

Mr. Pose: An Easy Guide for Pose Estimation with Google's MediaPipe

9 min read · Oct 31, 2022



Loges Siva

Follow



Listen



Share

Learn to estimate human poses for exercises using MediaPipe by Google

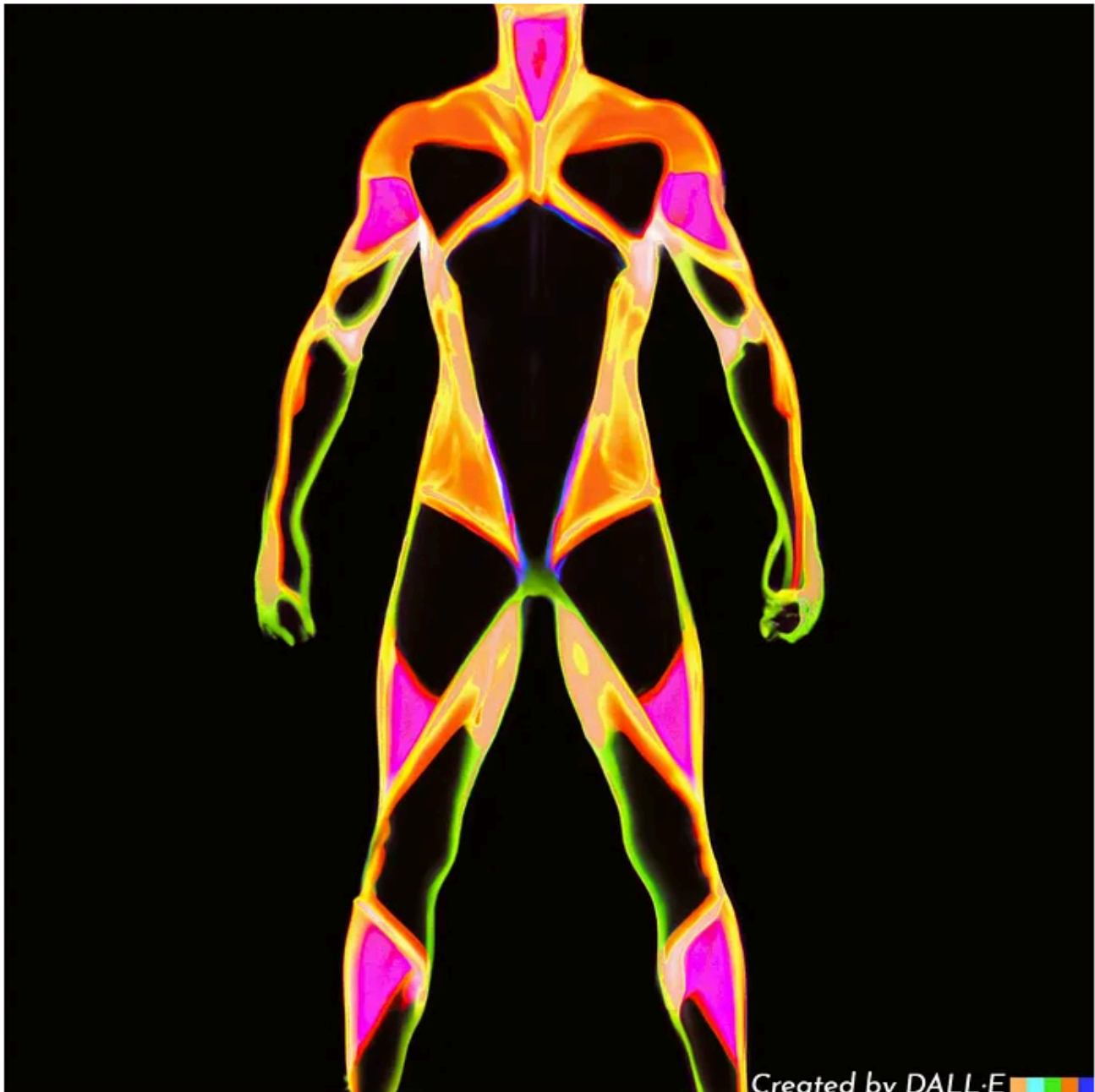


Image of a human pose generated by DALL.E

Hi, let's go over a short and informative read on pose estimation for exercises using the Mediapipe library by Google. This post is for developers interested in Deep Learning and Computer Vision or who want to integrate pose estimation solutions into their projects. This post will not go into the math, architecture, and research behind Mediapipe library. If you're interested to learn the details, the links are attached in the references section.

