.1		19.5		**36.4**
eat (*object*)	75.6	26.5	80.4	26.5	79.6	**32.4**
eat (*instrument*)		2.7		**2.9**		2.0
hit (*object*)	82.8	56.7	83.9	55.3	88.0	**62.3**
hit (*instrument*)		42.4		41.3		**43.3**
point	5.0	–	4.0	–	1.8	–
run	76.9	–	77.8	–	77.2	–
smile	60.6	–	60.3	–	62.5	–
stand	88.5	–	88.3	–	88.0	–
walk	63.9	–	63.5	–	65.4	–
mean AP	65.1	31.8	67.8	32.6	69.2	**40.0**

Table 2. Detailed results on V-COCO `test`. We show two main baselines and InteractNet for each action. There are 26 actions defined in [14], and because 3 actions (*cut*, *eat*, *hit*) involve two types of target objects (instrument and direct object), there are 26+3 entries (more details in § 4). We bold the leading entries on AP_{role}.

Table 3 (a) *vs.* (c) shows that our target localization contributes significantly to AP_{role}. Removing it shows a degradation of 5.6 points from 37.5 to 31.9. This result shows the effectiveness of our target localization (see Figure 4). The per-category results are in Table 2.

With vs. *without the interaction branch.* We also evaluate a variant of our method when removing the interaction branch. We can instead use the action prediction from the human-centric branch (see Figure 3). Table 3 (b) *vs.* (c) shows that removing the interaction branch reduces AP_{role} just slightly by 0.7 point. This again shows the main effectiveness of our system is from the target localization.

	AP_{agent} ⟨human, verb⟩	**AP_{role} ⟨human, verb, object⟩**
baseline [14] (our implementation)	62.1	31.0
(a) InteractNet w/o target localization	65.1	31.9
(b) InteractNet w/o interaction branch	65.5	36.8
(c) InteractNet	**68.0**	**37.5**

Table 3. Ablation studies on the V-COCO `val` set, evaluated by AP_{agent} (*i.e.*, AP of the ⟨human, verb⟩ pairs) and AP_{role} (*i.e.*, AP of the ⟨human, verb, object⟩ triplets). All methods are based on ResNet-50-FPN, including our reimplementation of [14]. Table 2 shows the detail numbers of three entries: baseline, InteractNet without target density estimation, and our complete method on the V-COCO `test` set.

	AP_{agent} ⟨human, verb⟩	**AP_{role} ⟨human, verb, object⟩**
InteractNet w/ ResNet-50	65.0	35.9
InteractNet w/ ResNet-50-FPN	**68.0**	**37.5**

Table 4. Ablation on the V-COCO `val` for vanilla ResNet-50 *vs.* ResNet-50-FPN [21] backbones.

	AP_{agent} ⟨human, verb⟩	**AP_{role} ⟨human, verb, object⟩**
InteractNet w/ pairwise concat + MLP	67.1	**37.5**
InteractNet	**68.0**	**37.5**

Table 5. Ablation on the V-COCO `val` set about the design of the pairwise interaction branch. See main text for explanations.

With vs. *without FPN.* Our model is a generic human-object detection framework and can support various network backbones. We recommend using the FPN [21] backbone, because it performs well for *small objects* that are more common in human-object detection.

Table 4 shows a comparison between ResNet-50-FPN and a vanilla ResNet-50 backbone. The vanilla version follows the ResNet-based Faster R-CNN presented in [17]. Specifically, the full-image convolutional feature maps are from the last residual block of the 4-th stage (res4), on which the RoI features are pooled. On the RoI features, each of the region-wise branches consists of the residual blocks of the 5-th stage (res5). Table 4 shows a degradation of 1.6 points in AP_{role} when not using FPN. We argue that this is mainly caused by the degradation of the small objects' detection AP, as shown in [21]. Moreover, the vanilla ResNet-50 backbone is much slower, 225ms versus 135ms for FPN, due to use of res5 in the region-wise branches.

Pairwise Sum vs. *MLP.* In our interaction branch, the pairwise outputs from two RoIs are added (Figure 3). Although simple, we have found that more complex variants do not improve results. We compare with a more complex transform in Table 5. We concatenate the two 1024-d features from the final fully-connected layers of the interaction branch for the two RoIs and feed it into an 2-layer MLP (512-d with ReLU for its hidden layer), followed by action classification. This variant is slightly worse (Table 5), indicating that it is not necessary to perform a complex pairwise transform (or there is insufficient data to learn this).

