**Project Documentation: Hand Tracking with Google Generative AI Integration**

**Project Title: Real-time Hand Tracking and Math Problem Solving Using Google Generative AI**

**File Name: main.py**

**Overview**

This project integrates real-time hand tracking using OpenCV with the Google Generative AI model (Gemini-1.5-flash) to solve math problems drawn by hand on a virtual canvas. The system captures input from a webcam, detects hand gestures, and allows the user to draw on the screen. Specific hand gestures trigger the AI to interpret and solve the drawn math problems.

**Requirements**

* **Python Libraries**:
  + cvzone: For easier hand detection and tracking.
  + cv2 (OpenCV): For capturing video frames and drawing on the canvas.
  + numpy: For creating and manipulating arrays (used for the canvas).
  + google.generativeai: For interacting with Google's generative AI model.
  + PIL (Pillow): For handling image conversions.
* **Hardware**:
  + A webcam (to capture video feed).

**Setup**

1. **Install Required Libraries**:

pip install cvzone opencv-python numpy google-generativeai pillow

1. **Google Generative AI API Key**:
   * Configure the API key from Google Generative AI:
   * genai.configure(api\_key="YOUR\_API\_KEY")
2. **Configure the Webcam**:
   * The code initializes the webcam using OpenCV:

cap = cv2.VideoCapture(0)

cap.set(3, 1280) # Set width

cap.set(4, 720) # Set height

**Main Components**

1. **Hand Detection**:
   * **HandDetector** from cvzone is used to detect and track hand landmarks in real-time.
   * The hand information, including the list of 21 landmarks and the state of the fingers (up/down), is obtained through the getHandInfo function.
2. **Drawing on Canvas**:
   * The draw function is responsible for drawing lines on a blank canvas based on the index finger's position (lmList1[8]).
   * The line is drawn when the gesture [0, 1, 0, 0, 0] is detected (only the index finger is up).
   * A gesture with all fingers up [1, 1, 1, 1, 1] clears the canvas.
3. **Sending Image to AI**:
   * When the gesture [1, 1, 1, 1, 0] (all fingers except the thumb up) is detected, the current state of the canvas is converted to an image and sent to the Google Generative AI model to interpret and solve the problem.
   * The AI response is printed to the console.

**Code Walkthrough**

**1. Import Libraries**

python

Copy code

import cvzone

import cv2

import numpy as np

from cvzone.HandTrackingModule import HandDetector

import google.generativeai as genai

from PIL import Image

**2. Configure the Generative AI Model**

python

Copy code

genai.configure(api\_key="YOUR\_API\_KEY")

model = genai.GenerativeModel('gemini-1.5-flash')

**3. Initialize Webcam and Hand Detector**

python

Copy code

cap = cv2.VideoCapture(0)

cap.set(3, 1280)

cap.set(4, 720)

detector = HandDetector(staticMode=False, maxHands=1, modelComplexity=1, detectionCon=0.9, minTrackCon=0.5)

**4. Detect Hand Information**

python

Copy code

def getHandInfo(img):

hands, img = detector.findHands(img, draw=False, flipType=True)

if hands:

hand1 = hands[0]

lmList1 = hand1["lmList"]

fingers = detector.fingersUp(hand1)

return fingers, lmList1

else:

return None

**5. Draw on the Canvas**

python

Copy code

def draw(info, prev\_pos, canvas):

fingers, lmList1 = info

current\_pos = None

if fingers == [0, 1, 0, 0, 0]:

current\_pos = lmList1[8][0:2]

if prev\_pos is None:

prev\_pos = current\_pos

cv2.line(canvas, current\_pos, prev\_pos, (255, 0, 255), 10)

elif fingers == [1, 1, 1, 1, 1]:

canvas = np.zeros\_like(img)

return current\_pos, canvas

**6. Send Canvas to AI for Problem Solving**

python

Copy code

def sendToAI(model, canvas, fingers):

if fingers == [1, 1, 1, 1, 0]:

pil\_image = Image.fromarray(canvas)

response = model.generate\_content(["Solve the Math Problem ", pil\_image])

print(response.text)

**7. Main Loop**

python

Copy code

prev\_pos = None

canvas = None

while True:

success, img = cap.read()

img = cv2.flip(img, 1)

if canvas is None:

canvas = np.zeros\_like(img)

info = getHandInfo(img)

if info:

fingers, lmlist1 = info

print(fingers)

prev\_pos, canvas = draw(info, prev\_pos, canvas)

sendToAI(model, canvas, fingers)

image\_combines = cv2.addWeighted(img, 0.65, canvas, 0.35, 0)

cv2.imshow("image\_combines", image\_combines)

cv2.waitKey(1)

**Usage**

* **Draw on the Canvas**: Hold up only the index finger to draw.
* **Clear the Canvas**: Hold up all five fingers.
* **Solve the Math Problem**: Hold up four fingers (excluding the thumb) to send the drawn image to the AI for solving.

**Conclusion**

This project demonstrates the integration of real-time computer vision with a generative AI model to interactively solve math problems. It serves as a foundation for more advanced applications in human-computer interaction and AI-driven problem-solving.