МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

ДВНЗ «УЖГОРОДСЬКИЙ НАЙЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ»

**ЗВIT**

**ПРАКТИЧНА РОБОТА**

З дисципліни «Інформаційно-технічний супровід »

**Багатошаровий перцептрон. Задача класифікації**

**Частина 3**

Виконав:

Ролич Михайло Михайлович,

студент 3-го курсу

Перевірив:

проф. Поліщук Володимир

Ужгород 2025

**Варіант:** 3 Багатошаровий перцептрон. Задача класифікації.

**Інструкція користувача до вебдодатку «Класифікація фруктів»**

Призначення програми: Цей вебдодаток дозволяє класифікувати фрукти на основі введених характеристик: **вага**, **колір (R, G, B)** та **солодкість**. Після натискання кнопки "Класифікувати" штучна нейронна мережа, навчена у браузері, виводить найімовірніший тип фрукта: **яблуко**, **банан** або **виноград**.

Інтерфейс програми:

**Кнопка "Навчити модель**

Запускає процес навчання нейронної мережі. Після завершення з’явиться повідомлення:  
**"Статус: Навчання завершено!"**

Форма введення параметрів фрукта:

* Вага (грам): наприклад, 150
* Колір:

R (Red)**:** значення від 0 до 1 (наприклад, 0.8)

G (Green): значення від 0 до 1 (наприклад, 0.6)

B (Blue): значення від 0 до 1 (наприклад, 0.2)

* Солодкість: значення від 0 (несолодкий) до 1 (максимально солодкий)

**Кнопка "Класифікувати"**

Активується після навчання моделі. На основі введених даних виводить назву фрукта, який найбільше відповідає цим характеристикам.

**Поле результату:**

Відображає результат класифікації, наприклад:

Тип фрукта: яблуко

**Графік "Втрата (Loss)"**

Після завершення навчання на графіку відображається, як змінювалася втрата під час епох — це допомагає зрозуміти, наскільки успішно навчалась модель.

**Покрокове керівництво**

1. Відкрийте вебдодаток у браузері. Працює в Chrome, Firefox, Edge або Safari.
2. Натисніть кнопку "Навчити модель". Модель почне тренуватись прямо в браузері. На це піде кілька секунд.
3. Після завершення навчання введіть дані: Наприклад: вага — 180, R — 0.9, G — 0.1, B — 0.1, солодкість — 0.85
4. Натисніть "Класифікувати". Результат з’явиться миттєво нижче — наприклад: Тип фрукта: банан.
5. Оцініть графік навчання. На ньому видно, як зменшувався показник "втрати" протягом 150 епох.

**Попередження та обмеження**

* Усі значення кольору та солодкості вводяться як десяткові числа від 0 до 1.
* Модель потрібно спочатку навчити, інакше кнопка "Класифікувати" не працюватиме.
* Якщо ви введете некоректні або порожні значення, результат може бути неточним або не з’явиться зовсім.

**Спрощене технічне завдання (СТЗ)**

Мета: Розробити простий вебдодаток з використанням TensorFlow.js, який дозволяє класифікувати фрукти (яблуко, банан, виноград) за характеристиками: вага, колір (RGB), солодкість. Модель має навчатися прямо у браузері, без використання серверної частини.

