МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6

тема: «Нейрокомпьютерные сети основанные на соревновании.

Сеть Хемминга.»

Выполнил: ст. группы МИВТ-221

Харитонов Сергей Дмитриевич

Белгород 2022 г.

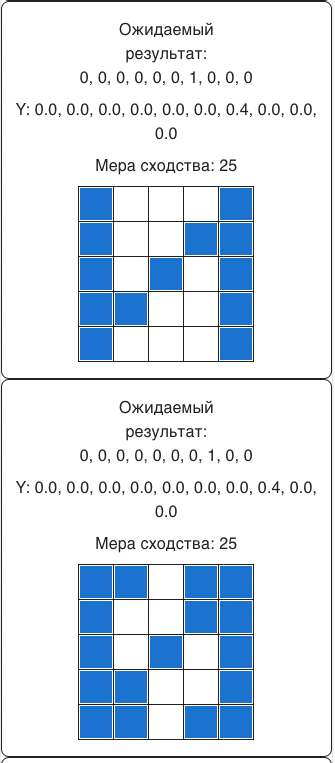
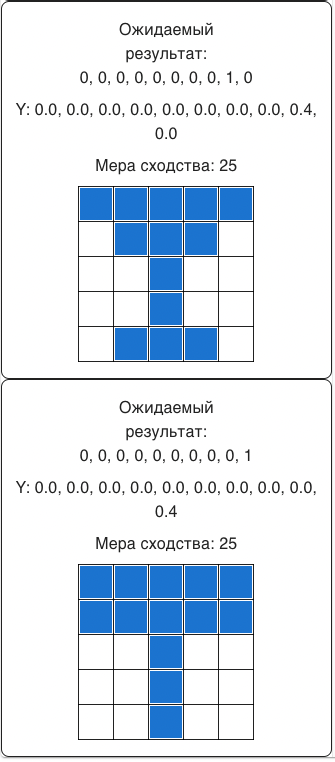
**Лабораторная работа №6**

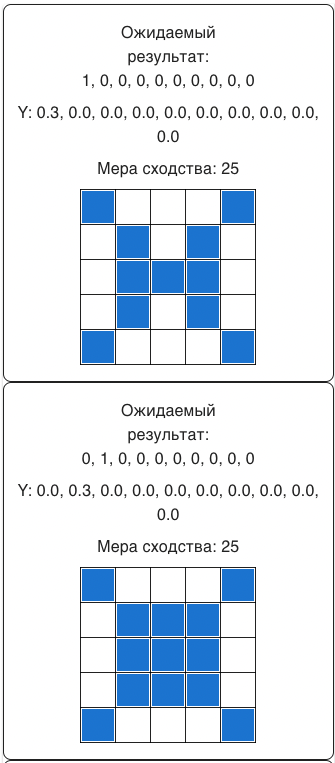
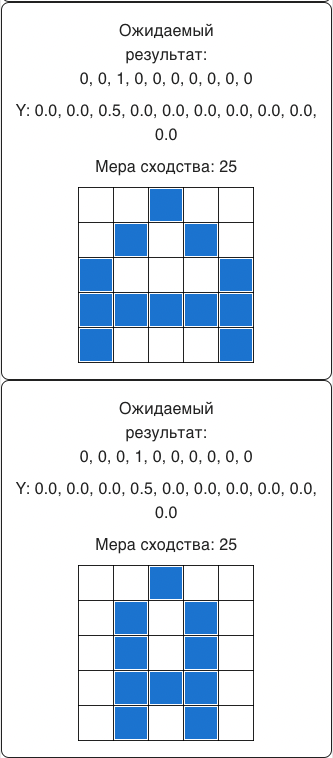
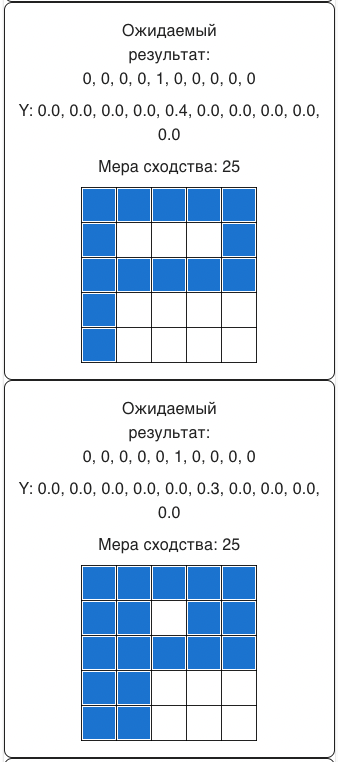
**Нейрокомпьютерные сети основанные на соревновании.**

**Сеть Хемминга.**

**Цель работы:** Приобретение и закрепление знаний, получение практических навыков работы с нейрокомпьютерными сетями, основанными на соревновании, исследование сети Хемминга.

