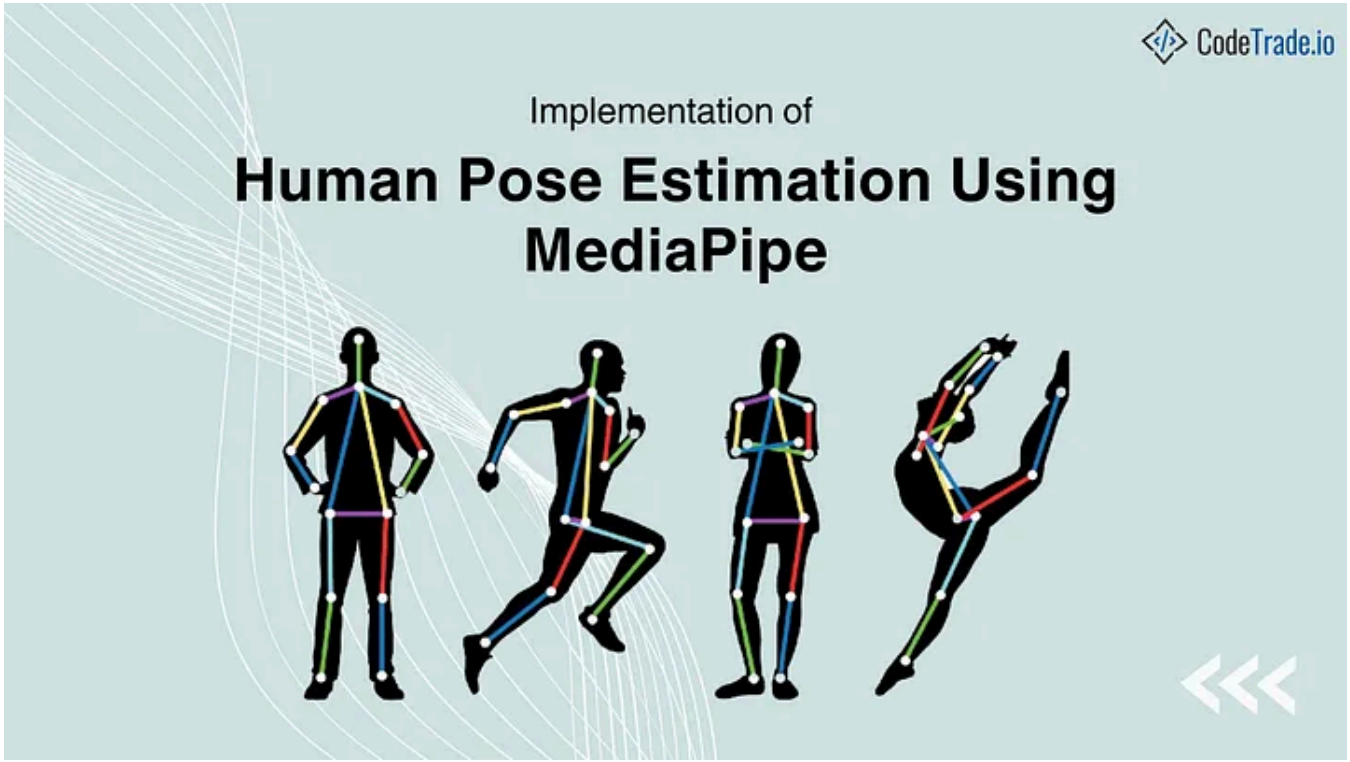
Implementation of Human Pose Estimation Using MediaPipe

5 min read · Jul 23, 2024

CodeTrade India Follow

Listen Share

As the **artificial intelligence and machine learning** landscape evolves, there are many tools and frameworks available for developers. Among these, Mediapipe stands out as a comprehensive solution for various computer vision tasks, including human pose estimation.



Pose estimation is the process of detecting and tracking human body landmarks in images and videos. It is a powerful tool that has a wide range of applications, such as fitness, gaming, augmented reality, and medical imaging.

Here we'll delve into the intricacies of human pose estimation and demonstrate how to implement it using mediapipe.

What is Human Pose Estimation?

Human pose estimation is a computer vision technique that detects and tracks the positions of key points on a human body from an image or video. These key points typically include joints such as elbows, knees, shoulders, and ankles.

By understanding the human body posture and movement, applications can be developed for diverse fields such as sports analytics, healthcare, animation, and augmented reality.

> *Explore: Type of Pose Estimation Models*

What is MediaPipe and Why it is Used?

