POSE ESTIMATION USING TENSORFLOW

An Industry Oriented Mini Project Work (CS705PC)

Submitted

in partial fulfillment of the requirements for

the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

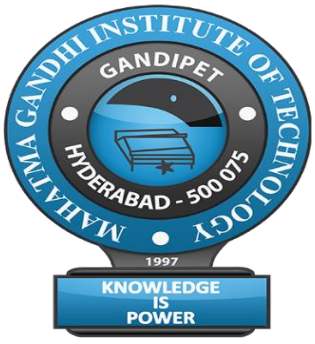
Mr. G.ROHAN

(17261A05D8)

Under the Guidance of

Ms. GOUSIYA BEGUM

(Assistant Professor)



Dept. of Computer Science and Engineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

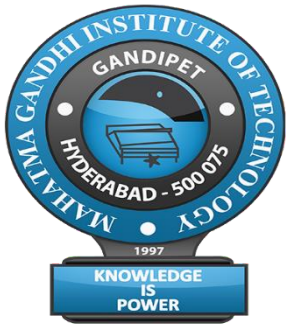
GANDIPET, HYDERABAD - 500075, INDIA

2020- 2021

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Gandipet, Hyderabad - 500075, Telangana.

CERTIFICATE



This is to certify that the project entitled **Pose Estimation using Tensorflow** is being submitted by **G.Rohan** bearing roll no:**17261A05D8** in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out by him under our guidance and supervision.

Supervisor
Ms. Gousiya Begum
Asst. Professor

Head of the Department
Dr. C.R.K. Reddy
Professor

External Examiner

DECLARATION

This is to certify that the work reported in this project titled "**Pose Estimation using Tensorflow"** is a record of work done by me in the **Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.**

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred to in the text. This report is based on the work done entirely by me and not copied from any other source.

The results embodied in this project have not been submitted to any other university or Institute for the award of any degree or diploma.

G.ROHAN(17261A05D8)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible because success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance. So, I acknowledge all those whose guidance and encouragement served as a beacon light and crowned my efforts with success.

I am also thankful to **Dr. K. Jaya Sankar, Principal MGIT**, for providing the work facilities in the college.

I convey my heartfelt thanks to **Dr. C.R.K Reddy**, **Professor & HOD**, Department of Computer Science and Engineering, MGIT, for all the timely support and valuable suggestions during the period of the project.

I am extremely thankful to **Dr. M Rama Bai, Professor** and **Dr. V. Subbaramaiah, Senior Assistant Professor**, Department of CSE, Major Project Coordinators, MGIT, for their encouragement and support throughout the project.

I would like to express my sincere thanks to my guide **Ms. Gousiya Begum**, **Assistant Professor**, Department of CSE, MGIT, for her constant guidance, encouragement and moral support throughout the project.

Finally, I would like to thank all the faculty and staff of the CSE Department who helped me directly or indirectly, for completing this project.

G.ROHAN(17261A05D8)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Human body posture recognition has been an important concern of research in many fields. In this field, a human pose estimation algorithm called OpenPose has been more widely used. But its efficiency is very low. This work proposes deep learning methods based on TensorFlow to recognize human body postures. And work applied it to the judgment of the teacher's teaching states. In order to choose the algorithm that works best for our scenario, this work designed eight sets of experimental schemes through combining the classification model and the detection algorithm. Then these groups of schemes were used to detect and classify the teacher's teaching states,including standing, sitting, etc. At last, the analysis of the experimental results in depth is done and selected the most suitable algorithm for our scenario.Video-based human pose estimation in crowded scenes is a challenging problem due to occlusion, motion blur, scale variation and viewpoint change, etc. Prior approaches always fail to deal with this problem because of (1) lacking of usage of temporal information; (2) lacking of training data in crowded scenes. In this project, the main focus is given on improving human pose estimation in videos of crowded scenes from the perspectives of exploiting temporal context and collecting new data.

1. INTRODUCTION

Human body posture recognition has been widely used in intelligent monitoring, human-computer interaction, video retrieval, virtual reality, etc. It has always been an active research direction in the field of computer vision. Early human body postures recognition mainly relies on multiple sensors in different parts of the human body[1]. The method of human body posture recognition based on sensors of smartphones and wearable devices(such as wristbands and watches) has been already mainstream .Traditional machine learning methods, such as SVM or Bayesian networks, time-domain and frequency domain analysis, etc.. These machine learning methods require professional knowledge in the human body posture domain for feature extraction. In recent years, there are two major directions about the recognition of human body posture[2]. One is to use object detection methods based on deep learning, directly use CNN to train, identify and classify human body postures.

1.1 Problem Definition:

Pose Estimation using Tensorflow can be implemented on a jupyter notebook using python. The main aim of this project is to estimate the pose of an Articulated Human with flexible mixtures of parts in an image or a video[3]. In this project, the bottom-up algorithm is used to obtain the key points and then obtain the backbone. It can achieve accurate estimation of the human body posture in the picture[4].

1.2 Project Scope

The scope of this project is to:

- Create a learning slope to get more people interested.
- Provide a rudimentary understanding of Pose Estimation and possibly spark an interest in the field.
- Enhance security surveillance.
- Provide an environment for training robots.
- Track motion for Consoles.
- Capture motion and Augmented Reality.
- Track the variations in the pose of a person over a period of time for activity, gesture and gait recognition.

1.3 Existing System

There are a lot of models of humans for estimating their pose. Each model gets more complex as the number of people gets increased in a video or an image. All the existing models give less accuracy and precision.

Problems in Existing System

- Each image may contain an unknown number of people that can appear at any position or scale Less accuracy and precision.
- Interactions between people induce complex spatial interference, due to contact, occlusion, or limb articulations, making association of parts difficult.
- Runtime complexity tends to grow with the number of people in the image, making real-time performance a challenge Less attributes.
- Complicated background and depth ambiguities.

1.4 Proposed System

The proposed method of OpenPose uses a nonparametric representation to "connect" each finds body joints on an image, associating them with individual people. OpenPose finds the key points on an image of the number of people on it. Performance varies based on your device and output stride . Tensorflow library is image size invariant, which means it can predict pose positions in the same scale as the original image regardless of whether the image is downscaled. This means Tensorflow can be configured to have a higher accuracy at the expense of performance.

The proposed system consists of the following goals and advantages:

Goals

- To estimate the pose of a human regardless of the number of people in an image.
- To connect the joints accurately and generate pose for utilizing its applications.
- To get high accuracy and precision.

Advantages

- They have fewer intra-class variances.
- It is very easy to implement.
- It is very efficient.
- Using pose representations simplifies learning for action recognition, since relevant high level information has already been extracted..
- Use of confidence maps to map individual body parts/regions (i.e. shoulder*)*.

- Scales well to GPU over CPU.
- High accuracy and precision.
- Greedy parsing algorithms are effective in terms of runtime.
- High accuracy without compromise on execution performance.

1.5 Requirements Specification

1.5.1 Software Requirements

The following are the software requirements of Pose Estimation of Tensorflow.

Operating System	: Window 7/8/10.
Platform	: Jupyter Notebook.
Language	: Python 3.5 and above.
library used	: TensorFlow.
Packages required	: Matplotlib, Computer Vision(cv)

1.5.2 Hardware Requirements

The following details are the hardware requirements for Pose Estimation Using Tensorflow.

RAM	: 4 Gb or more
Hard Disk	: 10 Gb minimum hard disk
Processor	: Any Intel Processor

1.6 Software Tools Used

1.6.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is simple and easy to learn. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy and a bug or bad input will never cause a segmentation fault[5]. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and

global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. The Proposed System works on python 3.5 and above.

1.6.2 Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

1.6.3 TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming. TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (withoptional CUDA and SYCL extensionsfor general-purpose computing on graphics processing units).TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

1.6.4 Computer Vision(cv) package

Computer Vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do. The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multidimensional data from a 3D scanner, or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems.

1.6.5 Django

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. Django is very demanding due to its rapid development feature. It takes less time to build an application after collecting client requirements. By using Django, we can build web applications in very less time[1]. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only. The features of Django are Rapid development, Secure, Scalable, Versatile and Open source.

2. LITERATURE SURVEY

- **Author-A. Agarwal and B. Triggs, "Recovering 3d human pose from monocular images".**

  They developed it by Marker-less tracking of human joints; ConvNet-based 2D joint detection is combined with generative tracking algorithms based on sums of Gaussians.A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights[1].

- **Author-M. Andriluka, S. Roth, "People detection and articulated pose estimation".**

  They developed it by Loose-Limbed body model representation with continuous-valued parameters and bottom-up part detectors.Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen[2].

- **Author-M. Andriluka, S. Roth, and B. Schiele, "Rich feature hierarchies for accurate object detection and semantic segmentation"**

  They developed it by training a deep ConvNet; and joint point regression to estimate the positions of joint points relative to the root position and joint point detection to classify whether one local window contains the specific joint[3].

- **Author-M. Bergtholdt, J. Kappes,"A study of parts based object class detection using complete graphs"**

  They developed it by employing a 2D part detector with an occlusion detection step; Create multiple views synthetically with a twin-GPR in a cascaded manner; Kinematic and orientation constraints to resolve remaining ambiguities[4].

Table 2.1 shows the comparison study of literature survey of Pose Estimation Using TensorFlow. It contains the name of the author, title of proposed work, method, accuracy, objective, results of various existing systems.

Table 2.1: Literature survey of Pose Estimation using Tensorflow

S.No	Author	Title	Method	Accuracy	Advantages
1	A. Agarwal and B. Triggs	Human Pose Analysis	ConvNet	90.8%	Direct regression, multi-stage refinement
2	Andriluka,	3D Human Detection	Bottom-up algorithm	67.9%	Iterative error feedback refinement from initial pose.
3	Roth	Human Postures and Object Detection	LR model	70.8%	Bone based representation as additional constraint, general for both 2D/3D HP
4	Schiele	Analysis of Human Gesture Recognition	SVM	75.3%	Heatmap representation, multi-scale input, MRF-like Spatial-Mode
5	Kappes	Human Pose Estimation using a Joint Pixel-wise and Part-wise Formulation	Naive Bayes	69.8%	An extension of Mask R-CNN framework; predicts keypoints and human mask synchronouslyfrom an RGB image

3. DESIGN METHODOLOGY OF POSE ESTIMATION

3.1 OpenPose

The openPose from Carnegie Mellon University is one of the most accurate methods for human pose estimation. This convolutional neural network based approach attacks the problem using a multi-stage classifier where each stage improves the results of the previous one.The first stage takes the input image and predicts the possible locations of each keypoint in the image with a confidence score(called the confidence map).