Экранные копии разработанных изображений букв





После инициализации веса для слоя определения расстояния Хемминга выглядят следующим образом

Изображение выглядит как текст

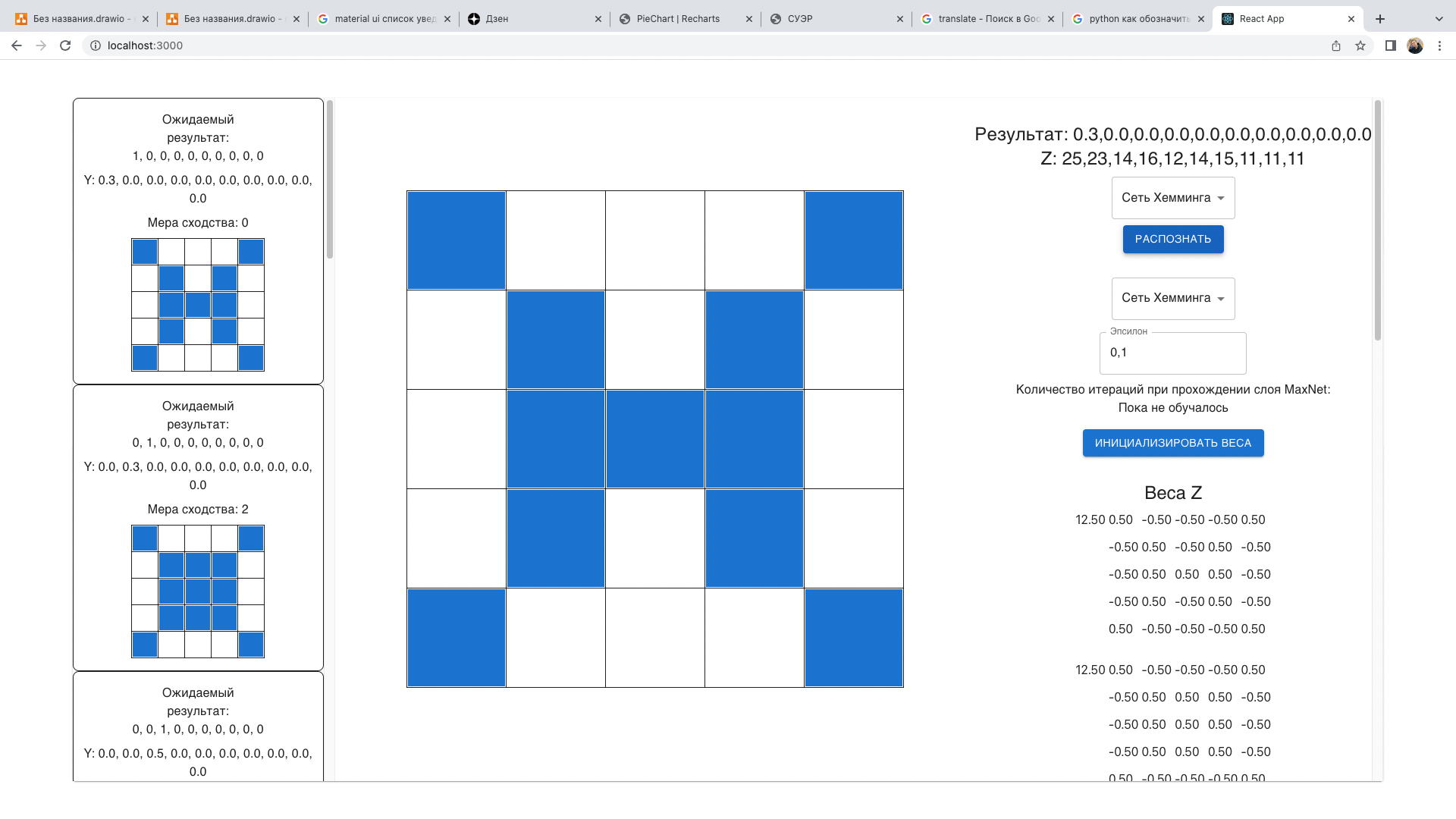
Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описаниеИзображение выглядит как стол

