HTML file

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>AI</title>

</head>

<body>

    <canvas id = "canvas"></canvas>

    <input type = "file" id = "fileInput" accept = "image/bmp" multiple >

    </input>

    <input type="file" name="photo" id="addEtalonBtn" accept = "image/bmp" multiple>

    <label for="addEtalonBtn" class="btn">Додайте еталон</label>

    </input>

    <button class="recognizeBtn btn">Розпізнати</button>

    <p class="output"></p>

</body>

<script type = "module" src = "main.js"></script>

</html>

Js file

const fileInput = document.querySelector("#fileInput");

const addEtalonBtn = document.querySelector("#addEtalonBtn");

const recognizeBtn = document.querySelector(".recognizeBtn");

const recognizeOut = document.querySelector(".output");

let pixelColors;

let n;

const meshSize = 4;

let fileName = "";

let clasters = {};

let vector;

const perceptron\_size=36;

class Perceptron {

  constructor(inputSize) {

    this.weights = new Array(inputSize);

    this.learningRate = 0.15;

    for (let i = 0; i < inputSize; i++) {

      this.weights[i] = Math.random() \* (1 - (-1)) -1; // ініціалізація ваг від -1 до 1

    }

  }

  predict(inputs) {

    let sum = 0;

    for (let i = 0; i < this.weights.length; i++) {

      sum += inputs[i] \* this.weights[i];

    }

    return this.activate(sum);

  }

  activate(sum) {

    return sum >= 0 ? 1 : -1;

  }

  train(inputs, target) {

    const guess = this.predict(inputs);

    const error = target - guess;

    // Оновлення ваг згідно з правилом навчання перцептрона

    for (let i = 0; i < this.weights.length; i++) {

      this.weights[i] += error \* inputs[i] \* this.learningRate;

    }

  }

}

addEtalonBtn.addEventListener("change", async () => {

  console.log(addEtalonBtn.files)

  for (let i = 0; i < addEtalonBtn.files.length; i++) {

    let imageFile = addEtalonBtn.files[i];

    const img = document.createElement('img');

    img.src = URL.createObjectURL(imageFile);

    const canvas = document.createElement('canvas');

    img.onload = async () => {

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

      canvas.width = img.width;

      canvas.height = img.height;

      n = img.width;

      ctx.drawImage(img, 0, 0, img.width, img.height);

      const imageData = ctx.getImageData(0, 0, img.width, img.height);

      pixelColors = imageData.data;

      vector = generateVector(pixelColors);

      fileName = imageFile.name.split("\_")[0];

      const target = getTargetForFileName(fileName);

      perceptron.train(vector, target);

      console.log(perceptron.weights);

    };

  }

})

const perceptron = new Perceptron(perceptron\_size);

console.log(perceptron.weights)

fileInput.addEventListener("change", async () => {

    let imageFile = fileInput.files[0];

    fileName = imageFile.name.split("\_")[0];

    const reader = new FileReader();

    reader.onload = async () => {

      const imageBuffer = reader.result;

      const img = document.createElement('img');

      img.src = URL.createObjectURL(imageFile);

      img.onload = async () => {

        const canvas = document.querySelector('#canvas');

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

        canvas.width = img.width;

        canvas.height = img.height;

        n = img.width;

        ctx.drawImage(img, 0, 0, img.width, img.height);

        const imageData = ctx.getImageData(0, 0, img.width, img.height);

        pixelColors = imageData.data;

        console.log(pixelColors)

        vector = generateVector(pixelColors);

      };

      // Завантажте зображення

      img.src = URL.createObjectURL(imageFile);

    };

    reader.readAsArrayBuffer(imageFile);;

});

function getTargetForFileName(fileName) {

  if (fileName.includes("1")) {

    return 1;

  } else if (fileName.includes("0")) {

    return -1;

  } else {

    return 0; // Якщо не можна визначити таргет, повертаємо 0 або інше значення

  }

}

recognizeBtn.addEventListener("click", () => {

  if (!fileInput.files[0]) {

    recognizeOut.textContent = "Виберіть зображення";

    return;

  }

  let prediction = perceptron.predict(vector);

  if (prediction==-1){

    prediction=0;

  }

  recognizeOut.textContent = "Відповідь: " + prediction;

  console.log(perceptron.weights)

})

function normVector(vector){

  return vector.map((num) => num / Math.max(...vector));

}

function getAvgVector(x1,x2){

  return x1.map((obj, index)=> (obj + x2[index])/2);

}

function generateVector(pixels) {

  let colorBits = [];

  let vector = [];

  for (let i = 0; i <= pixels.length - meshSize; i += meshSize) {

    colorBits.push(pixels[i] === 0 ? 1 : 0);

  }

  console.log("colorBits", colorBits, "Довжина", colorBits.length);

  for (let i = 0; i < n \* n - n; i += n \* 2) {

    for (let j = 0; j < n; j += Math.sqrt(meshSize)) {

      let count = 0;

      count += colorBits[i + j];

      count += colorBits[i + j + 1];

      count += colorBits[i + j + 12];

      count += colorBits[i + j + 13];

      vector.push(count);

      count = 0;

    }

  }

  vector = normVector(vector);

  console.log(vector)

  return vector;

}