An open-source, cross-platform machine learning framework called MediaPipe offers a range of options for problems like pose estimation, face detection, and hand tracking.

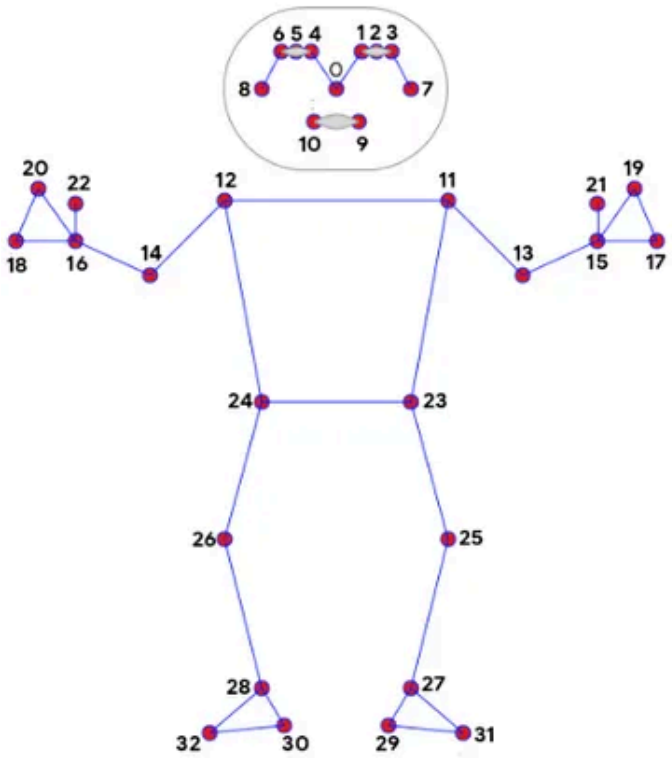
The head, shoulders, elbows, wrists, hands, hips, knees, and ankles are just a few of the 33 3D landmarks on the human body that MediaPipe stance can identify to estimate a person's stance.

> *Explore More: MediaPipe Vs. TensorFlow: Battle Of The Human Pose Estimation Giants*

Here are some reasons why Mediapipe is a preferred choice:

- **Cross-Platform Support:** Mediapipe supports deployment across multiple platforms, including Android, iOS, web, and desktop.

- **Real-Time Performance:** It offers high-performance, real-time capabilities, making it suitable for applications requiring immediate feedback.
- **Ease of Use:** With its modular and extensible design, Mediapipe allows developers to customize and integrate various components effortlessly.
- **Pre-Trained Models:** Mediapipe comes with pre-trained models, which significantly reduces the complexity and time required to develop pose estimation solutions.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

MediaPipe Pose is easy to use and can be deployed on a variety of platforms, including mobile devices, desktop computers, and servers. It is also relatively lightweight and efficient, making it suitable for real-time applications.

Steps to Implement Human Pose Estimation Using MediaPipe

MediaPipe is a framework for real-time and offline pose estimation tasks. It provides a user-friendly API and high-performance models. Let's dive into the steps involved:

Step 1: Installation

We need to install the MediaPipe library using the pip command as follows:

```
$ pip install mediapipe
```

```
Collecting mediapipe
  Downloading mediapipe-0.10.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (33.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33.5/33.5 MB 47.2 MB/s eta 0:00:00
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from mediapipe) (1.4.0)
Requirement already satisfied: attrs>=19.1.0 in /usr/local/lib/python3.10/dist-packages (from mediapipe) (23.1.0)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from mediapipe) (23.5.26)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from mediapipe) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from mediapipe) (1.23.5)
Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.10/dist-packages (from mediapipe) (4.8.0.76)
Requirement already satisfied: protobuf<4,>=3.11 in /usr/local/lib/python3.10/dist-packages (from mediapipe) (3.20.3)
Collecting sounddevice>=0.4.4 (from mediapipe)
  Downloading sounddevice-0.4.6-py3-none-any.whl (31 kB)
Requirement already satisfied: CFFI>=1.0 in /usr/local/lib/python3.10/dist-packages (from sounddevice>=0.4.4->mediapipe) (1.15.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mediapipe) (2.8.2)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.21)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->mediapipe) (1.16.0)
Installing collected packages: sounddevice, mediapipe
Successfully installed mediapipe-0.10.5 sounddevice-0.4.6
```

For the implementation purpose, we will use **google colab**, you can also use your favorite Editor as well.