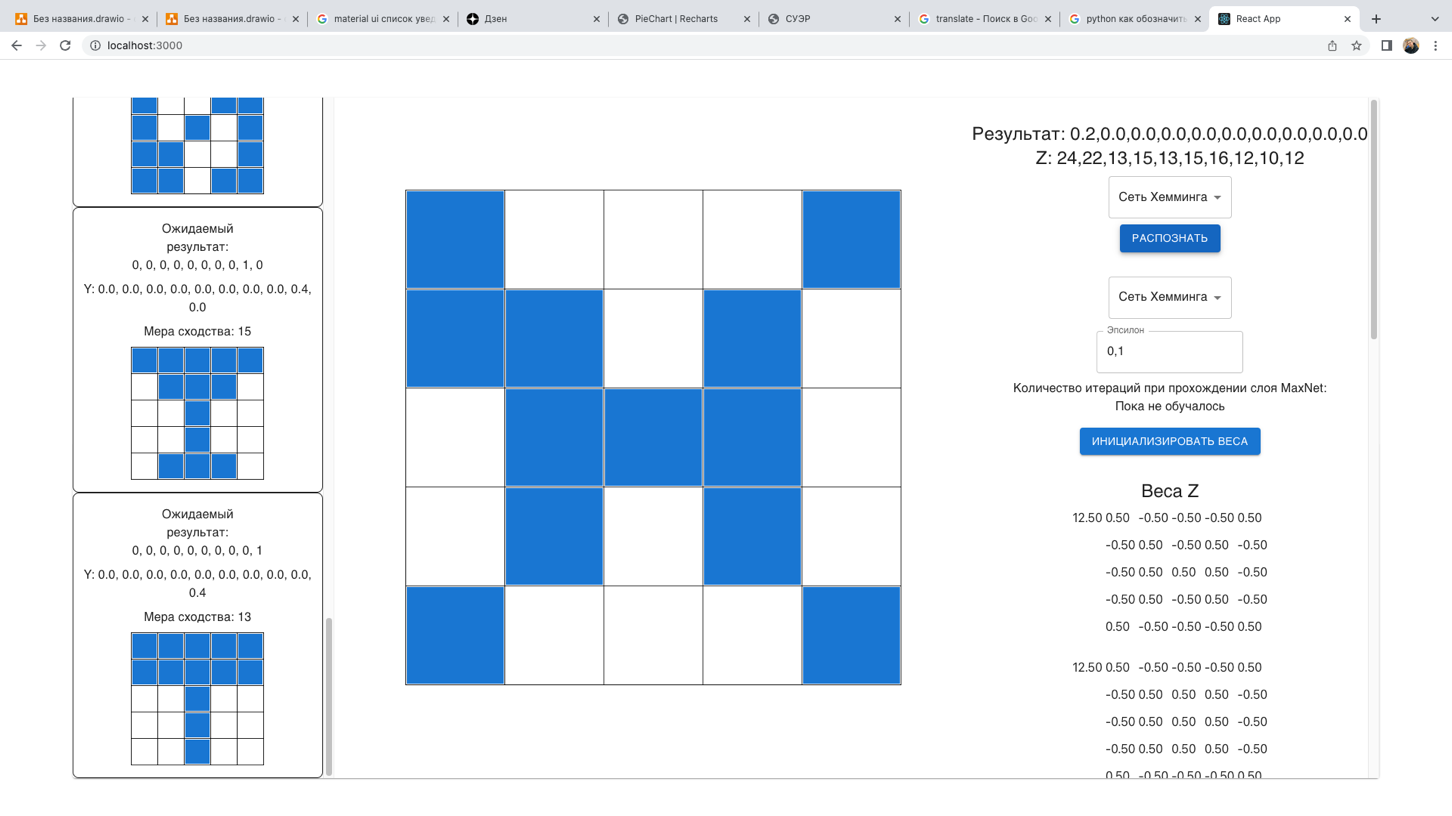
Автоматически созданное описание

Результаты исследования сети Хемминга по распознаванию изображений

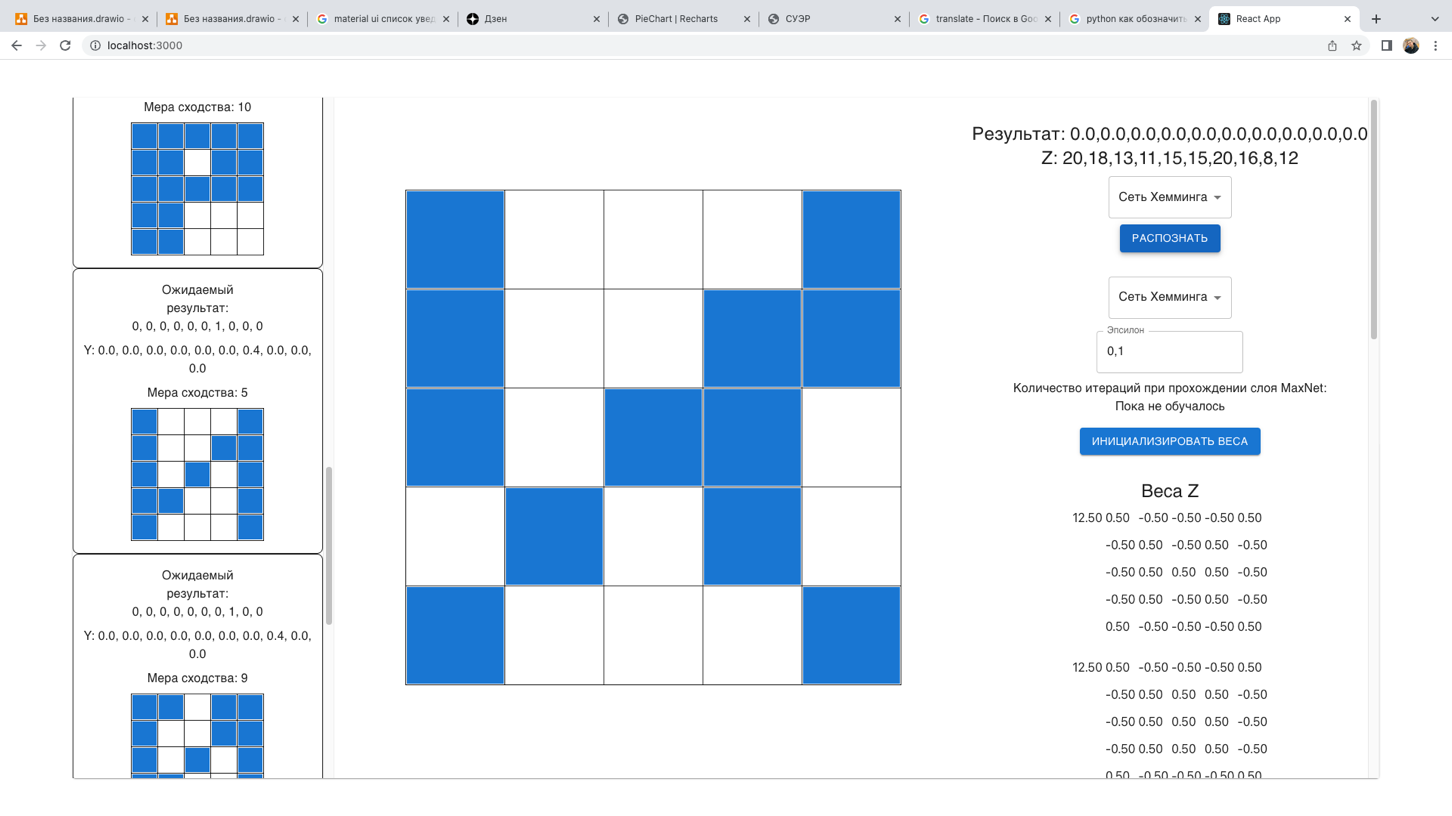
После реализации ИНС было проведено исследование по точечному переходу между буквами Х и И, в результате которого отмечены следующие ключевые точки, исследование происходило при ε = 1/n = 1/10 = 0.1



В начале исследования максимальное сходство с образцом рассчитано верно, равно 25, в следствии чего сеть MaxNet находит максимальное значение в первом нейроне слоя Хемминга и единственным выходом с положительным значением является первый.