Fig 3.1(a) The first stage with an image

Fig 3.1(b) Generating confidence map

Each subsequent stage takes not only the image data but also the confidence map of the previous stage to use the context of the nearby area from the previous stage. This improves the prediction after each step, i.e. the confidence map is more accurate after passing through all 4 stages.

Figure 3.2 After three stages of human postures.

It achieves this capability by propagating known person locations forward and backward in time and searching for poses in those regions.Both use case scenarios require the action recognition solution to have near real-time responses, and be able to run on a resource-constrained edge device.

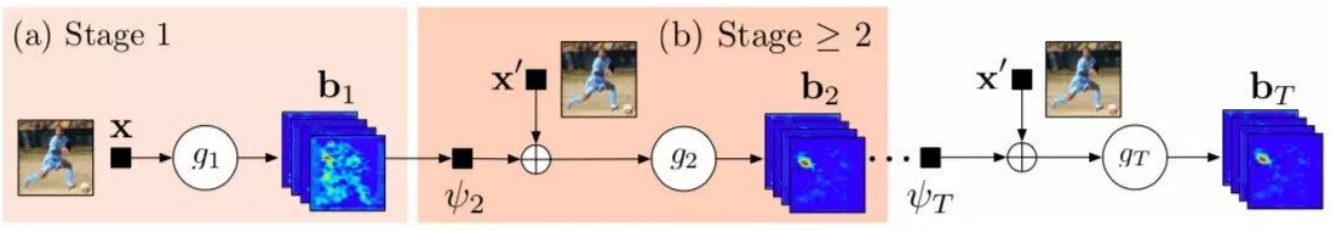


Figure 3.3 Confidence map of 3 stages

Confidence map is good for single person pose estimation. However, the authors tried to build a solution for a general multi-person human pose estimation. Multi-person human pose estimation has additional many challenges such as an unknown number of people in the image, occlusion, variation in people scale.

3.2 Preprocessing

Confidence map is good for single person pose estimation. However, the authors tried to build a solution for a general multi-person human pose estimation. Multi-person human pose estimation has additional many challenges such as an unknown number of people in the image, occlusion, variation in people scale. Using confidence, you can only generate all the key points as shown in the image below:

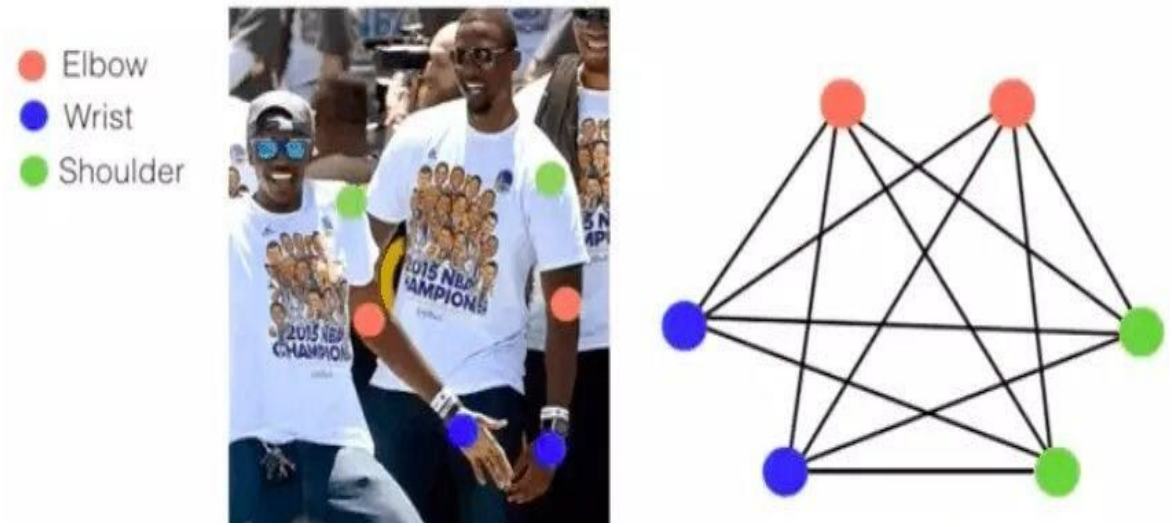


Figure 3.4 Part Association for Full-body Pose.

So, the idea of part affinity maps has been introduced. It uses the fact that certain joints are attached via limbs so another cnn branch is introduced which predicts 2D vector fields called part affinitmaps(PAFs). PAFs encodes the location and orientation of the limbs.

Confidence Maps and Part Affinity Fields

- **Confidence Maps:** A Confidence Map is a 2D representation of the belief that a particular body part can be located in any given pixel. Confidence Maps are described by following equation:

$$S = (S1,S2\ldots\ldots,Sj) \text{ where } Sj \in Rw*h, \quad j \in 1\ldots.J$$

where *J* is the number of body parts locations.

Part Affinity Fields: Part Affinity is a set of 2D vector fields that encodes location and orientation of limbs of different people in the image. It encodes thee data in the form of pairwise connections between body parts.

$$L = (L1,L2\ldots..,Lg) \text{ where } Lc \in Rw*h*c, \quad c \in 1\ldots.g$$

Figure 3.5 Part Affinity Fields for Part-to-Part Association

Orange circle is the part 1 of the connection and blue circle is the part 2 of the connection.Later we can use the direction of the part affinity maps to predict human poses accurately in multiple people pose estimation problems. Now, in the final step, we can just connect these points using greedy inference to generate the pose keypoints for all the people in the image. The connection for one type of human limbs is determined first and then this is repeated for other types of limbs.

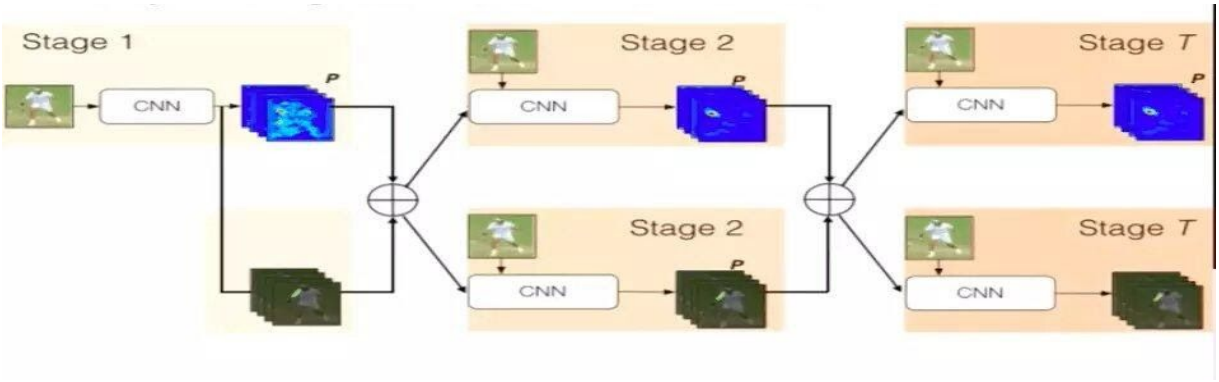


Figure 3.6 Jointly Learning Parts Detection and Parts Association

More specifically:

So, In summary, there are three steps:

I. **Key Points localization**: One branch of the network is responsible for predicting all the keypoints with a confidence score. This is called a confidence Map.

II. **Part Affinity Fields**: Another branch of the network predicts a 2D vector field that predicts the key points association data.

III. **Greedy Inference**: Now, we connect all the keypoints using greedy inference.

3.3 Feature Extraction

The 2D pose estimation algorithm can estimate many poses/persons in an image. It is more complex,but it has the advantage that if multiple people appear in a picture, their detected key points

are less likely to be associated with the wrong pose. For that reason, even if the use case is to detect a single person's pose, this algorithm may be more desirable. Moreover, an attractive property of this algorithm is that performance is not affected by the number of persons in the input image. Whether there are 15 people to detect or 5, the computation time will be the same.

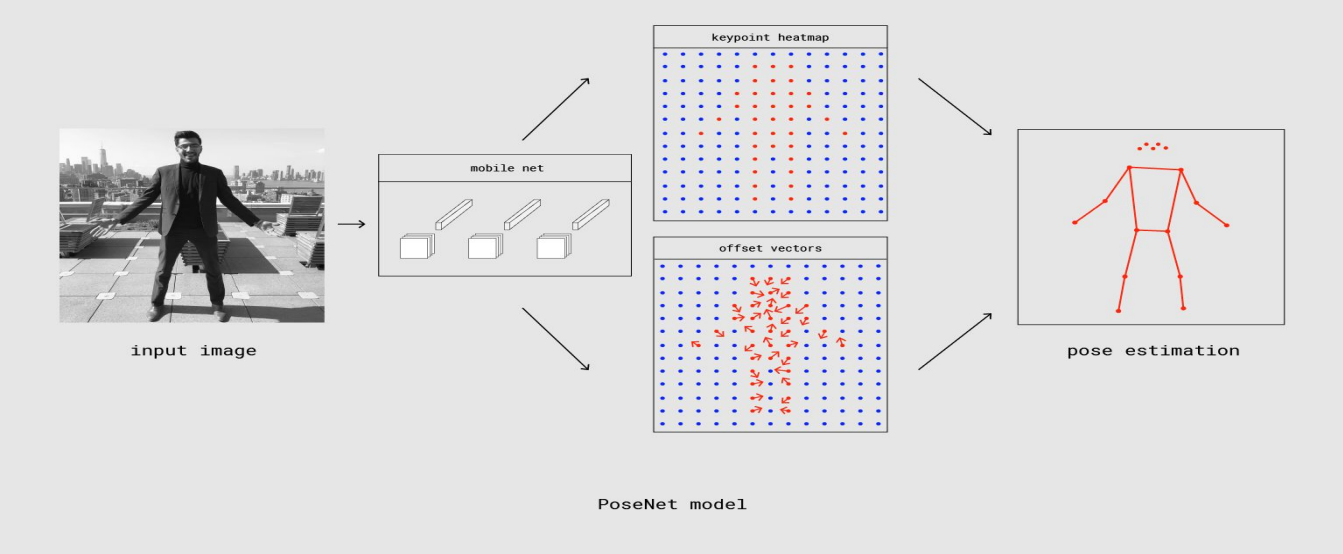


Figure 3.3 Feature Extraction

3.4 System Architecture

First, an input image is taken and processed. Second, the body modelling and preprocessing is done and then feature extraction in which certain joints are connected and then finally 2D pose is estimated based on certain limits.