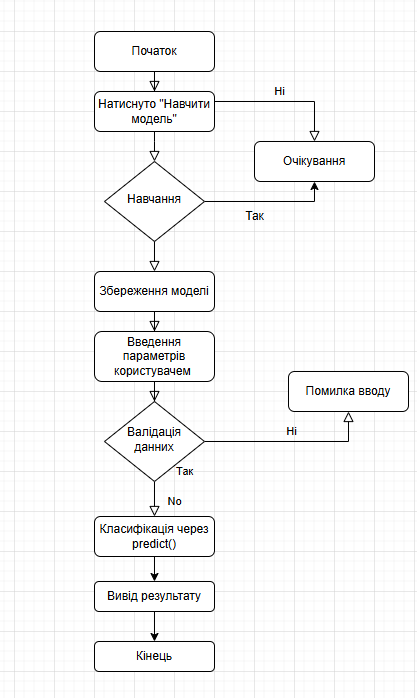
Функціональні вимоги

1. **Інтерфейс користувача:**
   1. Форма для введення даних фрукта:
      1. Вага (в грамах) — число
      2. Колір (Red, Green, Blue) — значення від 0 до 1
      3. Солодкість — значення від 0 до 1
   2. Кнопка "Навчити модель"
   3. Кнопка "Класифікувати"
   4. Вивід результату: назва фрукта
   5. Графік втрати навчання (Loss) — побудований за допомогою Chart.js
2. **Модель нейронної мережі:**
   1. Реалізується за допомогою TensorFlow.js
   2. Містить щонайменше 2 шари (вхідний, прихований та вихідний)
   3. Вхідні параметри: масив із 5 чисел [weight, r, g, b, sweetness]
   4. Вихід: ймовірність приналежності до класу яблуко/банан/виноград
3. **Навчання моделі:**
   1. Запускається вручну кнопкою "Навчити модель"
   2. Модель навчається на заздалегідь закладених (вшитих у код) тренувальних даних
   3. Графік показує зміну втрати (loss) під час кожної епохи
4. **Класифікація:**
   1. Після навчання користувач вводить нові параметри фрукт
   2. Натискання кнопки "Класифікувати" повертає назву фрукта з найбільшою ймовірністю

**Нефункціональні вимоги**

* Уся логіка виконується в браузері (JavaScript, без бекенду)
* Модель навчається динамічно, а не заздалегідь
* Код має бути зрозумілим, з коментарями
* Інтерфейс має бути адаптований для роботи на комп’ютерах (можна без мобільної версії)
* Підключені бібліотеки:
  + TensorFlow.js
  + Chart.js
  + JavaScript,

**Блок Схема**

****

**Лістинг**

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

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

    <title>Класифікація Фруктів</title>

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

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

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <script type="importmap">     {

        "imports": {

            "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"

        }

    }</script>

    <script src='./script.js' defer></script>

    <style>

        article {

            border: 1px solid black;

            margin: 10px;

            padding: 10px;

            border-radius: 10px;

        }

    </style>

</head>

<body>

    <header class="header">

        <nav class="header\_\_menu">

            <h1>lOGO</h1>

            <ul class="header\_\_menu-list">

                <li></li>

            </ul>

        </nav>

    </header>

    <h2>Класифікація фруктів</h2>

    <button onclick="trainModel()"> Навчити модель</button>

    <p id="status">Статус: очікується запуск...</p>

    <h3>Ввести характеристики фрукта</h3>

    <input type="number" id="input1" placeholder="Вага (г)">

    <input type="number" max="1" min="0" id="input2" placeholder="Колір R (0-1)">

    <input type="number" max="1" min="0" id="input3" placeholder="Колір G (0-1)">

    <input type="number" max="1" min="0" id="input4" placeholder="Колір B (0-1)">

    <input type="number" max="1" min="0" id="input5" placeholder="Солодкість (0-1)">

    <button onclick="predict()"> Класифікувати</button>

    <h3>Результат:</h3>

    <p id="prediction">Очікується введення...</p>

    <h3>Графік навчання</h3>

    <canvas id="lossChart"></canvas>

</body>

</html>

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 20px;

    background-color: #f5f5f5;

    color: #333;

}

h2 {

    text-align: center;

    color: #007BFF;

}

h3 {

    margin-top: 20px;

    color: #555;

}

button {

    background-color: #007BFF;

    color: white;

    padding: 10px 20px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 16px;

}

button:hover {

    background-color: #0056b3;

}

input {

    width: 150px;

    padding: 8px;

    margin: 10px 5px;

    border: 1px solid #ddd;

    border-radius: 5px;

    font-size: 14px;

}

input:focus {

    border-color: #007BFF;

    outline: none;

}

#prediction {

    font-size: 18px;

    font-weight: bold;

    color: #333;

}

#status {

    font-size: 16px;

    color: #888;

}

canvas {

    margin-top: 30px;

    max-width: 100%;

    height: 400px;

    border: 1px solid #ddd;

    background-color: #fff;

}

.container {

    width: 100%;

    max-width: 900px;

    margin: 0 auto;

    background-color: white;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

.form-container {

    display: flex;

    flex-direction: column;

    align-items: center;

}

.form-container input {

    margin-bottom: 10px;

}

.form-container button {

    margin-top: 10px;

}

let model;

let chart;

function createChart() {

    const ctx = document.getElementById('lossChart').getContext('2d');

    if (chart) chart.destroy();

    chart = new Chart(ctx, {

        type: 'line',

        data: { labels: [], datasets: [{ label: 'Втрата', data: [], borderColor: 'red', fill: false }] },

        options: { responsive: true, scales: { x: { title: { text: 'Епоха', display: true } }, y: { title: { text: 'Втрата', display: true } } } }

    });

    return chart;

}

function updateChart(chart, losses) {

    chart.data.labels = losses.map((\_, i) => i + 1);

    chart.data.datasets[0].data = losses;

    chart.update();