Далее по мере того как разница между мерой сходства между нейронами становится меньше и само значение меры сходства для первого нейрона уменьшается, так же уменьшается и выходное значение первого нейрона.



В середине исследования наступает переломный момент, когда меры сходства для 1го нейрона и 7го(к которому мы стремимся) становятся равны, в результате чего они оба затирают друг друга и на выходе мы получаем отсутствие активных нейронов.

Изображение выглядит как стол

Автоматически созданное описание

После чего выходное значение 1го нейрона становится равным 0, а 7ой нейрон наоборот начинает активироваться и набирать значение.

В результате исследования было выяснено, что сеть Хемминга отлично справляется с классификацией изображений, однако при наличии двух одинаковых вариантов не может выбрать один из них и возвращает 0 для всех нейронов выходного слоя.

Результаты исследования, влияния коэффициента скорости обучения сети MaxNet, на процесс выявления «победителя»

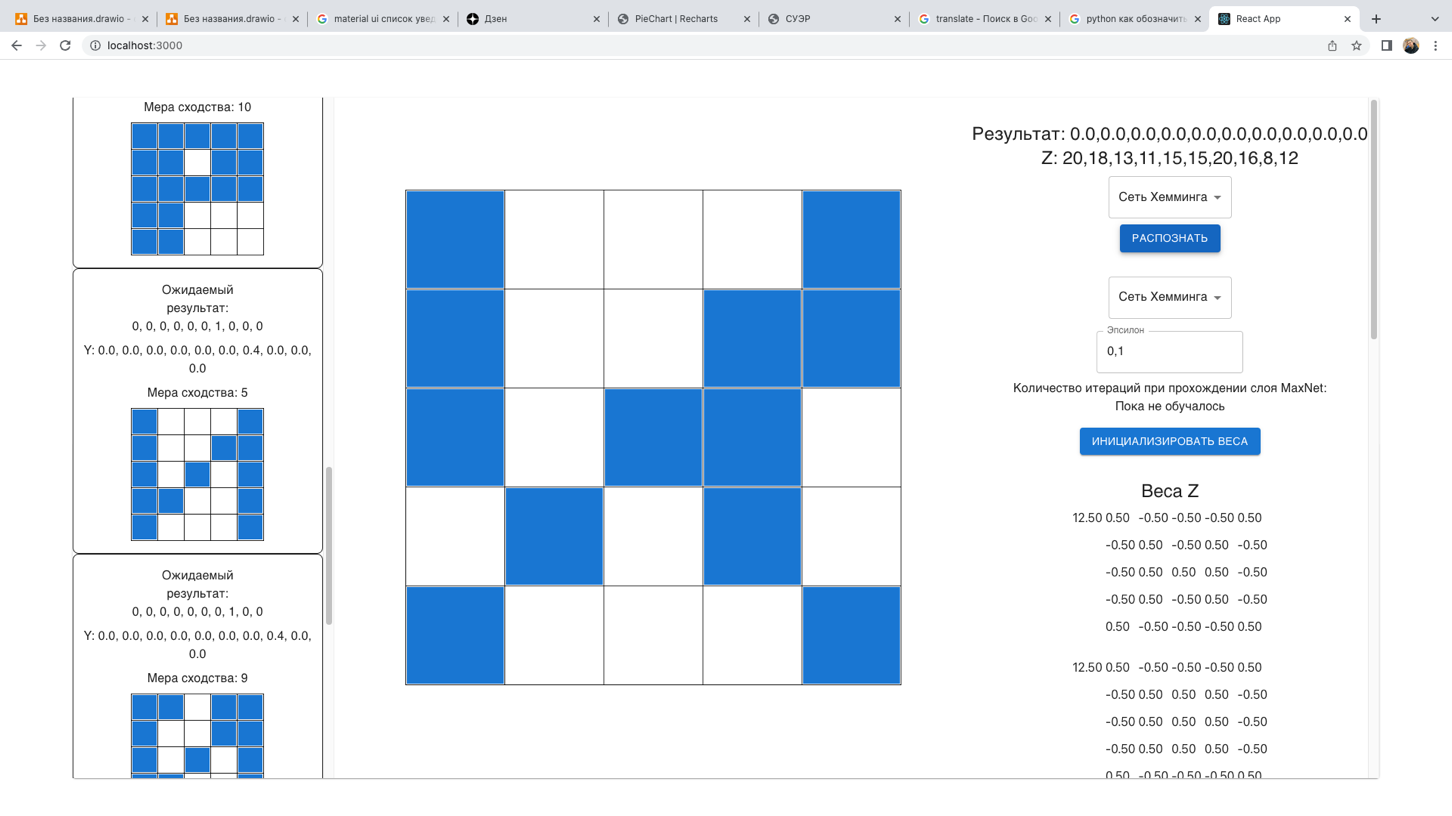
Было произведено исследование результата распознавания и его скорости в зависимости от ε, распознавалась буква Х(первый вариант) в диапазонах [0.05,0.19] с шагом 0.01.