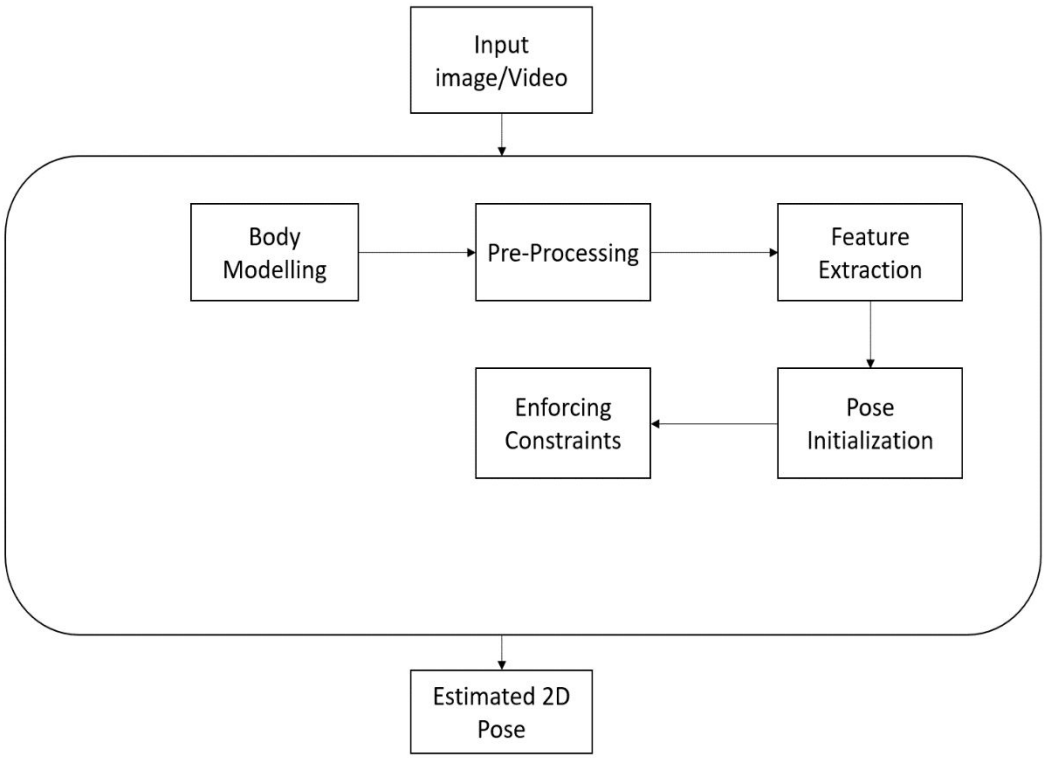


Figure 3.4 System Architecture of Pose Estimation

3.5 Convolutional Neural Network Algorithm

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a Convolutional Neural Network.

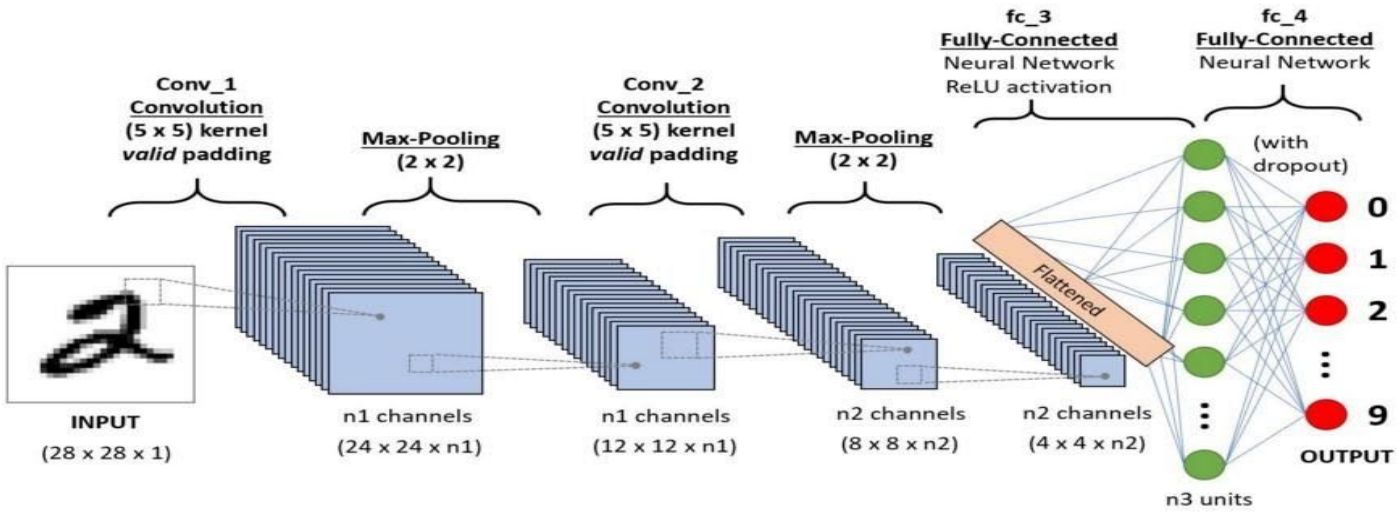


Figure 3.5 A CNN sequence for handwritten digits.

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

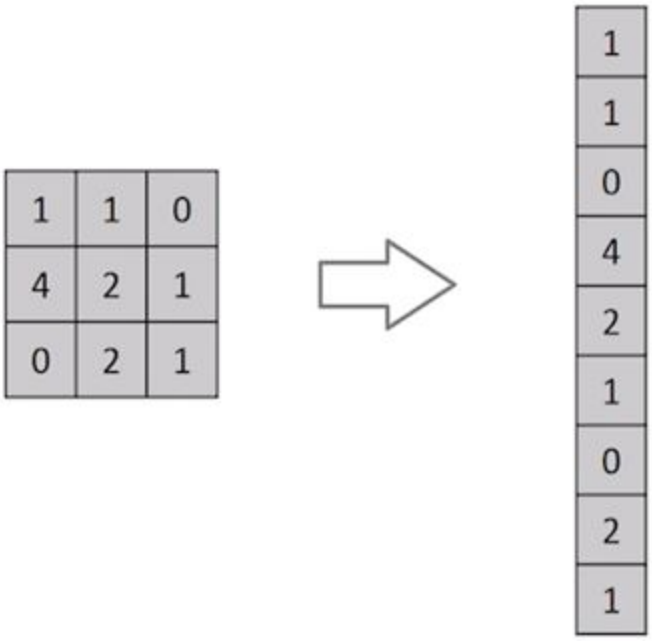


Figure 3.5.1 Flattening of 3*3 matrix into 9*1 vector.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

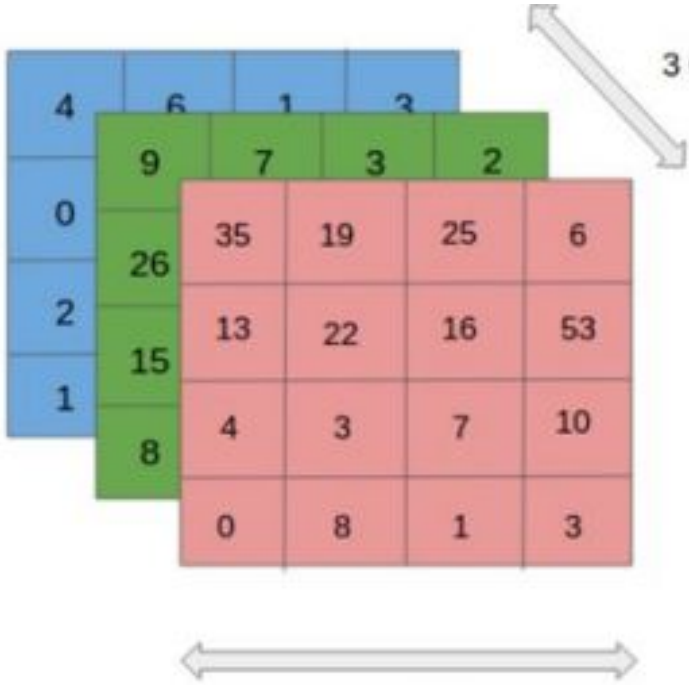


Figure 3.5.1 4*4*3 RGB Image(↔ is width, ↕ is height, ↘ is 3 color channels)

In the figure, we have an RGB image which has been separated by its three colour planes — Red, Green, and Blue. There are a number of such colour spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc. You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the CNN is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

3.6 POSE ESTIMATION ALGORITHM

Algorithm 1. The flow-based inference algorithm for video human pose tracking

input: video frames $\{I^k\}$, $Q = []$, Q's max capacity L_Q.
$B^0_{\text{det}} = \mathcal{N}_{\text{det}}(I^0)$
$\mathcal{J}^0 = \mathcal{N}_{\text{pose}}(I^0, B^0_{\text{det}})$
$\mathcal{P}^0 = (\mathcal{J}^0, id)$ ▷ initialize the id for the first frame
$Q = [\mathcal{P}_0]$ ▷ append the instance set $\mathcal{P}_0$ to Q
for $k = 1$ **to** ∞ **do**
    $B^k_{\text{det}} = \mathcal{N}_{\text{det}}(I^k)$
    $B^k_{\text{flow}} = \mathcal{F}_{\text{FlowBoxGen}}(\mathcal{J}^{k-1}, F_{k-1\to k})$
    $B^k_{\text{unified}} = \mathcal{F}_{\text{NMS}}(B^k_{\text{det}}, B^k_{\text{flow}})$ ▷ unify detection boxes and flow boxes
    $\mathcal{J}^k = \mathcal{N}_{\text{pose}}(I^k, B^k_{\text{unified}})$
    $M_{\text{sim}} = \mathcal{F}_{\text{sim}}(Q, \mathcal{J}^k)$
    $\mathcal{P}^k = \mathcal{F}_{\text{AssignID}}(M_{\text{sim}}, \mathcal{J}^k)$
    append $\mathcal{P}^k$ to Q ▷ update the Q
end for

In the first step the image is passed through a baseline CNN network to extract the feature maps of the input. In this paper the authors used the first 10 layers of VGG-19 network.

