Absolutely! Let's convert your Streamlit MNIST digit recognizer into a **Flask app** that:

✅ Uses the same trained mnist\_ann\_model.h5  
✅ Lets users **draw digits via HTML5 canvas**  
✅ Preprocesses the image correctly  
✅ Predicts and displays the digit + confidence bar chart

### 🧠 Flask App Structure:

flask\_mnist\_app/

├── app.py

├── static/

│ └── style.css (optional)

├── templates/

│ └── index.html

├── mnist\_ann\_model.h5

### 1️⃣ app.py – Flask App Code

from flask import Flask, render\_template, request, jsonify

import numpy as np

import cv2

import base64

import re

from tensorflow.keras.models import load\_model

import matplotlib.pyplot as plt

import io

import os

app = Flask(\_\_name\_\_)

model = load\_model("mnist\_ann\_model.h5")

@app.route("/")

def index():

return render\_template("index.html")

@app.route("/predict", methods=["POST"])

def predict():

try:

# Get base64 image data

data\_url = request.form["image"]

encoded\_data = re.sub('^data:image/.+;base64,', '', data\_url)

img\_bytes = base64.b64decode(encoded\_data)

# Convert to NumPy array

nparr = np.frombuffer(img\_bytes, np.uint8)

img = cv2.imdecode(nparr, cv2.IMREAD\_UNCHANGED) # RGBA

# Preprocess

img = cv2.cvtColor(img, cv2.COLOR\_RGBA2GRAY)

\_, img = cv2.threshold(img, 100, 255, cv2.THRESH\_BINARY)

img = cv2.resize(img, (28, 28))

# Center using bounding box

contours, \_ = cv2.findContours(img, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

if contours:

x, y, w, h = cv2.boundingRect(contours[0])

digit = img[y:y+h, x:x+w]

padded = cv2.copyMakeBorder(digit, 10, 10, 10, 10, cv2.BORDER\_CONSTANT, value=0)

img = cv2.resize(padded, (28, 28))

# Normalize

img = img.astype("float32") / 255.0

img\_flat = img.reshape(1, 784)

# Predict

pred = model.predict(img\_flat)

pred\_class = int(np.argmax(pred))

pred\_probs = pred[0].tolist()

return jsonify({

"prediction": pred\_class,

"probabilities": pred\_probs

})

except Exception as e:

return jsonify({"error": str(e)})

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

### 2️⃣ templates/index.html – Drawing UI

<!DOCTYPE html>

<html>

<head>

<title>MNIST Digit Recognizer</title>

<style>

canvas {

border: 2px solid #000;

background-color: black;

}

#probabilities {

margin-top: 20px;

}

</style>

</head>

<body>

<h2>Draw a Digit Below 👇</h2>

<canvas id="canvas" width="280" height="280"></canvas>

<br>

<button onclick="submitCanvas()">✅ Submit</button>

<button onclick="clearCanvas()">🧹 Clear</button>

<h3 id="result"></h3>

<div id="probabilities"></div>

<script>

const canvas = document.getElementById("canvas");

const ctx = canvas.getContext("2d");

let drawing = false;

canvas.addEventListener("mousedown", () => drawing = true);

canvas.addEventListener("mouseup", () => drawing = false);

canvas.addEventListener("mouseout", () => drawing = false);

canvas.addEventListener("mousemove", draw);

function draw(e) {

if (!drawing) return;

ctx.strokeStyle = "white";

ctx.lineWidth = 15;

ctx.lineCap = "round";

ctx.lineTo(e.offsetX, e.offsetY);

ctx.stroke();

ctx.beginPath();

ctx.moveTo(e.offsetX, e.offsetY);

}

function clearCanvas() {

ctx.fillStyle = "black";

ctx.fillRect(0, 0, canvas.width, canvas.height);

ctx.beginPath();

document.getElementById("result").innerText = "";

document.getElementById("probabilities").innerHTML = "";

}

function submitCanvas() {

const dataURL = canvas.toDataURL("image/png");

fetch("/predict", {

method: "POST",

headers: { "Content-Type": "application/x-www-form-urlencoded" },

body: "image=" + encodeURIComponent(dataURL)

})

.then(response => response.json())

.then(data => {

if (data.error) {

document.getElementById("result").innerText = "❌ Error: " + data.error;

} else {

document.getElementById("result").innerText = "✅ Predicted Digit: " + data.prediction;

// Show probabilities

let bars = "";

data.probabilities.forEach((p, i) => {

bars += `<div>${i}: <progress value="${p}" max="1"></progress> ${Math.round(p \* 100)}%</div>`;

});

document.getElementById("probabilities").innerHTML = bars;

}

});

}

// Initialize canvas

clearCanvas();

</script>

</body>

</html>

### ✅ To Run It:

pip install flask tensorflow opencv-python

python app.py

Then open [http://localhost:5000](http://localhost:5000/)

### Want to Upgrade It?

Let me know if you’d like:

CNN-based model version

Save drawings to dataset

Convert to REST API

Ready to ship this anywhere 🚀