| e | Потребовалось итераций (эпох) | Результат выхода первого нейрона |
| --- | --- | --- |
| 0.05 | 28 | 0.3 |
| 0.06 | 23 | 0.3 |
| 0.07 | 20 | 0.3 |
| 0.08 | 18 | 0.3 |
| 0.09 | 16 | 0.3 |
| 0.10 | 14 | 0.3 |
| 0.11 | 12 | 0.3 |
| 0.12 | 11 | 0.2 |
| 0.13 | 10 | 0.2 |
| 0.14 | 8 | 0.2 |
| 0.15 | 7 | 0.2 |
| 0.16 | 6 | 0.2 |
| 0.17 | 4 | 0.1 |
| 0.18 | 2 | 0.1 |
| 0.19 | 2 | 0.0 |

В ходе исследования выяснилось, что при увеличении коэффициента скорости ε, происходит постепенное уменьшение количества итераций, однако снижается точность результата и в случаях если общая сумма мер сходства довольно большая, а максимум не сильно выражен при достижении определенной границы MaxNet перестает определять победителя, ввиду того что остальные сигналы затирают его на первых итерациях.

Решается это использованием ε = 1/n в таком случае даже если все сигналы близки к максимальному, максимальный сигнал не будет сразу же отброшен и в результате выиграет. Дальнейшее же уменьшение ε не приведет к улучшению результатов, однако увеличит время выполнения алгоритма, из этого следует что наилучший выбор это ε = 1/n где n количество выходных нейронов.

Исследования работы сеть Хемминга, при распознавании изображения, имеющего с двумя и более эталонными одинаковые максимальные меры близости



При распознавании изображения, имеющего с двумя и более эталонными одинаковые максимальные меры близости, происходит взаимное уничтожение выходных сигналов, в результате чего они затирают друг друга и на выходе получается вектор из 0, что свидетельствует о том, что сеть MaxNet не смогла выбрать победителя.

Листинг основных функций программы

import {*attach*, *combine*, *createEffect*, *createEvent*, *createStore*} from "effector";  
import {$M} from '../presets'  
import {$weight} from "../weight";  
import {$recognize} from "../recognize";  
  
export const *getS* = (Xs, Ws) =>  
 Xs  
 .reduce(  
 (result, x, index) => result + x \* Ws[index],  
 0  
 )  
  
export const *getDiff* = (A, B) => A.reduce(  
 (res, a, indexB) => {  
 if (a !== B[indexB])  
 return res + 1;  
 return res;  
 },  
 0  
)  
  
export const hemmingTeacher = ({  
 id: 1,  
 name: 'Сеть Хемминга',  
 epsilon: 0.1  
})  
  
export const teachers = [hemmingTeacher]  
  
export const $teacher = *createStore*(hemmingTeacher)  
  
export const changeTeacher = *createEvent*()  
  
export const teachEvent = *createEvent*()  
export const teachFx = *createEffect*(teachEvent)  
export const teach = *attach*({  
 effect: teachFx,  
 source: *combine*(  
 $M, $weight, $recognize,  
 (M, weight, recognize) => ({M, weight, recognize})  
 ),  
 mapParams: (\_, data) => (data)  
})

import {  
 $teacher,  
 changeTeacher, *getS*,  
 teachEvent,  
} from './index'  
import {setWeights} from "../weight";  
import {setSs} from "../presets";  
  
const handleChangeTeacher = (\_, teacher) => teacher  
const handleTeach = (teacher, {M, weight, recognize}) => {  
 const W = weight.map(  
 (w, indexM) =>  
 w.map(  
 (\_, index) =>  
 (  
 index  
 ? M[indexM].x[index - 1]  
 : M[indexM].x.length  
 ) / 2  
 )  
 )  
  
 setWeights(W)  
 setSs(  
 M  
 .map(  
 ({x}) => recognize.recognize([1, ...x], W, teacher.epsilon).Y  
 )  
 )  
}  
  