- The feature map is then processed in a multi-stage CNN pipeline to generate the Part Confidence Maps and Part Affinity Field.
- In the last step, the Confidence Maps and Part Affinity Fields that are generated above are processed by a greedy bipartite matching algorithm to obtain the poses for each person in the image.

4.SYSTEM DESIGN

4.1 Data Flow Diagram

Data flow diagram is a graphical representation that depicts information flow and the transforms that are applied as data move from input to output. The basic form of a data flow diagram, also known as a data flow graph or a bubble chart.Therefore, the DFD provides a mechanism for functional modelling as well as information flow modelling. The transformation of data from input to output, through processing, may be described logically and independently of physical components associated with the system.

The below figure 4.1 shows the complete data flow diagram for the Pose Estimation using TensorFlow.

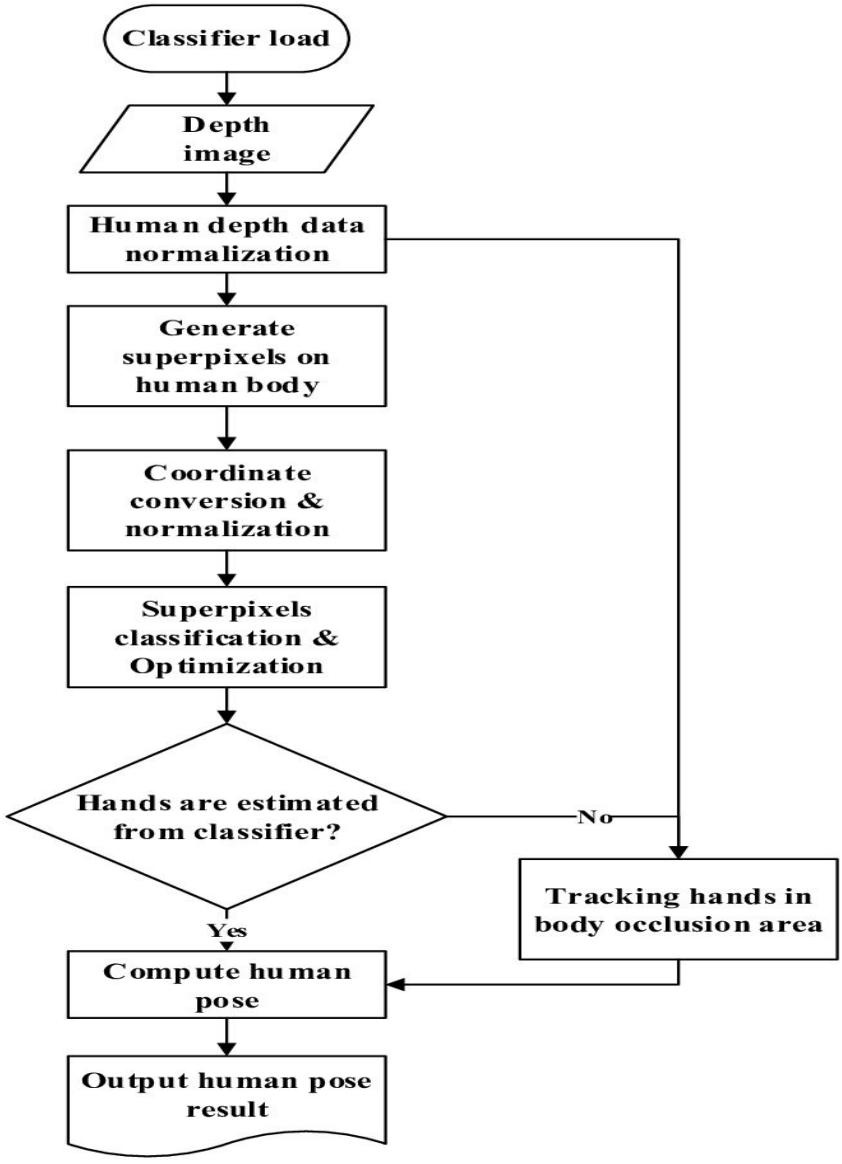


Figure 4.1 Data Flow Diagram

4.2 UML Diagrams

The Unified Modeling Language is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system[4]. UML diagrams are designed 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.

4.2.1 Class Diagram

A class diagram is a type of diagram and part of a unified modelling language (UML) that defines and provides the overview and structure of a system in terms of classes, attributes and methods, and the relationships between different classes.

Class diagram is used to illustrate and create a functional diagram of the system classes and serves as a system development resource within the software development life cycle.The first or top part specifies the class name, the second or middle specifies attributes of that class and the third or bottom section lists the methods or operations that specific class can perform.

Figure 4.2 shows the class diagram for Pose Estimation using Tensorflow. It contains poseSet, poseModel,Poseclassifier classes.
As shown in Figure 4.2 All the attributes of respective classes are listed below the class name followed by the operations that are performed in that class.

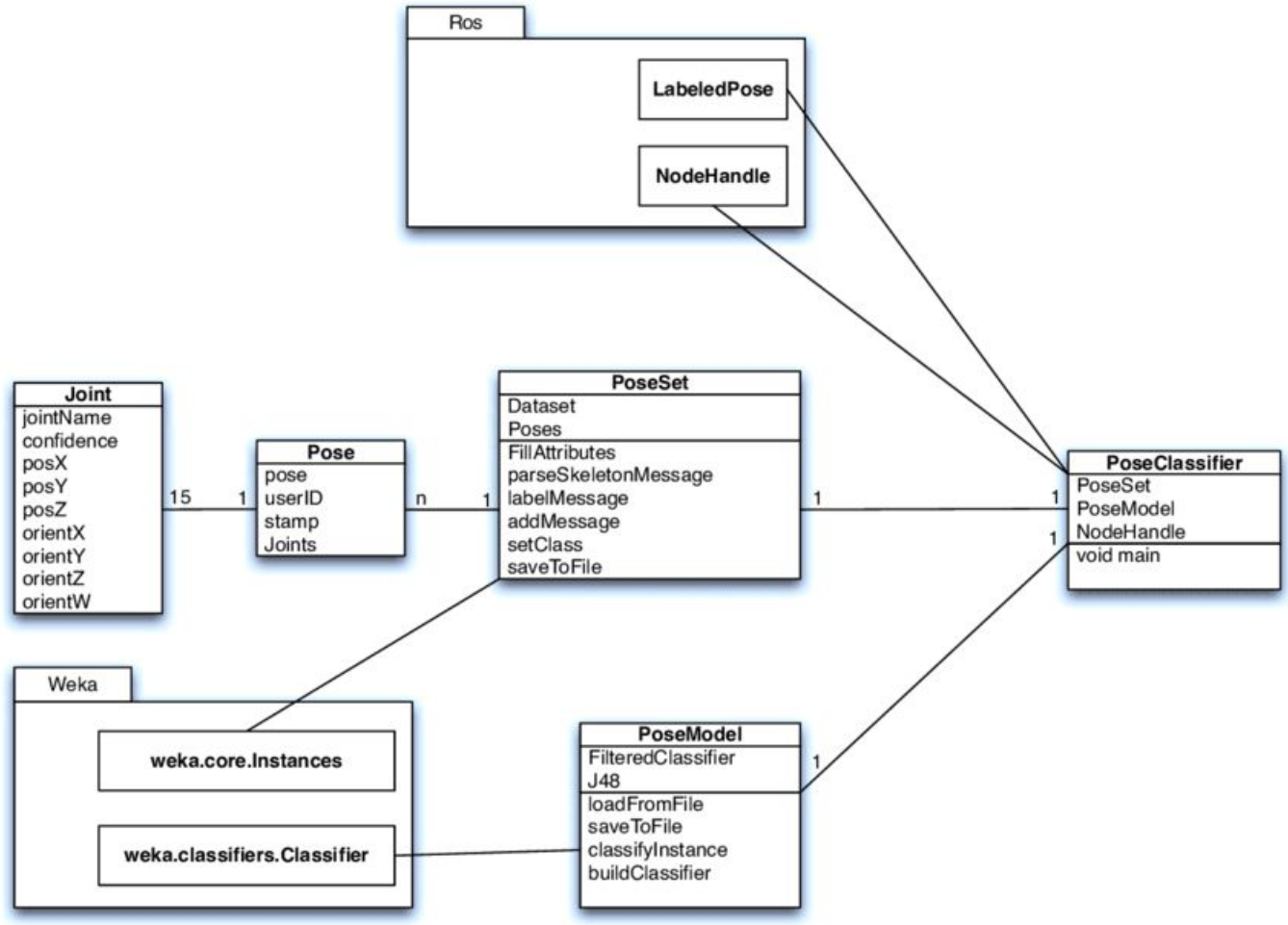


Figure 4.2 Class diagram for Pose Estimation using Tensorflow.

4.2.2 Use case Diagram

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system.Use cases focus on the behavior of the system from an external point of view. Actors are external entities that interact with the system.