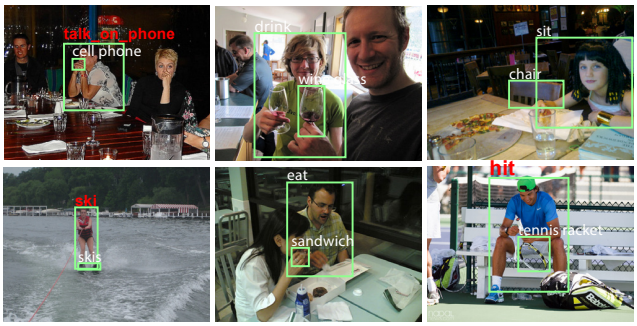


Figure 8. False positive detections of our method.

method	full	rare	non-rare
results from [3]	7.81	5.37	8.54
baseline [14] (our impl.)	9.09	7.02	9.71
InteractNet	**9.94**	**7.16**	**10.77**

Table 6. Results on HICO-DET `test` set. InteractNet outperforms the approach in [3] with a 27% relative improvement. We also include our baseline approach, as described in Table 1.

Per-action accuracy. Table 2 shows the AP for each action category defined in V-COCO, for the baseline, InteractNet without target localization, and our full system. We observe leading performance of AP_{role} consistently. The actions with largest improvement are those with high variance in the spatial location of the object such as *hold*, *look*, *carry*, and *cut*. On the other hand, actions such as *ride*, *kick*, and *read* show small or no improvement.

Failure Cases. Figure 8 shows some false positive detections. Our method can be incorrect because of false interaction inferences (*e.g.*, top left), target objects of another person (*e.g.*, top middle), irrelevant target objects (*e.g.*, top right), or confusing actions (*e.g.*, bottom left, *ski vs.surf*). Some of them are caused by a failure of *reasoning*, which is an interesting open problem for future research.

Mixture Density Networks. To improve target localization prediction, we tried to substitute the uni-modal regression network with a Mixture Density Network (MDN) [1]. The MDN predicts the mean and variance of M relative locations for the objects of interaction conditioned on the human appearance. Note that MDN with $M = 1$ is an extension of our original approach that also learns the variance in (2). However, we found that the MDN layer does not improve accuracy. More details and discussion regarding the MDN experiments can be found in [11].

HICO-DET Dataset. We additionally evaluate InteractNet on HICO-DET [3] which contains 600 types of interactions, composed of 117 unique verbs and 80 object types (identical to COCO objects). We train InteractNet on the `train` set, as specified by the authors, and evaluate performance on the `test` set using released evaluation code. Results are shown in Table 6 and discussed more in [11].

References

[1] C. M. Bishop. Mixture density networks. Technical report, 1994. 8

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *CVPR*, 2017. 1, 2

[3] Y.-W. Chao, Y. Liu, X. Liu, H. Zeng, and J. Deng. Learning to detect human-object interactions. In *arXiv preprint arXiv:1702.05448*, 2017. 2, 5, 8

[4] Y.-W. Chao, Z. Wang, Y. He, J. Wang, and J. Deng. HICO: A benchmark for recognizing human-object interactions in images. In *ICCV*, 2015. 5

[5] C.-Y. Chen and K. Grauman. Predicting the location of interactees in novel human-object interactions. In *ACCV*, 2014. 1

[6] V. Delaitre, I. Laptev, and J. Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *BMVC*, 2010. 1, 2

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. 2010. 5, 6

[9] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 3, 4, 5

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[11] G. Gkioxari, R. Girshick, P. Dollár, and K. He. Detecting and recognizing human-object intaractions. In *arXiv preprint arXiv:1704.07333*, 2017. 8

[12] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with R*CNN. In *ICCV*, 2015. 1, 2

[13] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *TPAMI*, 2009. 1, 2

[14] S. Gupta and J. Malik. Visual semantic role labeling. *arXiv 1505.04474*, 2015. 1, 2, 5, 6, 7, 8

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 2, 3, 6

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6, 8

[18] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[19] D. Le, R. Bernardi, and J. Uijlings. TUHOI: trento universal human object interaction dataset. In *Vision and Language Workshop at COLING*, 2014. 5

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. 1989. 6

[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6, 8

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[23] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016. 2

[24] A. Mallya and S. Lazebnik. Learning models for actions and person-object interactions with transfer to question answering. In *ECCV*, 2016. 2

[25] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 6

[26] V. Ramanathan, C. Li, J. Deng, W. Han, Z. Li, K. Gu, Y. Song, S. Bengio, C. Rossenberg, and L. Fei-Fei. Learning semantic relationships for better action retrieval in images. In *CVPR*, 2015. 2

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 3, 4, 6

[28] M. R. Ronchi and P. Perona. Describing common human visual actions in images. In *BMVC*, 2015. 5

[29] M. Sadeghi and A. Farhadi. Recognition using visual phrases. In *CVPR*, 2011. 2

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 6

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[32] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1, 2

[33] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010. 1, 2

[34] M. Yatskar, L. Zettlemoyer, and A. Farhadi. Situation recognition: Visual semantic role labeling for image understanding. In *CVPR*, 2016. 2

[35] L. Yu, E. Park, A. C. Berg, and T. L. Berg. Visual Madlibs: Fill in the blank Image Generation and Question Answering. In *ICCV*, 2015. 2