Step 2: Installation Verification

Print the version of the library to verify if the library is installed or not. Use the given command:

```
import mediapipe as mp
mp.__version__
```

Output:

```
'0.10.5'
```

Step 3: Import Necessary Libraries

Use the given code to import necessary libraries for human pose estimation.

```
#for pose detection
import mediapipe as mp
```

```
#for image input output operations
import matplotlib.pyplot as plt
import matplotlib.image as pimg
```

Step 4: Create Pose Estimation Object

To initialize the pose estimation object we will call the pose class from **mediapipe.solutions.pose** and create an object named **mp_pose**. You can replace the class name as per your requirements.

```
mp_pose = mp.solutions.pose.Pose(min_detection_confidence=0.7,
min_tracking_confidence=0.7)
#where the min_detection_conficence and min_tracking_confidence are the minimum
```

Step 5: Read the image and process for detecting the pose

To read the image using matplotlib use cv2, PIL, or any other library.

```
img = pimg.imread('/content/test_image.jpg')
```

The original image is:

```
plt.imshow(img)

#detecting the object using mediapipe
results = mp_pose.process(img)
```

- **Y:** the y coordinate of the landmark (key point) in the image given
- **Z:** the z coordinate of the landmark (key point) in the image given
- **Visibility:** a float value from 0 to 1 which indicates the visibility of a landmark.

View all landmarks:

```
print(results.pose_landmarks)
```

```
print(results.pose_landmarks)
```

```
landmark {
  x: 0.5276347398757935
  y: 0.23079752922058105
  z: -0.9188101291656494
  visibility: 0.999996542930603
}
landmark {
  x: 0.5429335832595825
  y: 0.2165457010269165
  z: -0.8730435967445374
  visibility: 0.9999927282333374
}
landmark {
  x: 0.5510205030441284
  y: 0.2170158177614212
  z: -0.8731083273887634
  visibility: 0.9999910593032837
}
landmark {
  x: 0.5595657229423523
  y: 0.21751096844673157
  z: -0.8732666373252869
  visibility: 0.9999903440475464
}
landmark {
  x: 0.517665445804596
  y: 0.21497708559036255
  z: -0.8755286335945129
  visibility: 0.9999938011169434
}
```

Once we have detected the landmarks we can perform many tasks using those landmarks. Let's draw a skeleton using those landmarks:

Draw a Skeleton Using Landmarks

Initialize the drawing class of MediaPipe:

```
mp_drawing = mp.solutions.drawing_utils
```

Draw the connections of the landmarks using `draw_landmarks()` function:

```
mp_drawing.draw_landmarks(
    img,
    results.pose_landmarks,
    mp.solutions.pose.POSE_CONNECTIONS,
    mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=4, circle_radius=2),
    mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=6, circle_radius=2)
```

```
)
plt.imshow(img)
```

This takes arguments as:

- **Image:** the image whose pose landmarks were detected.
- **Landmarks:** the list of landmarks.
- **Connections:** type of connections.
- **Landmark_drawing_spec:** attributes of landmarks points drawing.
- **Connection_drawing_spec:** attributes of connection lines

Let's take an example.

Examples of Real-Time Pose Estimation Implementation

You can also implement real-time pose estimation using video.