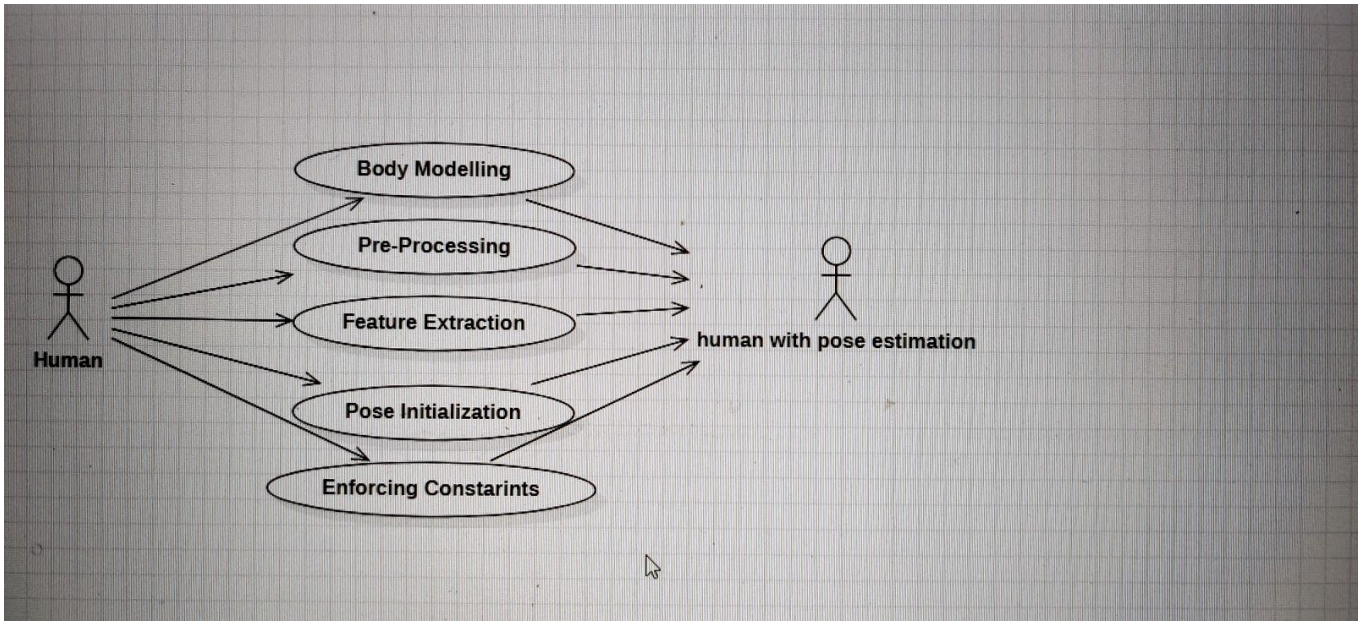


Figure 4.3: Use case diagram for Traffic Accidents Classification and Prediction.

Figure 4.3 shows the use case diagram for Pose Estimation USing Tensorflow. As shown in figure 4.3 Body modelling,pre-processing,Feature Extraction,Pose Initialization are steps for actors in use case diagrams.

4.2.3 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Figure 4.4 shows the sequence diagram of Pose Estimation using Tensorflow. It shows the step by step procedure followed by the project.

As shown in figure 4.4 the following are the steps mentioned in sequence diagram:

1. The kernel is initiated by starting the server.
2. An instruction is sent to the browser to open up a jupyter notebook.

3. The browser requests for libraries from file system via kernel (Ex.: Pandas, Numpy, etc.).
4. The Kernel interacts directly with the file system and requests for the required libraries
5. The file system responds to Kernel's request by fetching the libraries that were requested.
6. The Browser then requests for the dataset associated with the problem statement 7. The Kernel receives the request from Browser and passes it on to the file system.
8. The file system satisfies the request by fetching the required dataset.
9. Cleaning steps are requested for by the browser over the data set.
10. Various cleaning operations are performed over the data frame to get higher accuracy.
11. The target data is visualized.
12. The kernel responds with the kind of visualizations that were requested for by the browser.
13. Data transformation techniques are performed to normalize the attribute values in data set.
14. Different transformation techniques are applied over the data frame by the Kernel.
15. A model is trained over the data frame.
16. A tuned model is fit over the data set to provide a highly accurate model.
17. The associated predictions over the test data are visualized to analyze it's accuracy in comparison to the actual house prices.
18. Requested visualizations are displayed over the browser to better understand the accuracy of the trained model.
19. Evaluate the model on test set and visualize predictions.

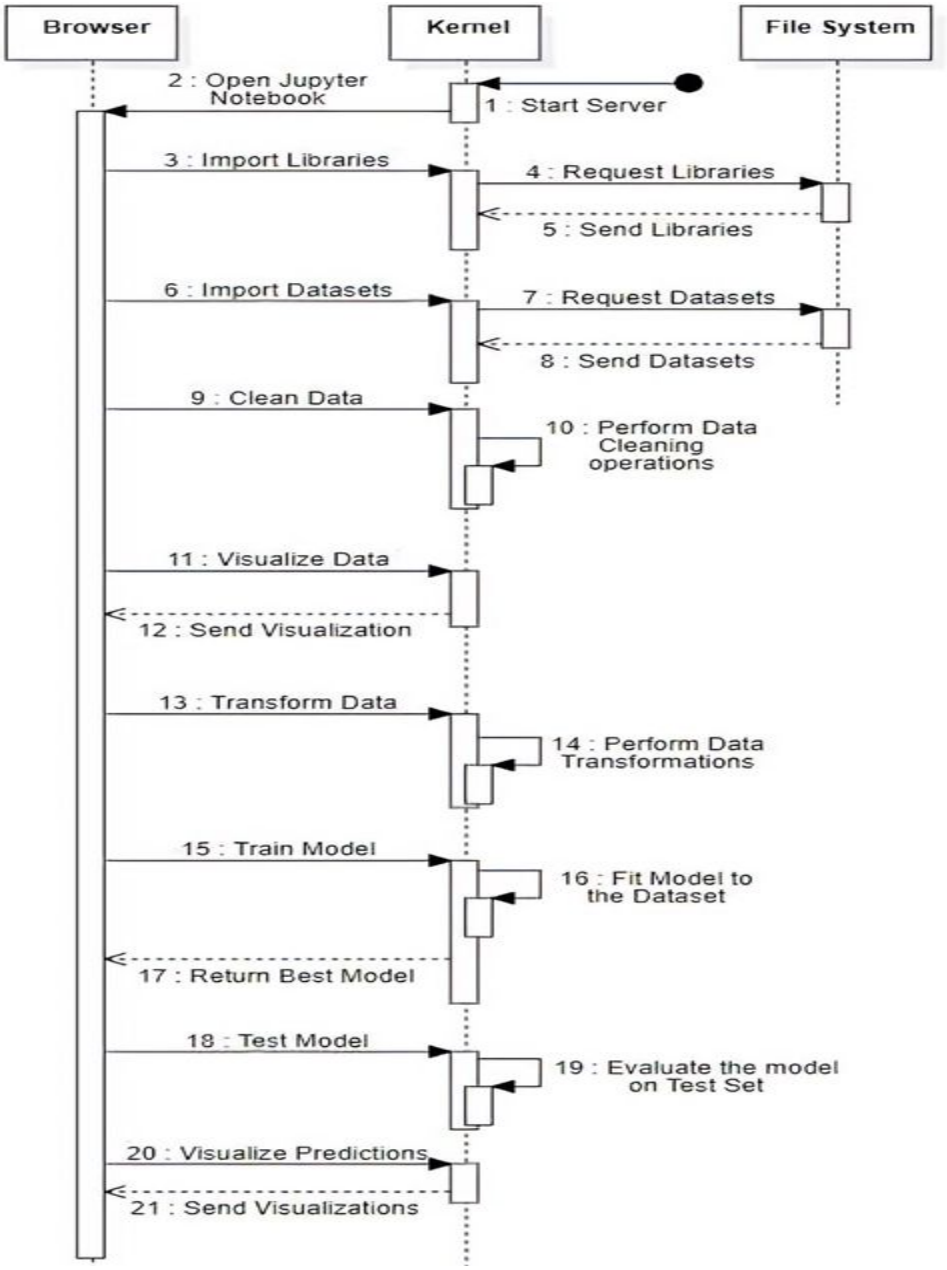


Figure 4.4: Sequence diagram for Pose Estimation using Tensor

5. IMPLEMENTATION

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

5.1 How to run project

Instructions on how to run the code using Jupyter notebook.

1) Go to anaconda prompt by searching it on the search bar.
2) Type Jupyter notebook and press Enter.
3) After getting into the Jupyter tab, search for the file that is required.
4) Double click on the file and open it.
5) Before running the code, import the packages named computer vision and matplotlib.
6) After importing them run the code using the run option above it.
7) Make sure to close all the remaining tabs or programs except for Jupyter notebook for smooth execution.
8) Go through the results.

5.1.1 How to Estimate Pose in a live web camera:

The following are the steps to Estimate Pose in a live web camera.

1) Install python version 3.4 or higher.
2) The code will be the same for the estimating the pose in image or a video.
3) To estimate pose in a video the following method “VideoCapture( )” should be used.
4) To estimate the pose in a live web camera the method “VideoCapture(1)” should be used.

Code:

The source code with the dataset developed for Pose Estimation Using Tensorflow is available at:

https://drive.google.com/file/d/1cwfWEqT0887l5Q91KFGps6LlxoJsAyBQ/view?usp=sharing

6. TESTING AND RESULTS

6.1 Testing

Machine Learning Testing (ML testing) refers to any activity designed to reveal machine learning bugs. An ML bug may exist in the data, the learning program, or the framework.

6.1.1 Offline Testing

At the very beginning, Pose estimation is a computer vision technique that predicts and tracks the location of a person or object. This is done by looking at a combination of the pose and the orientation of a given person/object. This is typically done by identifying, locating, and tracking a number of keypoints on a given object or person. For objects, this could be corners or other significant features. And for humans, thesṣts represent major joints like an elbow or knee. When the tests are ready, they need to be executed for developers to collect results. The test execution process involves building a model with the tests (when the tests are training data) or running a built model against the tests (when the tests are test data), as well as checking whether the test oracles are violated. After the process of test execution, developers may use evaluation metrics to check the quality of tests, i.e., the ability of the tests to expose ML problems.

