Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01– «Информатика и вычислительная техника»

**проект**

**по дисциплине**

**«Искусственный интеллект»**

**на тему**

**Разработка веб-страницы распознавания рукописного ввода**

Выполнили студент гр. КС-19-1б

Дворянских Демид Анатольевич

Проверил:

стар. преп. каф. ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

(дата)

Пермь 2022

Содержание

[Введение 3](#_Toc103245123)

[Реализация программы 4](#_Toc103245124)

[Пример работы программы 6](#_Toc103245125)

[ПРИЛОЖЕНИЕ А – Код программы 7](#_Toc103245126)

Введение

Нейросеть - разновидность машинного обучения, при котором программа работает по принципу человеческого мозга.

Общая схема или алгоритм следующий:  
— на входной слой нейронов происходит поступление определённых данных;  
— информация передаётся с помощью синапсов следующему слою, причём каждый синапс имеет собственный коэффициент веса, а любой следующий нейрон способен иметь несколько входящих синапсов;  
— данные, полученные следующим нейроном, — это сумма всех данных для нейронных сетей, которые перемножены на коэффициенты весов (каждый на свой);  
— полученное в итоге значение подставляется в функцию активации, в результате чего происходит формирование выходной информации;  
— информация передаётся дальше до тех пор, пока не дойдёт до конечного выхода. Примерная работа ИИ в упрощенном виде представлена на рисунке 1.

|  |
| --- |
|  |
| Рисунок 1 – Порядок распознавания рукописного ввода |

Данный искусственный интеллект будет распознавать рукописный ввод на основе выбора пользователя, подобному заполнению бланков с полем голоса “За” или “Против”. Нейросеть расчитана на прием изображения в виде “галочка” или “крест”, на основе рукописных изображений нейросеть научится, а затем сможет определить, какой выбор сделал пользователь.

Реализация программы

**Библиотеки:**

1. Brain.js

Для реализации потребуется единственная Javascript библиотека – Brain.js, используемая для искусственных нейронных сетей, заменяющая «мозговую» библиотеку, которая предлагает разные [типы](https://github.com/BrainJS/brain.js#neural-network-types) сетей в зависимости от конкретных задач. Ее использование возможно прямо в[браузере.](https://raw.githubusercontent.com/harthur-org/brain.js/master/browser.js)

**Среда разработки**

Для разработки использованы возможности языка JavaScript совместно с HTML+CSS.

Основная возможность, которая потребуется – элемент  [<canvas>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/canvas), добавленный в [HTML5](https://developer.mozilla.org/en-US/docs/HTML/HTML5), предназначен для создания графики с помощью [JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript" \o "JavaScript). Его используют для рисования графиков, создания фотокомпозиций, анимаций и т.п.

**Порядок работы**

Первый шаг – создание файла формата html, загрузка и подключение библиотеки Brain.js

Далее – разметка страницы и определение стилей для отображения графической составляющей интерфейса

Следующий шаг – создание и задание параметров рабочего графического поля(канваса), в котором будет производиться процесс рисования.

Затем прописываем необходимые функции для корректной отрисовки изображения по канвасу мышью, а также добавляем возможность использовать горячую клавишу для функции очистки изображения.

Далее – этап взаимодействия с библиотекой нейросети, для этого также добавляем горячие клавиши с функциями работы Brain.js, ими будут служить – запоминание(обучение) и распознавание. То есть при нажатии на одну клавишу нейросеть запоминает изображение и правильный ответ от человека для обработки, а при нажатии на другую – нейросеть выдаст свой результат на основе анализа своих данных.

Последний этап – выдать достаточное количество различных изображений нейросети и протестировать ее результаты, исправляя ее ошибочные результаты дополнением новыми данными.

Пример работы программы

Обучение ИИ представлено на рисунке ниже.

|  |
| --- |
|  |
| Рисунок 2 – Рисовка и отправка данных на основе пикселизации  Самостоятельное определение рукописи нейросетью на рисунке 3. |
|  |