Code

```
import cv2
import mediapipe as mp

mp_pose = mp.solutions.pose.Pose(min_detection_confidence=0.7, min_tracking_cor
mp_drawing = mp.solutions.drawing_utils
cap = cv2.VideoCapture('video_path')

while True:
    ret, frame = cap.read()
    if not ret:
        break

    results = mp_pose.process(frame)
    mp_drawing.draw_landmarks(
        frame,
        results.pose_landmarks,
        mp.solutions.pose.POSE_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=4, circle_radius=2)
        mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=6, circle_radius=2)
    )
    cv2.imshow("Frame", frame)

    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Output

Wrapping Up

MediaPipe Pose is a robust and versatile tool for pose estimation. Its ease of use and cross-platform compatibility make it ideal for various applications, including gaming, fitness, augmented reality, and medical imaging.

By leveraging MediaPipe, developers can create innovative solutions that accurately track and analyze human body movements.

Follow CodeTrade for More or visit www.codetrade.io

Human Pose Estimation | Pose Estimation | Mediapipe | Artificial Intelligence | AI

Written by CodeTrade India

96 followers · 11 following

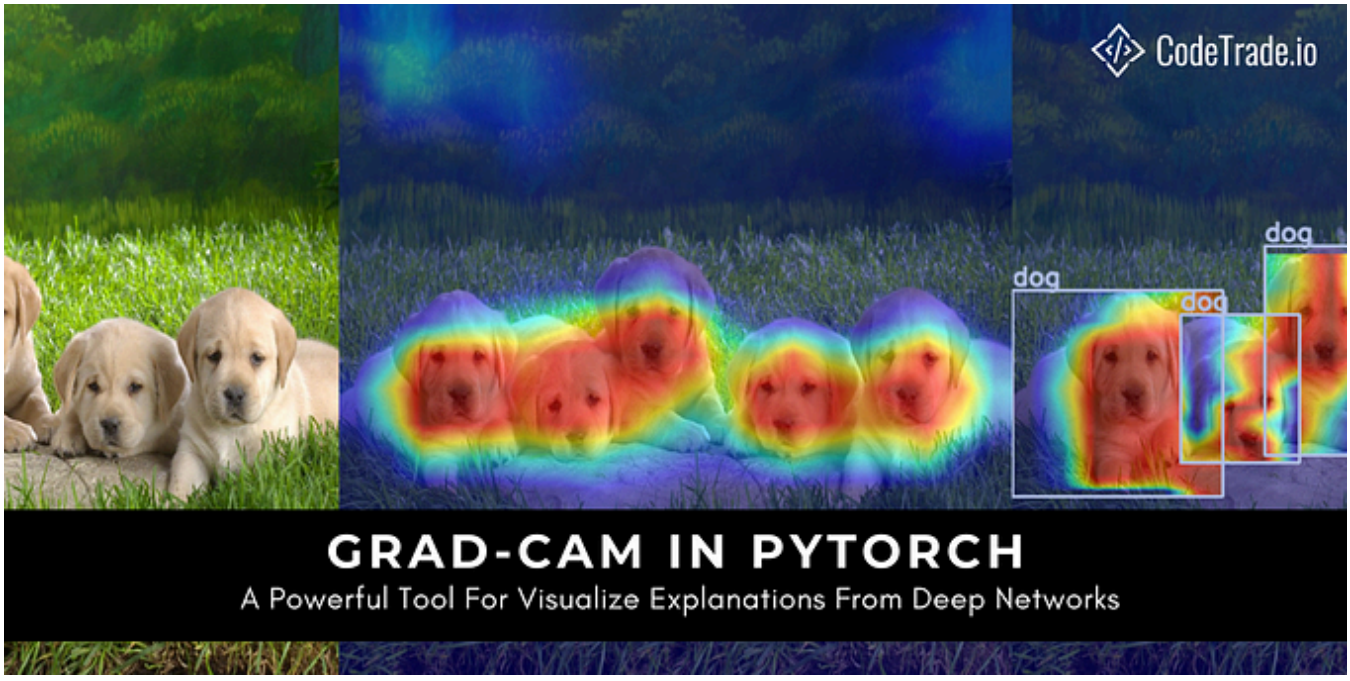
Top-notch custom software development service provider | Open edX Development | Odoo Development Agency | AI & ML Development | Mobile App Development

No responses yet

Write a response

What are your thoughts?

More from CodeTrade India

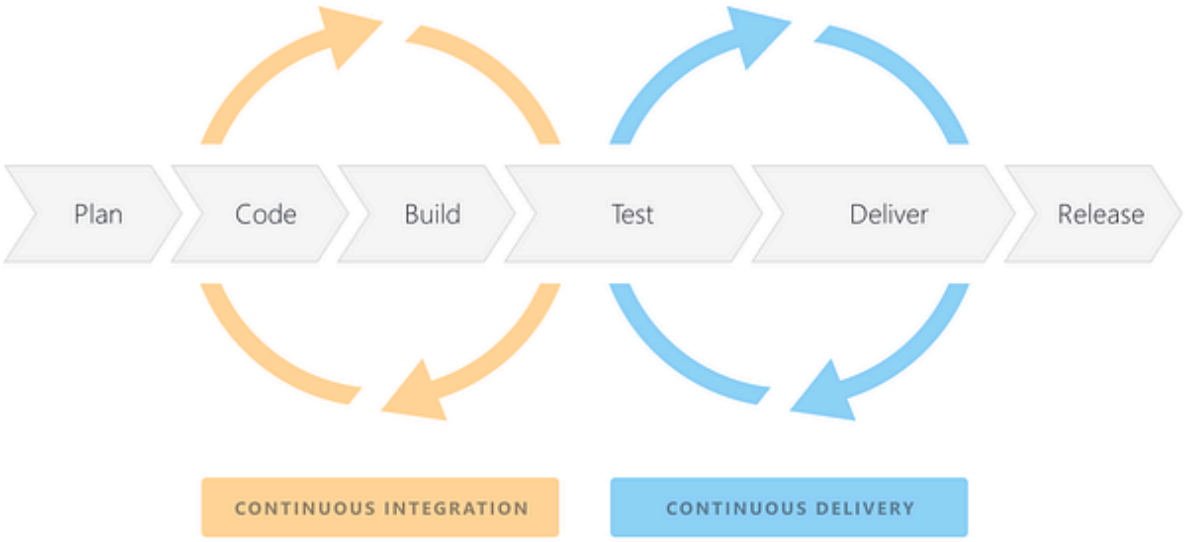