The test execution results yield a bug report to help developers to duplicate, locate, and solve the bug. Those identified bugs will be labelled with different severity and assigned for different developers. Once the bug is debugged and repaired, regression testing is conducted to make sure the repair solves the reported problem and does not bring new problems. If no bugs are identified, the offline testing process ends, and the model is deployed.

6.1.2 Online Testing

Offline testing tests the model with historical data without in the real application environment. Online testing complements the shortage of offline testing, and aims to detect bugs after the model is deployed online.

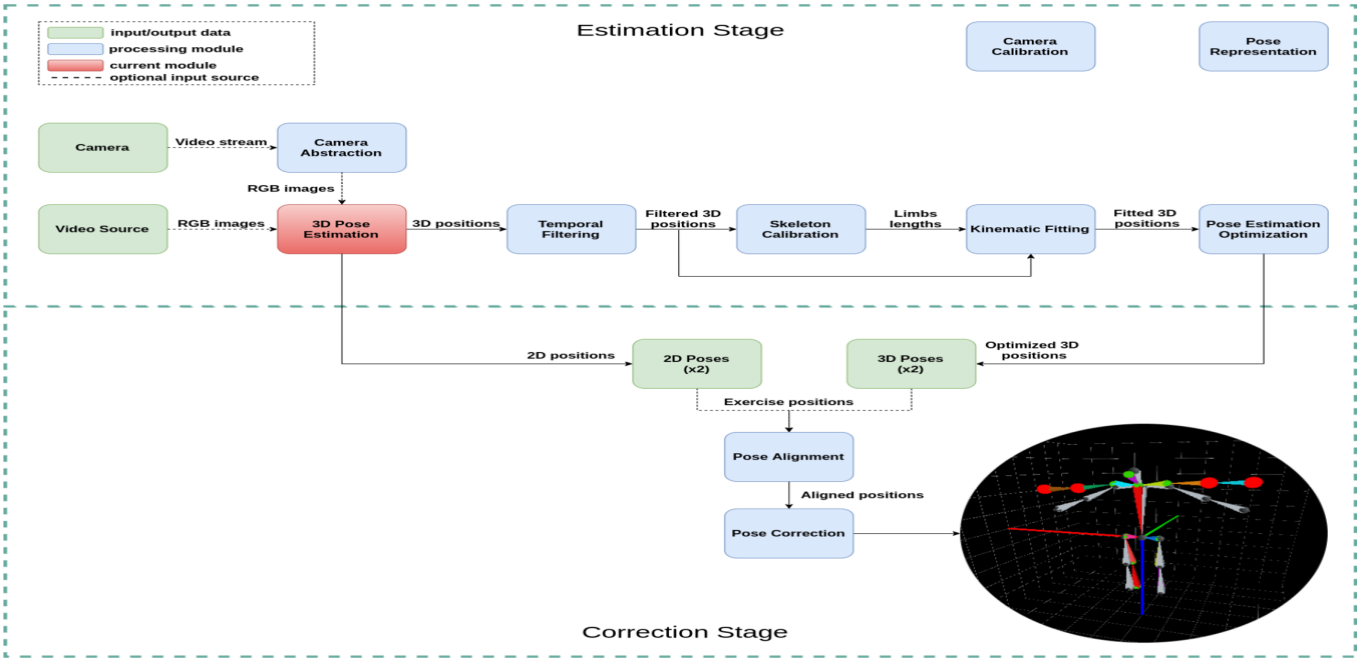


Figure 6.1 Workflow of Pose Estimation using Tensorflow.

The workflow of online testing is shown in Figure 6.1. The strategy proposed on this solution to the 3D pose estimation problem consists of using a neural network to predict the 2D position of the human body joints on each of the pictures captured by different cameras. Then, by using the corresponding camera calibration parameters, the system triangulates each joint's three-dimensional position on the world coordinate system.

To estimate the 2D position of the body joints on the input camera frames, the ILovePose's implementation of Fast Human Pose Estimation was used. After the 2D pose estimation library processes the input images, the estimated 2D position of the body joints are triangulated by the 3D reconstruction module developed using some of the tools provided by OpenCV.

Testing Properties

Testing properties refer to what to test in the testing process and for what conditions testing needs to guarantee for a trained model. The following are some typical properties of the proposed model.

- Correctness: Correctness measures the probability of the proposed system under test.
- Data Privacy: Data Privacy in machine learning is the proposed system's ability to preserve private data information.

- Efficiency: The efficiency of a machine learning system refers to its construction or prediction speed.
- Security: The security of a machine learning system is the system's resilience against potential harm, danger, or loss made via manipulating or illegally accessing ML components.
- Robustness: 'The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions'.

Accuracy

Accuracy is a metric for the performance analysis of any classification model. We can calculate accuracy by dividing the number of correct predictions with the total number of predictions [9]. The below Table II shows the parameters for prediction. Accuracy is the fraction of prediction. It determines the number of correct predictions over the total number of predictions made by the model. The formula of accuracy is:

$$\text{Accuracy} = \text{number of correct predictions} / \text{total number of predictions}$$

It has been observed that the number of correct predictions of the proposed system are eight thousand six (8764) out of ten thousand cases (10000).

$$\text{Accuracy} = 0.8764.$$

6.2 Output Screens

The following are screenshots of the coding part of the proposed system. Figure 6.2 shows the coding part for importing libraries.

```
In [1]: import cv2 as cv
        import matplotlib.pyplot as plt

In [2]: net = cv.dnn.readNetFromTensorflow("graph_opt.pb")

In [3]: inWidth = 368
        inHeight = 368
        thr = 0.2

In [4]: BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
                       "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
                       "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "REye": 14,
                       "LEye": 15, "REar": 16, "LEar": 17, "Background": 18 }

        POSE_PAIRS = [ ["Neck", "RShoulder"], ["Neck", "LShoulder"], ["RShoulder", "RElbow"],
                       ["RElbow", "RWrist"], ["LShoulder", "LElbow"], ["LElbow", "LWrist"],
                       ["Neck", "RHip"], ["RHip", "RKnee"], ["RKnee", "RAnkle"], ["Neck", "LHip"],
                       ["LHip", "LKnee"], ["LKnee", "LAnkle"], ["Neck", "Nose"], ["Nose", "REye"],
                       ["REye", "REar"], ["Nose", "LEye"], ["LEye", "LEar"] ]
```

Figure 6.2 Screenshot Showing the Libraries Imported in Proposed System.

```
In [8]: def pose_estimation(frame):
            frameWidth = frame.shape[1]
            frameHeight = frame.shape[0]
            net.setInput(cv.dnn.blobFromImage(frame, 1.0, (inWidth, inHeight), (127.5, 127.5, 127.5), swapRB=True, crop=False))
            out = net.forward()
            out = out[:, :19, :, :]  # MobileNet output [1, 57, -1, -1], we only need the first 19 elements

            assert(len(BODY_PARTS) == out.shape[1])

            points = []
            for i in range(len(BODY_PARTS)):
                # Slice heatmap of corresponging body's part.
                heatMap = out[0, i, :, :]

                # Originally, we try to find all the local maximums. To simplify a sample
                # we just find a global one. However only a single pose at the same time
                # could be detected this way.
                _, conf, _, point = cv.minMaxLoc(heatMap)
                x = (frameWidth * point[0]) / out.shape[3]
                y = (frameHeight * point[1]) / out.shape[2]
                # Add a point if it's confidence is higher than threshold.
                points.append((int(x), int(y)) if conf > thr else None)

            for pair in POSE_PAIRS:
                partFrom = pair[0]
                partTo = pair[1]
                assert(partFrom in BODY_PARTS)
                assert(partTo in BODY_PARTS)

                idFrom = BODY_PARTS[partFrom]
                idTo = BODY_PARTS[partTo]

                if points[idFrom] and points[idTo]:
                    cv.line(frame, points[idFrom], points[idTo], (0, 255, 0), 3)
                    cv.ellipse(frame, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv.FILLED)
                    cv.ellipse(frame, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv.FILLED)
```

Figure 6.3 Screenshot of building joints in an image.

Figure 6.3 shows the coding part for building joints of a model in python using Jupyter notebook platform and anaconda prompt. As shown in Figure 6.3 we can observe the running status of command.