Introduction 📄

In this article we will go through a pose estimation solution called "[Mr. Pose](#)" that leverages the MediaPipe library and math algorithms to estimate exercise.

Mr. Pose is a visual analytics application that helps humans to track the accuracy of exercise, and count repetitions or predict the exercise performed. MediaPipe lib provides a landmark model “BlazePose” that predicts the location of 33 landmarks in the human body. Mr. Pose uses the location and movement of the landmarks across the frames of video and exercises estimation algorithms to predict and track

Open in app ↗

Sign up

Sign in

Medium

🔍 Search



model for exercise prediction, it provides high throughput performance and memory efficiency. This advantage ensures a high frame rate in low-powered and low-memory hardware.

Mr. Pose application supports four different exercise predictions and measurements as follows,

- Pushup
- Plank
- Squat
- Jumping Jack

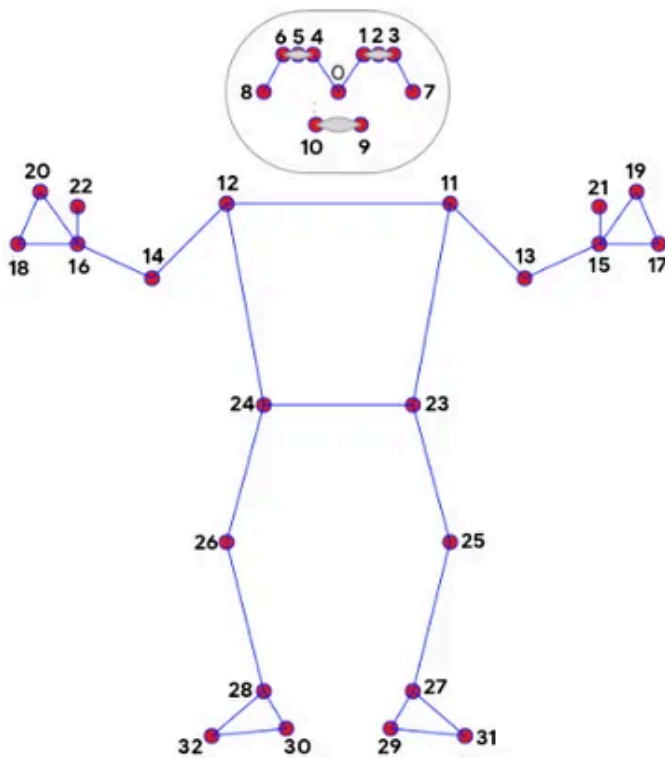
Google's MediaPipe

MediaPipe is a cross-platform, open-source library developed by Google providing cutting-edge ML solutions. The library is designed to work across platforms like iOS, Linux, Windows, and Android on high to low-powered hardware like Raspberry Pi. Some of the solutions provided by the library are Face Detection, Iris Detection, Human Pose Estimation, and Instant motion tracking. Refer to the [MediaPipe page](#) for solutions supported in different languages and platforms.

Simple python code snippet to initialize the Pose model and get results,

```
import mediapipe as mp

mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
results = pose.process(image)
```



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

33 pose landmarks. Source: MediaPipe Pose

Algorithms for Pose Prediction

Before we get into the algorithms for exercise prediction and measurement, let's go over some basic functions essential for the algorithms.

(1) Angle between two lines with one common point

Consider two lines A and B. Let line A be defined between points point1, and point2. Let line B be defined between points point2, and point3. Then, the angle between the two lines A and B with a common point point2 is calculated as follows,

```

1  def angle(self, point1, point2, point3):
2      """ Calculate angle between two lines """
3      if(point1==(0,0) or point2==(0,0) or point3==(0,0)):
4          return 0
5      numerator = point2[1] * (point1[0] - point3[0]) + point1[1] * \
6                  (point3[0] - point2[0]) + point3[1] * (point2[0] - point1[0])
7      denominator = (point2[0] - point1[0]) * (point1[0] - point3[0]) + \
8                   (point2[1] - point1[1]) * (point1[1] - point3[1])
9      try:
10         ang = math.atan(numerator/denominator)
11         ang = ang * 180 / math.pi
12         if ang < 0:
13             ang = 180 + ang
14         return ang
15     except:
16         return 90.0

```

two_line_angle.py hosted with ❤ by GitHub

[view raw](#)

