**Project Report: Teachable Machine Webpage with Database Integration**

**Introduction:**

In this project, a webpage was designed to deploy a model trained using **Teachable Machine**. The main goal was to use the camera to detect objects (such as earrings, glasses, hats) based on the trained model, store the number of detections in a **MySQL database**, and display the detection count on the webpage. Additionally, a button was added to reset the detection counter.

**Objectives:**

* **Load the Teachable Machine model** and use it to detect objects via the camera.
* **Store the number of detections** (made via the camera) in a **MySQL database**.
* **Display the detection count** on the webpage for the user to track.
* **Add a reset button** to reset the detection counter.

**Steps:**

1. **Setting Up the Environment:**
   * **HTML**, **CSS**, **JavaScript**, **PHP**, and **MySQL** were used to implement the project.
   * A MySQL database was set up to store the detection count.
2. **Folder Structure:**
   * The folder structure was organized as follows:

bash

/ project\_folder

├── index.html

├── style.css

├── script.js

├── model (folder containing the model files)

└── update\_counter.php

1. **Model Loading:**
   * The model trained via **Teachable Machine** was loaded using **TensorFlow.js** and **@teachablemachine/image**.
2. **Camera Setup:**
   * The camera was accessed using **getUserMedia** to capture video and analyze frames with the trained model.
3. **Storing Data in MySQL:**
   * The detection count and class names were sent to the database using **PHP**.

**Files:**

**1. index.html**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Teachable Machine Detection</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Teachable Machine Detection</h1>

<video id="video" width="640" height="480" autoplay></video>

<p>Detection Count: <span id="detectionCount">0</span></p>

<button onclick="resetCounter()">Reset Counter</button>

<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>

<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image"></script>

<script src="script.js"></script>

</body>

</html>

**2. style.css**

css

body {

background-color: #f8d0e7; /\* Light Pink Background \*/

font-family: Arial, sans-serif;

text-align: center;

}

h1 {

color: #333;

}

#video {

border: 2px solid #fff;

margin-top: 20px;

}

button {

background-color: #fff;

color: #333;

border: none;

padding: 10px 20px;

cursor: pointer;

margin-top: 20px;

}

button:hover {

background-color: #f0f0f0;

}

**3. script.js**

javascript

let model;

let videoElement = document.getElementById('video');

let detectionCount = 0;

let detectionCountElement = document.getElementById('detectionCount');

// Load the model from Teachable Machine

async function loadModel() {

const modelURL = 'model/model.json'; // Your model URL

model = await tmImage.load(modelURL);

detectFrame();

}

// Set up the camera

navigator.mediaDevices.getUserMedia({ video: true }).then(stream => {

videoElement.srcObject = stream;

});

// Detect objects

async function detectFrame() {

const prediction = await model.predict(videoElement);

const topPrediction = prediction[0];

if (topPrediction.probability > 0.8) {

simulateDetection(topPrediction.className); // Replace simulateDetection with your code

}

requestAnimationFrame(detectFrame); // Keep detecting for each frame

}

// Update the counter when a detection occurs

function simulateDetection(className) {

detectionCount++;

detectionCountElement.textContent = detectionCount;

// Send the class name to the database

fetch('update\_counter.php', {

method: 'POST',

body: new URLSearchParams({

'detection\_count': detectionCount,

'class\_name': className

})

})

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

.then(data => console.log('Counter updated:', data))

.catch(error => console.error('Error updating counter:', error));

}

// Reset the detection counter

function resetCounter() {

detectionCount = 0;

detectionCountElement.textContent = detectionCount;

}

loadModel(); // Load the model when the page loads

**4. update\_counter.php**

php

نسختحرير

<?php

// Connect to MySQL database

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "teachable\_machine\_db"; // Replace with your database name

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Receive detection count and class name from the request

if ($\_SERVER['REQUEST\_METHOD'] == 'POST' && isset($\_POST['detection\_count']) && isset($\_POST['class\_name'])) {

$detectionCount = (int)$\_POST['detection\_count'];

$className = $\_POST['class\_name'];

// Update the counter in the database

$sql = "UPDATE detection\_counter SET count = $detectionCount WHERE id = 1";

if ($conn->query($sql) === TRUE) {

// Optionally, store the detection log in another table

$logSql = "INSERT INTO detection\_log (class\_name, detection\_count) VALUES ('$className', $detectionCount)";

$conn->query($logSql);

echo json\_encode(['status' => 'success', 'count' => $detectionCount, 'class\_name' => $className]);

} else {

echo json\_encode(['status' => 'error', 'message' => $conn->error]);

}

}

$conn->close();

?>

**MySQL Database:**

To store the detection count and class names of detected objects, you need to create the following database and tables in MySQL:

sql

CREATE DATABASE teachable\_machine\_db;

USE teachable\_machine\_db;

CREATE TABLE detection\_counter (

id INT AUTO\_INCREMENT PRIMARY KEY,

count INT DEFAULT 0

);

CREATE TABLE detection\_log (

id INT AUTO\_INCREMENT PRIMARY KEY,

class\_name VARCHAR(255),

detection\_count INT

);

**Conclusion:**

In this project, a webpage was developed to interact with the camera and detect objects using **Teachable Machine**. The detection count and the class name of detected objects were sent to a MySQL database for storage. Additionally, a button was included to reset the counter. This project allows the tracking and storing of interactions with the trained model.