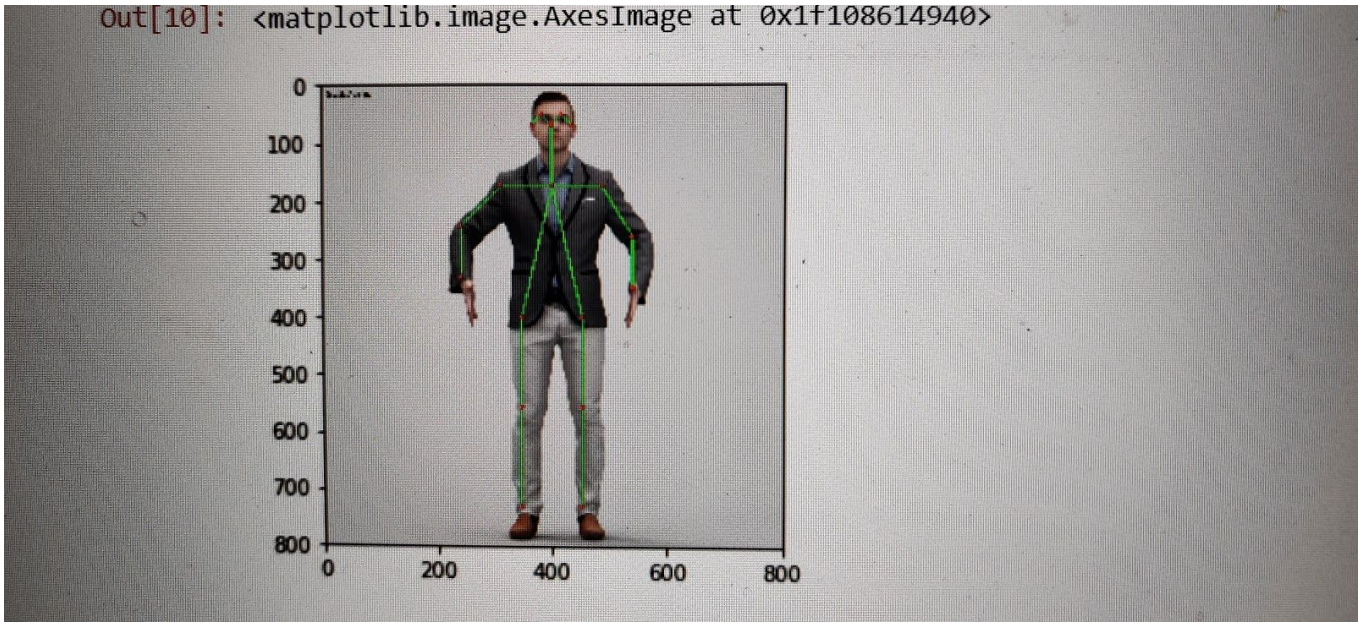


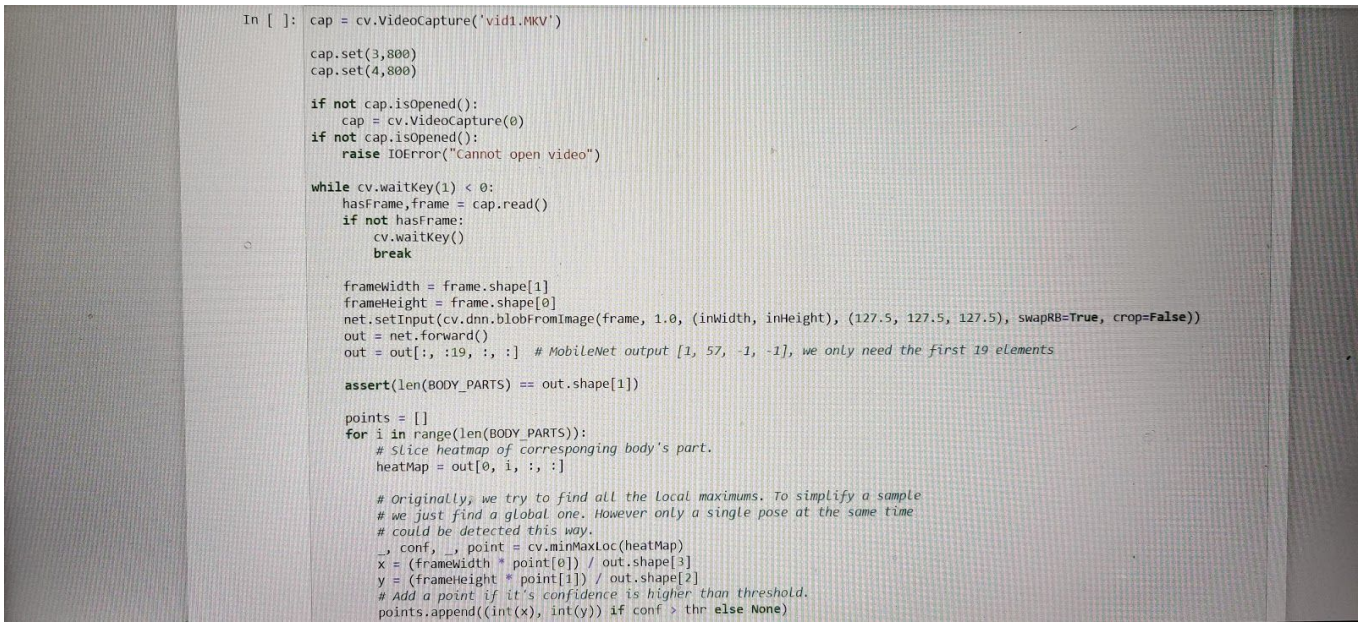
Figure 6.4 Output of estimation of pose in an image.

Figure 6.4 shows the output of the proposed system. The accuracy percentage and test error percentage of estimation is shown in Figure 6.4. It has been observed that the proposed model gives an efficient estimation of pose which results in 87 percent accuracy and error of 13 percent approximately.

Fig 6.5 Screenshot of Output of multiple persons

The fig 6.5 shown above shows the pose estimation of multiple people.If the human pose estimation system uses video records as a data source, key points (joints locations) are detected

from a sequence of frames, not a single picture. It allows us to achieve more accuracy as the system analyzes an actual movement of a person, not a steady position.



```
In [ ]: cap = cv.VideoCapture('vid1.MKV')

        cap.set(3,800)
        cap.set(4,800)

        if not cap.isOpened():
            cap = cv.VideoCapture(0)
        if not cap.isOpened():
            raise IOError("Cannot open video")

        while cv.waitKey(1) < 0:
            hasFrame,frame = cap.read()
            if not hasFrame:
                cv.waitKey()
                break

            frameWidth = frame.shape[1]
            frameHeight = frame.shape[0]
            net.setInput(cv.dnn.blobFromImage(frame, 1.0, (inWidth, inHeight), (127.5, 127.5, 127.5), swapRB=True, crop=False))
            out = net.forward()
            out = out[:, :19, :, :]  # MobileNet output [1, 57, -1, -1], we only need the first 19 elements

            assert(len(BODY_PARTS) == out.shape[1])

            points = []
            for i in range(len(BODY_PARTS)):
                # Slice heatmap of corresponging body's part.
                heatMap = out[0, i, :, :]

                # Originally, we try to find all the local maximums. To simplify a sample
                # we just find a global one. However only a single pose at the same time
                # could be detected this way.
                _, conf, _, point = cv.minMaxLoc(heatMap)
                x = (frameWidth * point[0]) / out.shape[3]
                y = (frameHeight * point[1]) / out.shape[2]
                # Add a point if it's confidence is higher than threshold.
                points.append((int(x), int(y)) if conf > thr else None)
```

Figure 6.5 Screenshot of building joints in a video.

Figure 6.5 shows the building joints of pose estimation in a video which later will be implemented.After analysis and practical experience of working with 3D human pose estimation systems, we have come to our own vision of how it can be implemented.

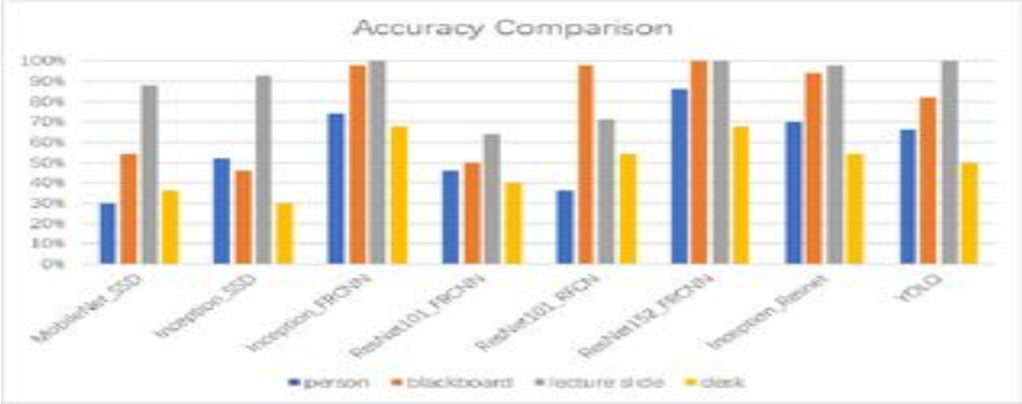


Figure 6.6 Accuracy Comparison of Eight Groups of Experiments.

As shown in Figure 6.6 we can observe different accuracies on pose estimation. The experiments are conducted using SSD,CNN, YOLQ. When having the positions of keypoints (joints) extracted, they should be compared with the reference video's positions.

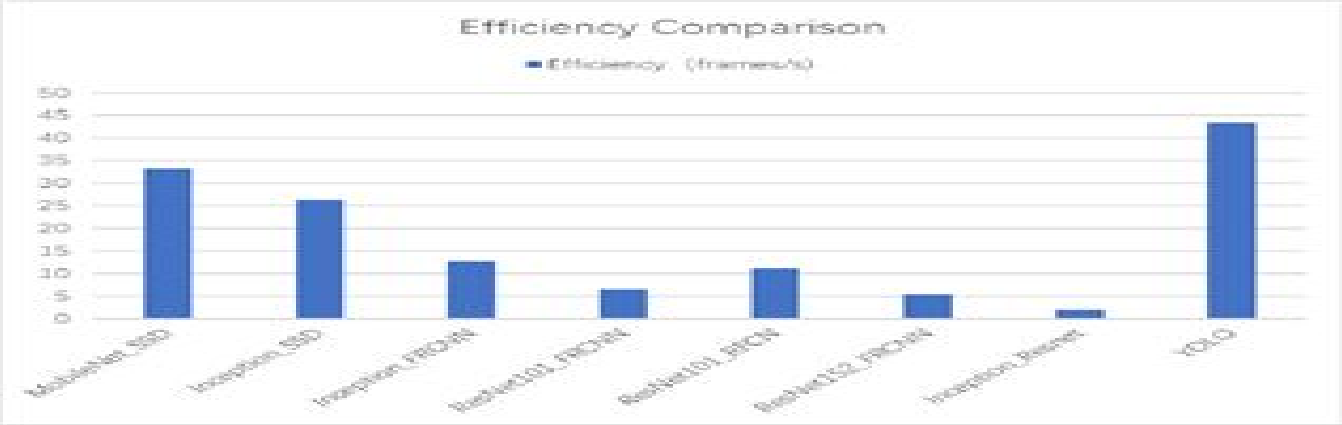


Figure 6.7 Efficiencies of Eight Groups of Experiments.

As shown in Figure 6.7 different efficiencies are compared because it illustrates well how it might work by example. But the flow might be changed depending on business requirements or other factors.

7. CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

Pose Estimation uses several existing detection algorithms based on deep learning and classification models based on CNN to evaluate the teaching state of teachers. From the experimental results, it is clear that the angle of the classroom in the picture and illumination have a certain impact on the detection. And when the classroom is large and the platform is far away from the camera, the detection is not easy too. In eight sets of experiments, the Resnet152 model based on the Faster R-CNN detection algorithm has the highest accuracy. The Inception-v2 model based on Faster R-CNN is second. But the efficiency of the two is very low, it is difficult to meet real-time requirements. The YOLO model is very efficient, although the accuracy rate is lower than the previous two, but the efficiency comparison accuracy loss is still worth a certain degree. In subsequent work, the parameters are adjusted according to the performance of different sets of experiments to achieve better performance. Models will be improved and algorithms to ensure high accuracy while improving efficiency. It can also be tried to use the OpenPose method to improve our algorithm.

7.2 Future Scope

It is suggested to further improve the model reported in this study using more variables. Detection of people has long been a major focus for a range of industries, including security, business intelligence, health and safety, and entertainment. With recent developments in machine-learning algorithms, the accuracy of these detections, and the hardware requirements to run them, pose estimation has now reached a point where it has become commercially viable.

BIBLIOGRAPHY

[1] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(1):44–58, 2006.

[2] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[3] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[4] M. Bergtholdt, J. Kappes, S. Schmidt, and C. Schnorr. A study of parts based object class detection using complete graphs. International Journal of Computer Vision, 2010.

[5] L. Bo and C. Sminchisescu. Twin gaussian processes for structured prediction. International Journal of Computer Vision, 2010.

[6] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas. Fast algorithms for large scale conditional 3d prediction. In IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[7] M. Eichner and V. Ferrari. We are family: Joint pose estimation of multiple persons. In European Conference on Computer Vision, 2010.

[8] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. International Journal of Computer Vision, 61(1):55–79, 2005.

[9] V. Ferrari, M. J. Mark-Jimnez, and A. Zisserman. Progressive search space reduction for human pose estimation. In IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[10] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel. Optimization and filtering for human motion capture. International Journal of Computer Vision, 87(1– 2):75–92, 2010.

[11] H. Jiang. Human pose estimation using consistent max-covering. In IEEE International Conference on Computer Vision, 2009.

[12] A. Kanaujia, C. Sminchisescu, and D. Metaxas. Semi-supervised hierarchical models for 3d human pose reconstruction. In IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[13] M. W. Lee and I. Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In IEEE Conference on Computer Vision and Pattern Recognition, 2004.

APPENDIX

```
/* Pose Estimation Using Tensorflow */
# Jupyter notebook

# COMMAND ----------
#packages imported
import cv2 as cv
import matplotlib.pyplot as plt



# COMMAND ----------
#Usage of Tensorflow library and mentioning parts to estimate
net = cv.dnn.readNetFromTensorflow("graph_opt.pb")
inWidth = 368
inHeight = 368
thr = 0.2
BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
            "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
            "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "REye": 14,
            "LEye": 15, "REar": 16, "LEar": 17, "Background": 18 }

POSE_PAIRS = [ ["Neck", "RShoulder"], ["Neck", "LShoulder"], ["RShoulder",
"RElbow"],
            ["RElbow", "RWrist"], ["LShoulder", "LElbow"], ["LElbow", "LWrist"],
            ["Neck", "RHip"], ["RHip", "RKnee"], ["RKnee", "RAnkle"], ["Neck", "LHip"],
            ["LHip", "LKnee"], ["LKnee", "LAnkle"], ["Neck", "Nose"], ["Nose", "REye"],
            ["REye", "REar"], ["Nose", "LEye"], ["LEye", "LEar"] ]



# COMMAND ----------
#code for displaying an image
img = cv.imread("image.jpg")
```

```
plt.imshow(img)
plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))


# COMMAND ----------
#code for estimating the pose in a image
def pose_estimation(frame):
    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]
    net.setInput(cv.dnn.blobFromImage(frame, 1.0, (inWidth,
inHeight),      (127.5, 127.5, 127.5), swapRB=True, crop=False))
    out = net.forward()
    out = out[:, :19, :, :]  # MobileNet output [1, 57, -1, -1], we only
need the first 19 elements assert(len(BODY_PARTS) == out.shape[1])
   points = []
    for i in range(len(BODY_PARTS)):
        # Slice heatmap of corresponding body's part.
        heatMap = out[0, i, :, :]


        # Originally, we try to find all the local maximums. To simplify
sample
        # we just find a global one. However only a single pose at the
same time
        # could be detected this way.
        _, conf, _, point = cv.minMaxLoc(heatMap)
        x = (frameWidth * point[0]) / out.shape[3]
        y = (frameHeight * point[1]) / out.shape[2]
        # Add a point if it's confidence is higher than threshold.
        points.append((int(x), int(y)) if conf > thr else None)
    for pair in POSE_PAIRS:
        partFrom = pair[0]
        partTo = pair[1]
        assert(partFrom in BODY_PARTS)
```

```
            assert(partTo in BODY_PARTS)
            idFrom = BODY_PARTS[partFrom]
            idTo = BODY_PARTS[partTo]
              if points[idFrom] and points[idTo]:
                cv.line(frame, points[idFrom], points[idTo], (0, 255, 0), 3)
                cv.ellipse(frame, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255),
      cv.FILLED)
                cv.ellipse(frame, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255),
cv.FILLED)
           t, _ = net.getPerfProfile()
          freq = cv.getTickFrequency() / 1000
          cv.putText(frame, '%.2fms' % (t / freq), (10, 20),
cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
          return frame
      estimated_image = pose_estimation(img)
      plt.imshow(cv.cvtColor(estimated_image, cv.COLOR_BGR2RGB))



      # COMMAND ----------
      #code to estimate pose in a video
      cap = cv.VideoCapture('vid1.MKV')
       cap.set(3,800)
      cap.set(4,800)
      if not cap.isOpened():
          cap = cv.VideoCapture(0)
      if not cap.isOpened():
          raise IOError("Cannot open video")
      while cv.waitKey(1) < 0:
          hasFrame,frame = cap.read()
          if not hasFrame:
              cv.waitKey()
              break
```

```
        frameWidth = frame.shape[1]
        frameHeight = frame.shape[0]
        net.setInput(cv.dnn.blobFromImage(frame, 1.0, (inWidth,
inHeight), (127.5, 127.5, 127.5), swapRB=True, crop=False))
        out = net.forward()
        out = out[:, :19, :, :]  # MobileNet output [1, 57, -1, -1], we
only need the first 19 elements
            assert(len(BODY_PARTS) == out.shape[]
            points = []
        for i in range(len(BODY_PARTS)):
            # Slice heatmap of corresponding body's part.
            heatMap = out[0, i, :, :]

            # Originally, we try to find all the local maximums. To
simplify a sample
            # we just find a global one. However only a single pose at
the same time
            # could be detected this way.
            _, conf, _, point = cv.minMaxLoc(heatMap)
            x = (frameWidth * point[0]) / out.shape[3]
            y = (frameHeight * point[1]) / out.shape[2]
            # Add a point if it's confidence is higher than threshold.
            points.append((int(x), int(y)) if conf > thr else None)

        for pair in POSE_PAIRS:
            partFrom = pair[0]
            partTo = pair[1]
            assert(partFrom in BODY_PARTS)
            assert(partTo in BODY_PARTS)

            idFrom = BODY_PARTS[partFrom]
            idTo = BODY_PARTS[partTo]
```

```
            if points[idFrom] and points[idTo]:
                cv.line(frame, points[idFrom], points[idTo], (0, 255, 0),
3)
                cv.ellipse(frame, points[idFrom], (3, 3), 0, 0, 360, (0, 0,
255), cv.FILLED)
                cv.ellipse(frame, points[idTo], (3, 3), 0, 0, 360, (0, 0,
255), cv.FILLED)

        t, _ = net.getPerfProfile()
        freq = cv.getTickFrequency() / 1000
        cv.putText(frame, '%.2fms' % (t / freq), (10, 20),
cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))

        cv.imshow('pose estimation Tutorial ',frame)


    # COMMAND ----------
    # printing data for i in
AccidentSplitRDD.take(20): print(i)
 cap = cv.VideoCapture(1)
cap.set(cv.CAP_PROP_FPS, 10)
cap.set(3,800)
cap.set(4,800)
if not cap.isOpened():
   cap = cv.VideoCapture(0)
if not cap.isOpened():
   raise IOError("Cannot open webcam")
while cv.waitKey(1) < 0:
   hasFrame,frame = cap.read()
   if not hasFrame:
      cv.waitKey()
```

```
        break
    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]
    net.setInput(cv.dnn.blobFromImage(frame, 1.0, (inWidth, inHeight), (127.5, 127.5, 127.5),
swapRB=True, crop=False))
    out = net.forward()
    out = out[:, :19, :, :]  # MobileNet output [1, 57, -1, -1], we only need the first 19 elements
 assert(len(BODY_PARTS) == out.shape[1])
points = []
    for i in range(len(BODY_PARTS)):
        # Slice heatmap of corresponding body's part.
        heatMap = out[0, i, :, :]

        # Originally, we try to find all the local maximums. To simplify a sample
        # we just find a global one. However only a single pose at the same time
        # could be detected this way.
        _, conf, _, point = cv.minMaxLoc(heatMap)
        x = (frameWidth * point[0]) / out.shape[3]
        y = (frameHeight * point[1]) / out.shape[2]
        # Add a point if it's confidence is higher than threshold.
        points.append((int(x), int(y)) if conf > thr else None)

    for pair in POSE_PAIRS:
        partFrom = pair[0]
        partTo = pair[1]
        assert(partFrom in BODY_PARTS)
        assert(partTo in BODY_PARTS)
        idFrom = BODY_PARTS[partFrom]
        idTo = BODY_PARTS[partTo]
          if points[idFrom] and points[idTo]:
           cv.line(frame, points[idFrom], points[idTo], (0, 255, 0), 3)
           cv.ellipse(frame, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv.FILLED)
```

```
            cv.ellipse(frame, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv.FILLED)
    t, _ = net.getPerfProfile()
    freq = cv.getTickFrequency() / 1000
    cv.putText(frame, '%.2fms' % (t / freq), (10, 20), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0,
0))
    cv.imshow('pose estimation Tutorial ',frame)
```