$teacher  
 .on(changeTeacher, handleChangeTeacher)  
 .on(teachEvent, handleTeach)

import {*getS*} from "../teacher";  
import {*createEvent*, *createStore*} from "effector";  
import {setCountOperations} from "../info";  
  
export function *average*(nums) {  
 return nums.reduce((a, b) => (a + b)) / nums.length;  
}  
  
export function *equalArray*(a, b){  
 return JSON.stringify(a) === JSON.stringify(b)  
}  
  
export const recognizeFunctions = [  
 {  
 recognize: (xs, ws, epsilon) => {  
 const Z = ws.map(w => *getS*(xs, w))  
  
 let t = Z.map(z => z / 25)  
 let tNext = []  
 let count = 0  
 while (true){  
 count++;  
 tNext = t.map(  
 (Ui, i) => {  
 const sum = epsilon \* t.reduce((sum, Uj, j) => sum + (i === j ? 0 : Uj), 0)  
 const Uin = Ui - sum  
 return Uin > 0 ? Uin : 0  
 }  
 )  
  
 if (*equalArray*(t, tNext))  
 break;  
 else  
 t = [...tNext]  
 }  
  
 setCountOperations(count)  
 return {  
 Z,  
 Y: tNext  
 }  
 },  
 id: 1,  
 name: 'Сеть Хемминга'  
 },  
  
]  
  
export const $recognize = *createStore*(recognizeFunctions[0])  
  
export const setRecognize = *createEvent*()

import {*createEvent*, *createStore*} from "effector";  
  
  
export const $M = *createStore*([  
 {  
 x: [  
 1, -1, -1, -1, 1,  
 -1, 1, -1, 1, -1,  
 -1, 1, 1, 1, -1,  
 -1, 1, -1, 1, -1,  
 1, -1, -1, -1, 1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, -1, -1, -1, 1,  
 -1, 1, 1, 1, -1,  
 -1, 1, 1, 1, -1,  
 -1, 1, 1, 1, -1,  
 1, -1, -1, -1, 1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 -1, -1, 1, -1, -1,  
 -1, 1, -1, 1, -1,  
 1, -1, -1, -1, 1,  
 1, 1, 1, 1, 1,  
 1, -1, -1, -1, 1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 -1, -1, 1, -1, -1,  
 -1, 1, -1, 1, -1,  
 -1, 1, -1, 1, -1,  
 -1, 1, 1, 1, -1,  
 -1, 1, -1, 1, -1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, 1, 1, 1, 1,  
 1, -1, -1, -1, 1,  
 1, 1, 1, 1, 1,  
 1, -1, -1, -1, -1,  
 1, -1, -1, -1, -1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, 1, 1, 1, 1,  
 1, 1, -1, 1, 1,  
 1, 1, 1, 1, 1,  
 1, 1, -1, -1, -1,  
 1, 1, -1, -1, -1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, -1, -1, -1, 1,  
 1, -1, -1, 1, 1,  
 1, -1, 1, -1, 1,  
 1, 1, -1, -1, 1,  
 1, -1, -1, -1, 1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, 1, -1, 1, 1,  
 1, -1, -1, 1, 1,  
 1, -1, 1, -1, 1,  
 1, 1, -1, -1, 1,  
 1, 1, -1, 1, 1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, 1, 1, 1, 1,  
 -1, 1, 1, 1, -1,  
 -1, -1, 1, -1, -1,  
 -1, -1, 1, -1, -1,  
 -1, 1, 1, 1, -1,  
 ],  
 S: [],  
 diff: 0  
 },  
 {  
 x: [  
 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1,  
 -1, -1, 1, -1, -1,  
 -1, -1, 1, -1, -1,  
 -1, -1, 1, -1, -1,  
 ],  
 S: [],  
 diff: 0  
 },  
])  
  
  
export const setSs = *createEvent*()  
export const setDiffs = *createEvent*()