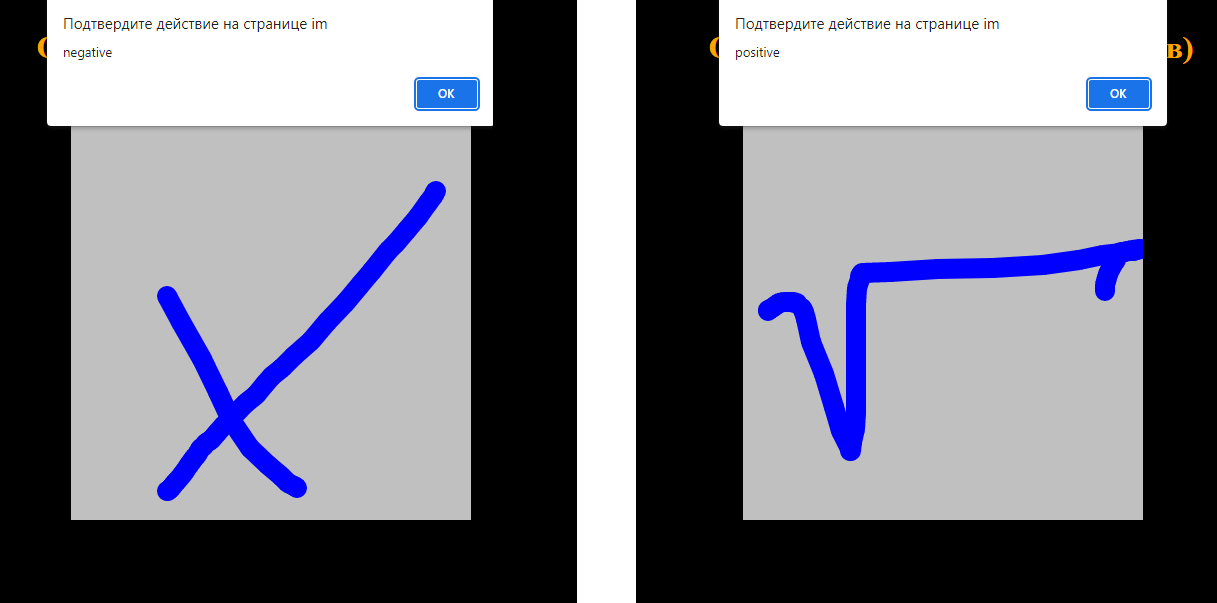


Рисунок 2 – Рисовка и отправка данных на основе пикселизации

ПРИЛОЖЕНИЕ А – Код программы

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>window</title>

<script src="//unpkg.com/brain.js"></script>

<style>

body {

background-color: black;

}

#canv {

position: absolute;

top: 0;right: 0;bottom: 0;left: 0;

margin: auto;

background-color: silver;

}

</style>

</head>

<body>

<div style = 'position:absolute;width:100%;height:30px;color:orange;text-align: center;display:block'>

<h1 style = "display:block">Определение выбора (За/Против)</h1></div>

<canvas id="canv" style="display: block;"></canvas>

<script>

**function** DCanvas(el)

{

*//настраиваю///////////////////////////////////////////////////////////////////////*

**const** ctx = el.getContext('2d');

**const** pixel = 20;

let is\_mouse\_down = **false**;

canv.width = 400;

canv.height = 400;

**var** color = 'black';

*//отрисовываю*

**this**.drawLine = **function**(x1, y1, x2, y2) {

ctx.beginPath();

ctx.strokeStyle = color;

ctx.lineJoin = 'miter';

ctx.lineWidth = 1;

ctx.moveTo(x1, y1);

ctx.lineTo(x2, y2);

ctx.stroke();

}

**this**.drawCell = **function**(x, y, w, h) {

ctx.fillStyle = 'green';

ctx.strokeStyle = 'green';

ctx.lineJoin = 'black';

ctx.lineWidth = 1;

ctx.rect(x, y, w, h);

ctx.fill();

}

**this**.clear = **function**() {

ctx.clearRect(0, 0, canv.width, canv.height);

}

**this**.drawGrid = **function**() {

**const** w = canv.width;

**const** h = canv.height;

**const** p = w / pixel;

**const** xStep = w / p;

**const** yStep = h / p;

**for**( let x = 0; x < w; x += xStep )

{

**this**.drawLine(x, 0, x, h);

}

**for**( let y = 0; y < h; y += yStep )

{

**this**.drawLine(0, y, w, y);

}

}

**this**.calculate = **function**(draw = **false**) {

**const** w = canv.width;

**const** h = canv.height;

**const** p = w / pixel;

**const** xStep = w / p;

**const** yStep = h / p;

**const** vector = [];

let \_\_draw = [];

**for**( let x = 0; x < w; x += xStep )

{

**for**( let y = 0; y < h; y += yStep )

{

**const** data = ctx.getImageData(x, y, xStep, yStep);

let nonEmptyPixelsCount = 0;

**for**( i = 0; i < data.data.length; i += 10 )

{

**const** isEmpty = data.data[i] === 0;

**if**( !isEmpty )

{

nonEmptyPixelsCount += 1;

}

}

**if**( nonEmptyPixelsCount > 1 && draw )

{

\_\_draw.push([x, y, xStep, yStep]);

}

vector.push(nonEmptyPixelsCount > 1 ? 1 : 0);

}

}

**if**( draw )

{

**this**.clear();

**this**.drawGrid();

**for**( \_d **in** \_\_draw )

{

**this**.drawCell( \_\_draw[\_d][0], \_\_draw[\_d][1], \_\_draw[\_d][2], \_\_draw[\_d][3] );

}

}

**return** vector;

}

*//*

el.addEventListener('mousedown', **function**(e) {

is\_mouse\_down = **true**;

ctx.beginPath();

})

el.addEventListener('mouseup', **function**(e) {

is\_mouse\_down = **false**;

})

el.addEventListener('mousemove', **function**(e) {

**if**( is\_mouse\_down )

{

ctx.fillStyle = 'blue';

ctx.strokeStyle = 'blue';

ctx.lineWidth = pixel;

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

ctx.stroke();

ctx.beginPath();

ctx.arc(e.offsetX, e.offsetY, pixel / 2, 0, Math.PI \* 2);

ctx.fill();

ctx.beginPath();

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

}

})

}

let vector = [];

let net = **null**;

let train\_data = [];

**const** d = **new** DCanvas(document.getElementById('canv'));

*//события////////////////////////////////////////////////////////////////////////////////////////////*

document.addEventListener('keypress', **function**(e) {

**if**( e.key.toLowerCase() == 'c' )

{

d.clear();

}

**if**( e.key.toLowerCase() == 'v' )

{

vector = d.calculate(**true**);

*//учить*

**if**( confirm('За?') )

{

train\_data.push({

input: vector,

output: {positive: 1}

});

} **else**

{

train\_data.push({

input: vector,

output: {negative: 1}

});

}

}

**if**( e.key.toLowerCase() == 'b' )

{

net = **new** brain.NeuralNetwork();

net.train(train\_data, {log: **true**});

**const** result = brain.likely(d.calculate(), net);

alert(result);

}

});

</script>