CodeTrade India

Grad-CAM In PyTorch: A Powerful Tool For Visualize Explanations From Deep Networks

In the realm of deep learning, understanding the decision-making process of neural networks is crucial, especially when it comes to...

Jun 6, 2024 251

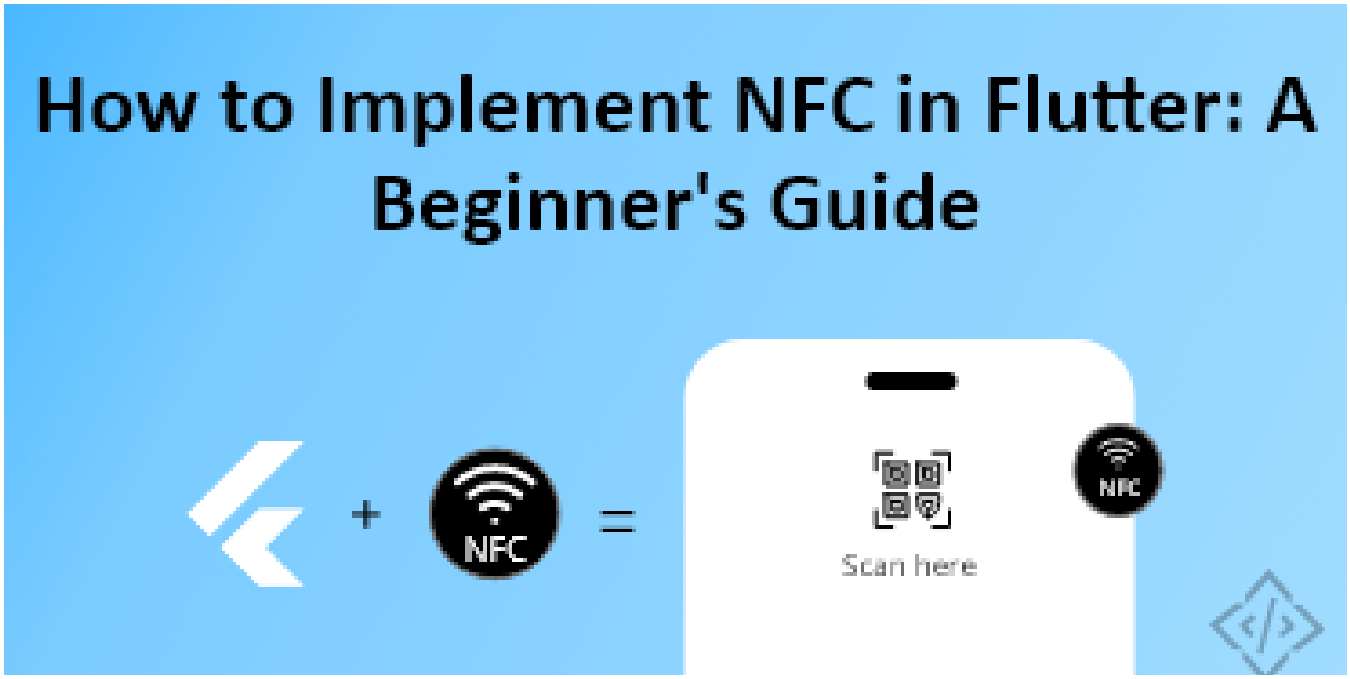


CodeTrade India

Step-by-Step Guide To Add Upload testflight Using Fastlane

Are you a Flutter developer or iOS developer, and you're tired of uploading ipa files on testflight with the manual approach and looking...

Feb 28, 2024 233 1

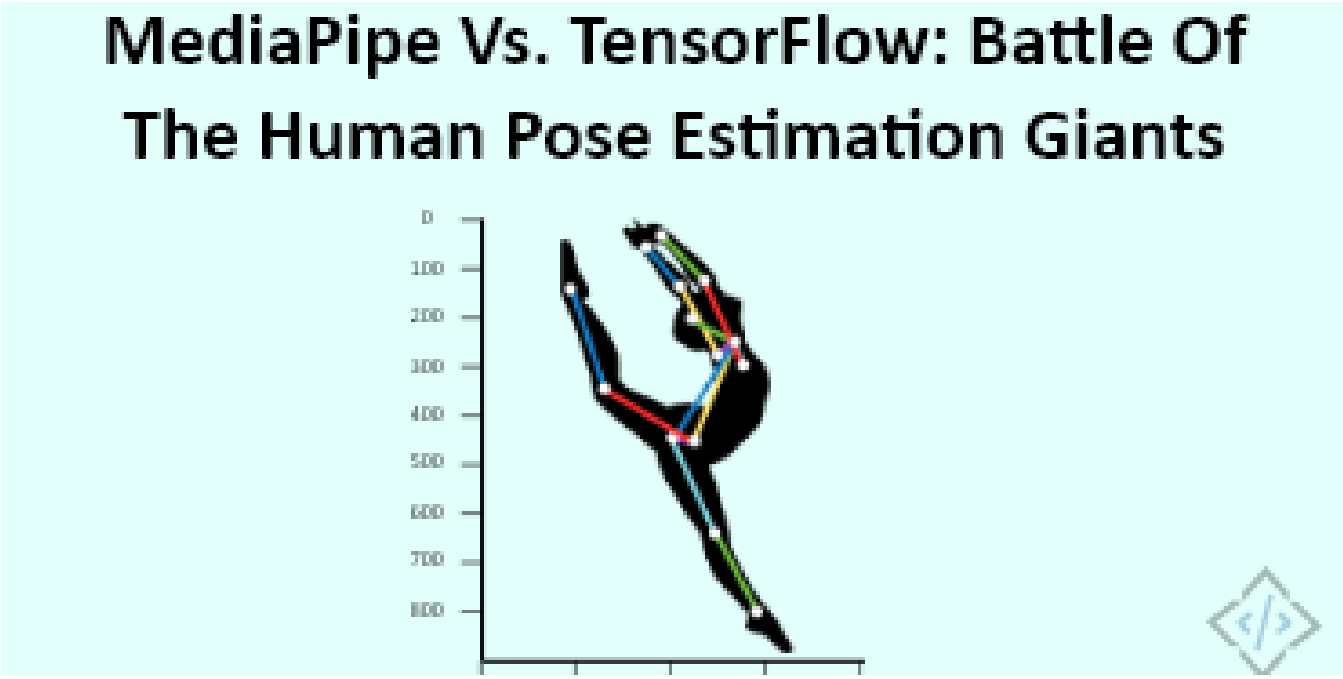


CodeTrade India

Implement NFC in Flutter To Transfer Peer-to-peer Data

Near-field communication (NFC) is a wireless technology that allows devices to exchange data over short distances. Used in a variety of...