Angle between two lines with one common point

(2) Angle between a line and horizontal

Consider a line between points point1, and point2. Then, the angle between the line and the horizontal (X-axis) is calculated as follows,

```

1  def angle_of_singleline(self, point1, point2):
2      """ Calculate angle of a single line """
3      x_diff = point2[0] - point1[0]
4      y_diff = point2[1] - point1[1]
5      return math.degrees(math.atan2(y_diff, x_diff))

```

one_line_angle.py hosted with ❤ by GitHub

[view raw](#)

Angle between a line and horizontal

(3) Euclidean distance between two points

Consider two points point1, point2. Then, the euclidean distance between the points is calculated as follows,

```

1  def dist_xy(self, point1, point2):
2      """ Euclidean distance between two points point1, point2 """
3      diff_point1 = (point1[0] - point2[0]) ** 2
4      diff_point2 = (point1[1] - point2[1]) ** 2
5      return (diff_point1 + diff_point2) ** 0.5

```

euclidean_distance.py hosted with ❤ by GitHub

[view raw](#)

(4) Point position from a line

Consider a point and a line. The position of the point (i.e. right or left) from the line is calculated as follows,

```

1  def point_position(self, point, line_pt_1, line_pt_2):
2      """
3      Left or Right position of the point from a line
4      """
5      value = (line_pt_2[0] - line_pt_1[0]) * (point[1] - line_pt_1[1]) - \
6              (line_pt_2[1] - line_pt_1[1]) * (point[0] - line_pt_1[0])
7      if value >= 0:
8          return "left"
9      return "right"

```

point_position.py hosted with ❤ by GitHub

[view raw](#)

Position of a point from a line

Pushup

To predict the exercise performed in the video is pushup, we focus on Shoulder, Hip, and Ankle landmarks (key points) in the human body.

For pushups, the following conditions are expected to be met,

1. The angle between Shoulder-Hip line and Hip-Ankle line is expected to be close to horizontal. That is, close to 0° or 180° depending on the position (left or right) of the person performing the exercise. This indicates that the person is positioned parallel to the ground.
2. The angle of line Shoulder-Ankle or Hip-Ankle are expected to be parallel to horizontal or at least close to it.
3. The angle between Shoulder-Elbow line and Elbow-Wrist line is tracked across the frames.
4. The angle of line Elbow-Wrist from horizontal is tracked across the frames.
5. The above two conditions are tracked for 24 continuous frames in video using a counter. This is to avoid any false positives in measurement.
6. After 24 continuous incrementation of counter, if the mean of angle from 4th condition for 24 frames is far from 90° and mean of difference between 1st, 12th

and 24th frame angle from 3rd condition is greater than 5 (a small constant value), then the prediction is **Pushup**.

Plank

To predict the exercise performed in the video is a plank, we focus on similar landmarks and conditions as the pushup algorithm. The key difference is the 6th condition.

After 24 continuous incrementation of counter, if the mean of angle from 4th condition for 24 frames is closer to 90° and mean of difference between 1st, 12th and 24th frame angle from 3rd condition is less than 5 (a small constant value), then the prediction is **Plank**.

Squat

To predict the exercise performed in the video is a squat, we focus on Head, Hand, Foot, Hip, and Knee landmarks (key points) in the human body. Head point can be one of the available points: Nose, Left ear, Right ear, Left eye, or Right eye. Hand point can be one of the available points: Left wrist, Right wrist, Left pinky, Right pinky, Left index, or Right index. Foot point can be one of the available points: Left foot index, Right foot index, Left heel, Right heel, Left ankle, or Right ankle.

For squat, the following conditions are expected to be met,

1. The Y-coordinate (i.e. height) of the head point is tracked across the frames.
2. The angle of Shoulder-Ankle line or Hip-Ankle line from horizontal is expected to be close to 90° .
3. The angle of Knee-Ankle line from horizontal is expected to be close to 90° .
4. The angle of line Hip-Knee from horizontal is expected to be close to 90° .
5. The mean height of head point from 1st condition for 24 frames is calculated. The mean value is normalized to the person height in the video based on head point and foot point as minval and maxval respectively.
6. If the normalized height from previous condition is less than 0 after 24 continuous frames, then it indicates that the person is doing a downward motion in the video.

7. All previous exercises (Pushup and Plank) conditions are expected to be not met. Then, the prediction is **Squat**.

Jumping Jack

To predict the exercise performed in the video is jumping jack, we focus on Head point and Hand point similar to the Squat exercise.