}

async function trainModel() {

    document.getElementById('status').innerText = "Статус: Навчання...";

    const trainXs = tf.tensor2d([

        [150, 1, 0, 0, 0.7], // Яблуко

        [120, 1, 0.2, 0.1, 0.8], // Яблуко

        [200, 1, 1, 0, 0.9], // Банан

        [180, 1, 1, 0.2, 0.85], // Банан

        [50, 0.5, 0, 0.5, 0.6], // Виноград

        [40, 0.4, 0.1, 0.7, 0.5], // Виноград

    ]);

    const trainYs = tf.tensor2d([

        [1, 0, 0], // Яблуко

        [1, 0, 0], // Яблуко

        [0, 1, 0], // Банан

        [0, 1, 0], // Банан

        [0, 0, 1], // Виноград

        [0, 0, 1], // Виноград

    ]);

    model = tf.sequential();

    model.add(tf.layers.dense({ units: 8, activation: 'relu', inputShape: [5] }));

    model.add(tf.layers.dense({ units: 8, activation: 'relu' }));

    model.add(tf.layers.dense({ units: 3, activation: 'softmax' }));

    model.compile({ optimizer: tf.train.adam(0.01), loss: 'categoricalCrossentropy', metrics: ['accuracy'] });

    const losses = [];

    const chart = createChart();

    await model.fit(trainXs, trainYs, {

        epochs: 150,

        batchSize: 4,

        callbacks: {

            onEpochEnd: async (epoch, logs) => {

                console.log(`Epoch ${epoch + 1}: Loss = ${logs.loss.toFixed(4)}`);

                losses.push(logs.loss);

                updateChart(chart, losses);

            }

        }

    });

    document.getElementById('status').innerText = "Статус: Навчання завершено!";

}

function predict() {

    const inputData = [

        parseFloat(document.getElementById('input1').value),

        parseFloat(document.getElementById('input2').value),

        parseFloat(document.getElementById('input3').value),

        parseFloat(document.getElementById('input4').value),

        parseFloat(document.getElementById('input5').value)

    ];

    let isValid = true;

    const inputs = [document.getElementById('input1'), document.getElementById('input2'), document.getElementById('input3'), document.getElementById('input4'), document.getElementById('input5')];

    inputs.forEach(input => input.style.borderColor = "");

    if (inputData.some(isNaN) || inputData[0] <= 0 || inputData.slice(1).some(val => val < 0 || val > 1)) {

        document.getElementById('prediction').innerText = "Помилка: некоректні вхідні дані";

        inputs.forEach((input, index) => {

            if (isNaN(inputData[index]) ||

                (index === 0 && inputData[index] <= 0) ||

                (index > 0 && (inputData[index] < 0 || inputData[index] > 1))) {

                input.style.borderColor = "red";

            }

        });

        return;

    }

    const inputTensor = tf.tensor2d([inputData]);

    const prediction = model.predict(inputTensor);

    const predictedClass = prediction.argMax(1).dataSync()[0];

    const labels = ['Яблуко', 'Банан', 'Виноград'];

    document.getElementById('prediction').innerText = `Модель передбачила: ${labels[predictedClass]}`;

}

**Висновки**

У процесі реалізації програмного забезпечення для класифікації фруктів були досягнуті наступні результати:

1. Розроблено веб-додаток, що дозволяє користувачеві класифікувати фрукти на основі введених характеристик (вага, колірні компоненти, солодкість) із використанням нейронної мережі на базі TensorFlow.js.
2. Навчена модель штучного інтелекту, що демонструє прийнятну точність у розпізнаванні трьох класів фруктів — яблук, бананів та винограду.
3. Реалізовано інтерактивний інтерфейс, який дозволяє вводити дані вручну, відображати результат класифікації та спостерігати динаміку навчання за допомогою графіку втрат.
4. Інтегровано Vue.js, що забезпечило гнучку організацію інтерфейсу та розширюваність застосунку (наприклад, додавання модуля зі створення постів).
5. Здійснено оформлення технічної документації, включаючи інструкцію користувача, спрощене технічне завдання, схеми та лістинг з коментарями, що сприяє зручності подальшої підтримки та доопрацювання ПЗ.

Таким чином, поставлені задачі були повністю виконані, а результат роботи може бути використаний як приклад інтелектуального рішення для задач класифікації в навчальних або демонстраційних цілях. У майбутньому систему можна розширити підтримкою інших типів фруктів або інтеграцією з камерою для автоматичного збору вхідних даних.