Dec 11, 2023 251 2



CodeTrade India

MediaPipe Vs. TensorFlow: Human Pose Estimation Giants

MediaPipe and TensorFlow are not exactly competitors; rather, they can complement each other in various ways. TensorFlow is a deep learning...

Jan 30, 2024 182

See all from CodeTrade India

Recommended from Medium

飞桨 PaddleO

In Generative AI by Rakib.ai

Unleash the Power of PaddleOCR: Your Guide to Best Open Source OCR

Want to find out about the best OCR that you can use to build AI applications at scale and earn a fortune!

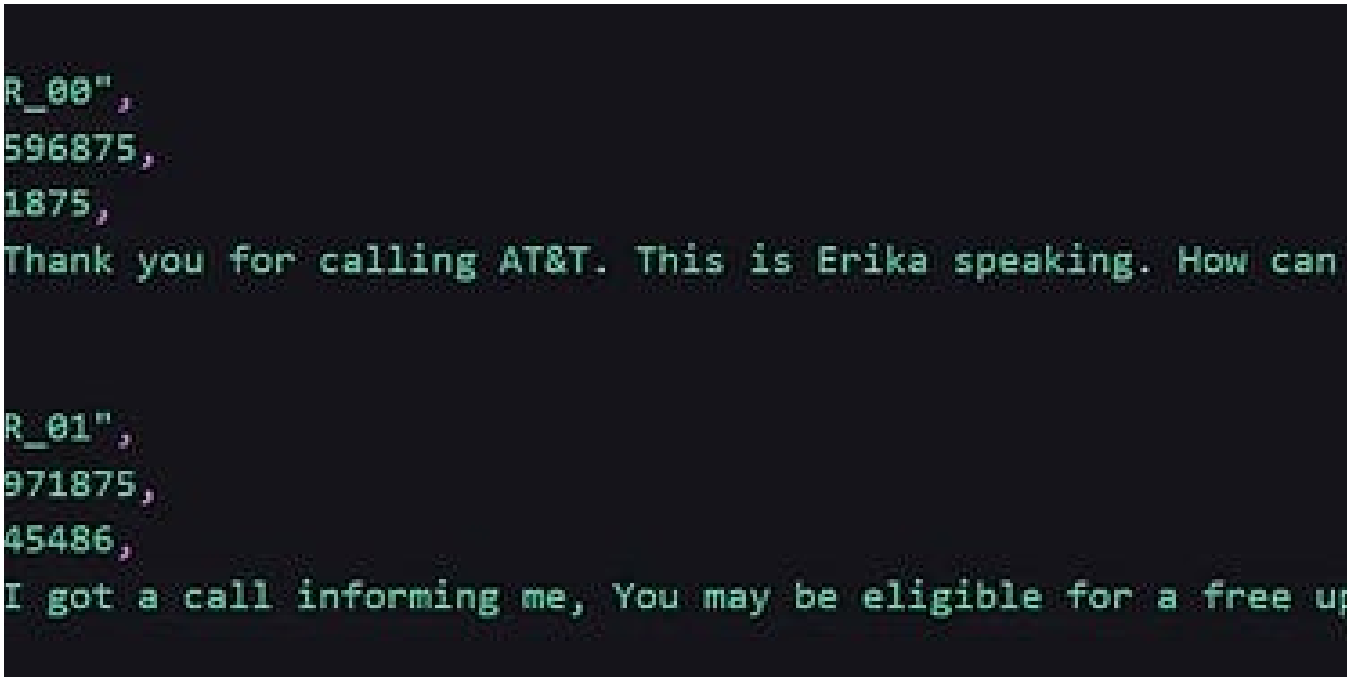
Feb 8 206

cheng zhang

Recent Research on Pose Detection Models: BlazePose, MoveNet and More

In a recent project requiring pose detection, I researched models including BlazePose and MoveNet. Below is a detailed comparison.