Get Loges Siva's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

Subscribe

For jumping jack, the following conditions are expected to be met,

1. The difference between Y-coordinates (i.e. height) of the head point and hand point are expected to be greater than 0. This indicates that the hand is above head position.
2. The angle of Shoulder-Ankle line or Hip-Ankle line from horizontal is expected to be close to 90° . This indicates that the person is standing.
3. All previous exercises (Pushup, Plank and Squat) conditions are expected to be not met. Then, the prediction is **jumping jack**.

Algorithms for Pose Measurement

Pushup

To measure the pushup exercise performed in the video, we focus on Head, Ankle, and Wrist points.

For one pushup repetition, the following conditions are expected to be met,

1. Get position of the Head point from the vertical line (at $[\text{frame width}/2, \text{frame height}/2]$) in the middle of the frame.
2. Calculate the angle between Head-Ankle line and Ankle-Wrist line. If the position of Head point is right, then the angle is expected to be closer to 0° . Else

if the position of Head point is left, then the angle is expected to be closer to 180° .

3. Calculate the Y-coordinate distance between Head point and Ankle point. If the distance is less than 250 (a constant value) and previous condition is met, set pushup flag to True.
4. If pushup flag is True and distance calculated by previous condition is greater than 300 (a constant value), increment the pushup counter and set pushup flag to False.
5. If the previous condition is met, then it indicates that one full pushup is complete. Repeat the same process for entire video to count all repetitions.

Pushups measurement result video

Plank

To measure the plank exercise performed in the video, we focus on Shoulder, Elbow, Wrist, Hip, and Ankle points.

For plank, the following conditions are expected to be met,

1. The angle between Shoulder-Hip line and Hip-Ankle line is expected to be close to horizontal. That is, close to 0° or 180° depending on the position (left or right) of the person performing the exercise.

2. The angle of line Shoulder-Ankle or Hip-Ankle are expected to be parallel to horizontal or at least close to it.
3. The angle between Shoulder-Elbow line and Elbow-Wrist line is tracked across the frames.
4. The angle of line Elbow-Wrist from horizontal is tracked across the frames.
5. The above two conditions are tracked for 24 continuous frames in video using a counter. This is to avoid any false positives in measurement.
6. After 24 continuous incrementation of counter, if the mean of angle from (4)th condition for 24 frames is far from 90° and mean of difference between 1st, 12th and 24th frame angle from (3)rd condition is greater than 5 (a small constant value), the plank timer (in HH:MM:SS format) is started.
7. If the previous condition fails, the plank timer is stopped till the condition is met once again in subsequent frames.

Plank measurement result video

Squat

To measure the squat exercise performed in the video, we focus on Head and Ankle points.

For one squat repetition, the following conditions are expected to be met,

1. Calculate the Y-coordinate distance between Head point and Ankle point. Normalize the distance with 0 and height of frame as minval and maxval respectively.
2. If the normalized distance is less than 0.5, the squat flag is set to True.
3. If the squat flag is True and normalized distance is greater than 0.5, the squat counter is incremented. The squat flag is set to False.
4. If the previous condition is met, then it indicates that one full squat is complete. Repeat the same process for entire video to count all the repetitions.

Squats measurement result video

Jumping Jack

To measure the jumping jack exercise performed in the video, we focus on Head, Hand, Shoulder, Ankle, and Hip points.

For one jumping jack repetition, the following conditions are expected to be met,

1. Calculate the Y-coordinate distance between Head point and Hand point. Normalize the distance with 0 and height of frame as minval and maxval respectively.
2. Calculate the angle of Shoulder-Ankle line or Hip-Ankle line from horizontal.

3. If the normalized distance is greater than 0 and angle is closer to 90°, the jumping jack flag is set to True.
4. If the jumping jack flag is True and normalized distance is less than 0, the jumping jack counter is incremented. Jumping jack flag is set to False.
5. If the previous condition is met, then it indicates that one full jumping jack is complete. Repeat the same process for entire video to count all the repetitions.

Mr. Pose application | Jumping Jacks measurement

shorts Input video: [https://www.pexels.com/video/elderly-man-doing-jumping-jacks-outside-7299359/Repositories:\[1\]...](https://www.pexels.com/video/elderly-man-doing-jumping-jacks-outside-7299359/Repositories:[1]...)

www.youtube.com

General Requirements 🧑

Follow the [readme](#) general requirements section in the repository and place the camera for live video or record a video of a person performing the exercise.

Code Requirements 🧑

Follow the [readme](#) code requirements section in the repository and set up the environment. This is an essential part of the forth-coming section of this article.