Mar 15 12

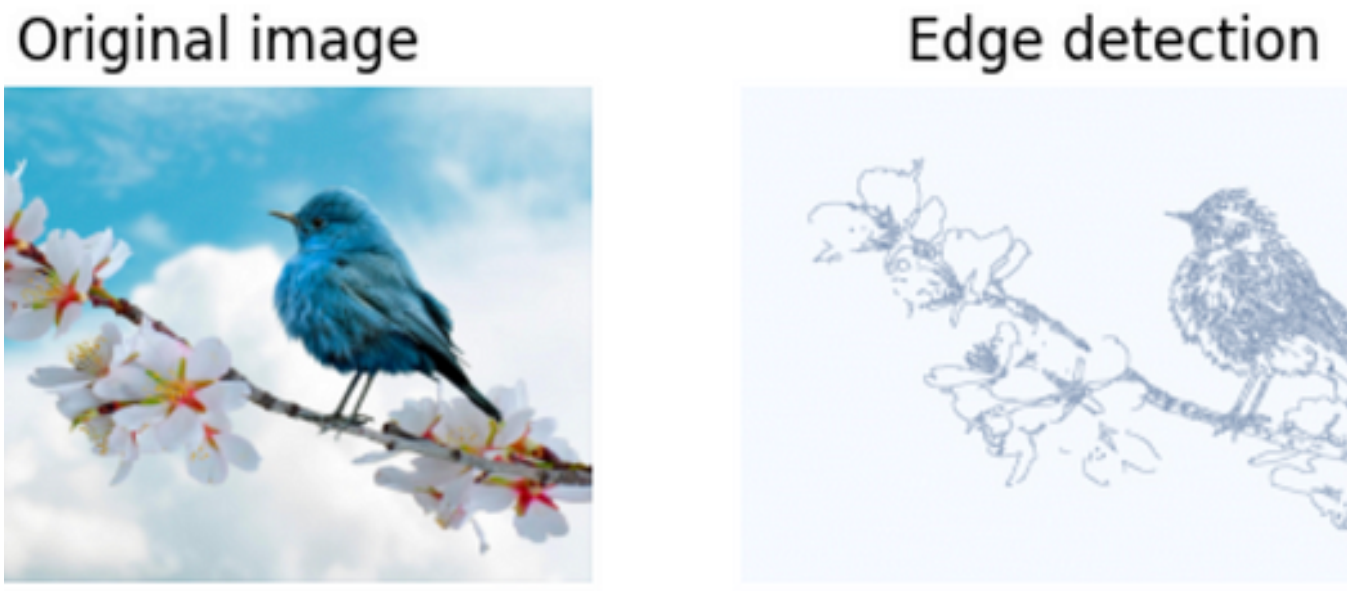


Adi Insights and Innovations

AI-Driven Customer Call Insights: From Audio to Action

Customer support calls hold a wealth of information, but finding the time to manually comb through these recordings for insights isn't...

Jul 1



In The ByteDoodle Blog by Hareesha Dandamudi

Canny Edge Detection

Image Processing with OpenCV

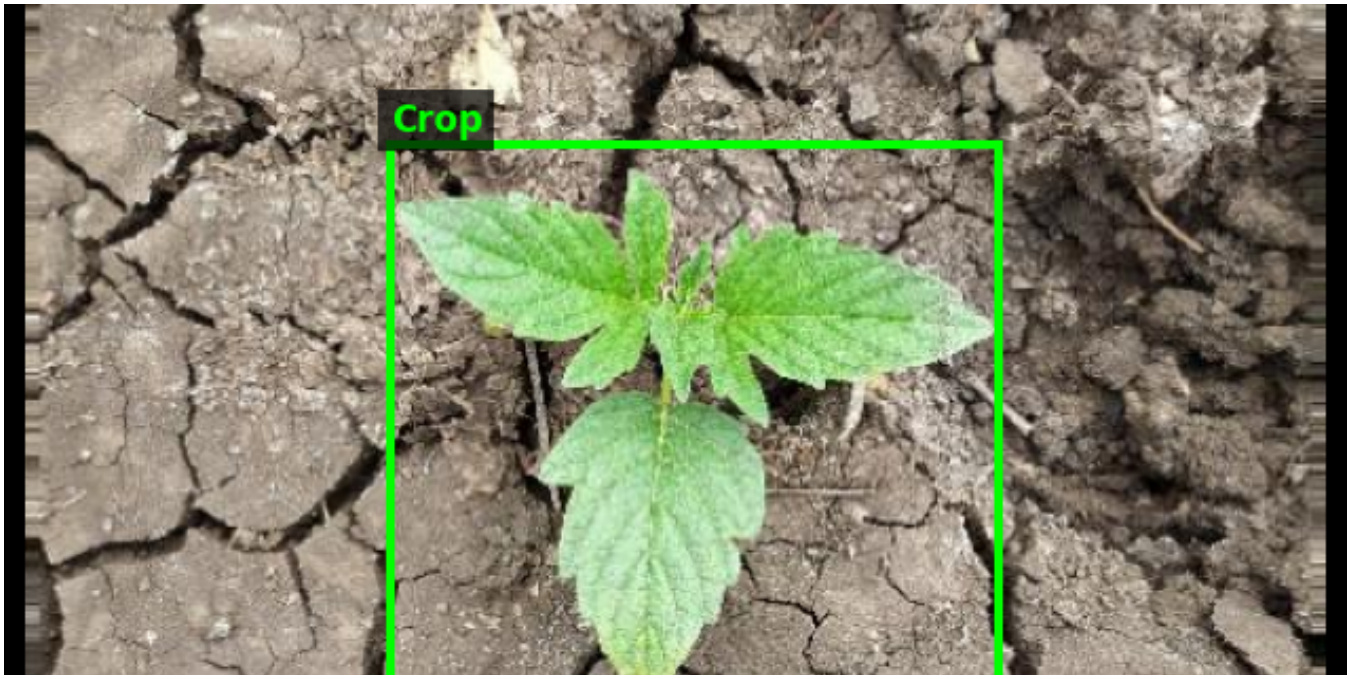
Apr 1

In Predict by iswarya writes

GPT-5 Is Coming in July 2025—And Everything Will Change

“It’s wild watching people use ChatGPT… knowing what’s coming.” —OpenAI insider

Jul 8 8.7K 292



Brian Hulela

Understanding Image Annotation Formats for Object Detection

A Beginner’s Guide to YOLO, COCO, Pascal VOC, and More

Apr 1