How to Run 🏃

To run the Mr. Pose application, clone the [repository](#), install the requirements and run following command,

```
python mrpose.py --video <path to video file> --exercise <exercise to be measured>
```

Optional Arguments:

— *video* : Path to video source file.

If argument is not provided, then Mr.Pose will launch webcam for live video.

Currently, live webcam video works only for exercise prediction.

— *exercise* : Choices are pushup, plank, squat, jumpingjack

If argument is not provided, then Mr.Pose will predict the exercise performed in the video. If argument is provided, then Mr.Pose will measure the mentioned exercise.

Conclusion 📄

Mr. Pose applications perform exercise prediction and measurement based on key-point positions, the angle between lines, and the distance between points using simple mathematics and algorithms. This application can be extended to many different exercises or other applications (like fall detection, walking, or running) using similar concepts.

Thanks to Google developers for open source MediaPipe library. Thanks to the creators of the BlazePose model which is used directly by Mr. Pose application through the library.

Hope this article was informative and detailed in explaining the use case of the MediaPipe library through an application. For any feedback or queries, please post them in the comments.

Happy Learning! 😊

References 📌

- [1] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, Matthias Grundmann, “MediaPipe: A Framework for Building Perception Pipelines”, arXiv:1906.08172, 2019
- [2] Logeswaran Sivakumar, “Mr.Pose”, <https://github.com/Logeswaran123/MrPose>, 2022
- [3] [MediaPipe Pose](#), Google

Mlearning.ai Submission Suggestions

How to become a writer on Mlearning.ai

medium.com

Pose Estimation

Mediapipe

Deep Learning

Computer Vision

ML So Good



Follow

Written by Loges Siva

30 followers · 2 following

Machine Learning Engineer

No responses yet



Write a response

What are your thoughts?


More from Loges Siva

Understanding and Exploring 3D Gaussian Splatting: A Comprehensive Overview

Blending Pixels and Realities with no compromise in Speed

Dec 28, 2023  84  1



 Maximilian Vogel

The ChatGPT list of lists: A collection of 3000+ prompts, GPTs, use-cases, tools, APIs, extensions...

Updated Feb-16, 2025. Added New Introductions, Prompts, Lists and Tools

Feb 8, 2023  13.6K  191





Kishan Modasiya

What the heck is random_state?

random_state = 0 or 42 or None

Jun 25, 2022



537



4



Loges Siva

Playground for Stable Diffusion

An Easy Guide to using Stable Diffusion model for image and video generation

Oct 16, 2022




30



See all from Loges Siva

Recommended from Medium



 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





 In TIER IV MEDIA by TIER IV

High-performance SAM2 inference framework with TensorRT

Keywords: SAM2, Instance Segmentation, TensorRT, ONNX Runtime, Optimization

Jun 25  2



 In Predict by iswarya writes

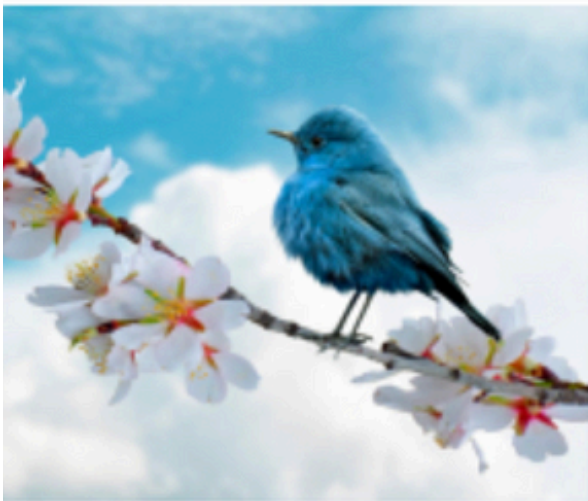
GPT-5 Is Coming in July 2025—And Everything Will Change

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

★ Jul 8 🖱 8.7K 💬 291



Original image



Edge detection



 In The ByteDoodle Blog by Hareesha Dandamudi

Canny Edge Detection

Image Processing with OpenCV

★ Apr 1





In ImageCraft by Francisco Zavala

Image histogram analysis

Histogram Calculation Explained: Counting Pixel Frequencies Across channel Levels.



Mar 13

👏 2



In Python in Plain English by Tarik Emre Yorulmaz

Streamlining Raster Data Reclassification: A Python Workflow with Rasterio and NumPy

Raster reclassification is a fundamental operation in Geographic Information Systems (GIS) and remote sensing. It involves changing the...

★ Jul 12 🤝 1



See more recommendations