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 №8

тема: «Восстановление неполных и искаженных данных с помощью нейрокомпьютерной сети Хопфилда.»

Выполнил: ст. группы МИВТ-221

Харитонов Сергей Дмитриевич

Белгород 2022 г.

**Лабораторная работа №8**

**Восстановление неполных и искаженных данных с помощью нейрокомпьютерной сети Хопфилда.**

**Цель работы:** Приобретение и закрепление знаний, получение практических навыков работы с нейрокомпьютерной сетью Хопфилда, исследование возможностей сети по восстановлению неполных и искаженных данных.

Реализация сети

В рамках лабораторной работы была организована сеть Хопфилда на основе кодовой базы предыдущих лабораторных.

Была реализована функция инициализации весов в соответствии с алгоритмом.

const Xs = M.map(({x}) => x);  
const newW = weight.map(  
 (wRow, i) =>  
 wRow.map(  
 (\_, j) => {  
 if (i === j)  
 return 0.0  
 else  
 return Xs.reduce(  
 (w, x) => w + x[i] \* x[j],  
 0.0  
 )  
 }  
 )  
)  
setWeights(newW)

А так же алгоритм проход нейронной сети на основе ранее инициализированных весов и пользовательского ввода.

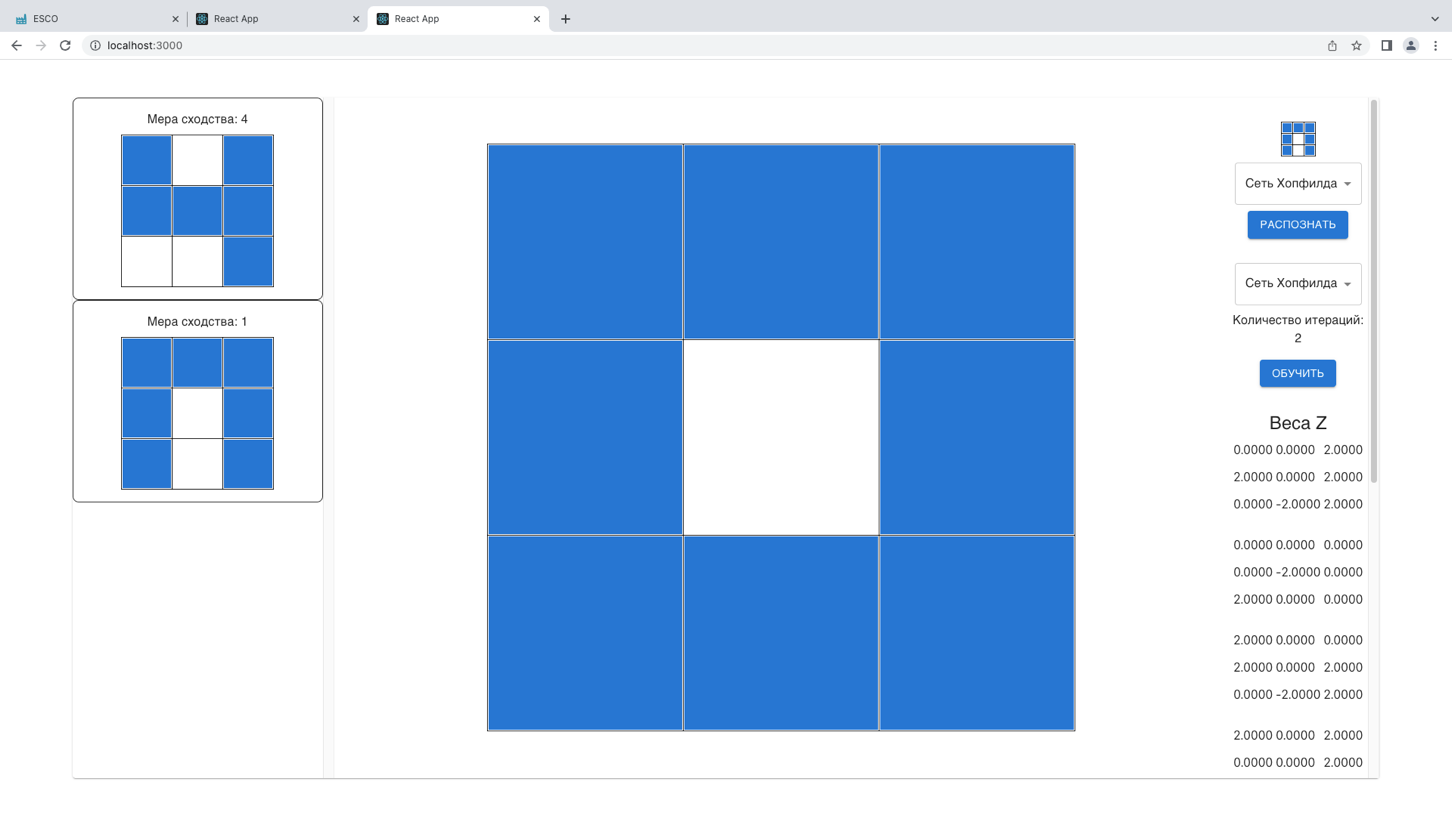
let prevY = [...xs];  
let nextY = [...xs];  
  
let countOperations = 0;  
  
const checkEpsilon = (prevW, nextW) => {  
 return prevW.reduce(  
 (result, prevWr, indexR) =>  
 result || (prevWr !== nextW[indexR]),  
 false  
 )  
}  
  
do {  
 countOperations++  
 prevY = [...nextY]  
 nextY = ws.map((w, t) =>{  
 const S = *getS*(prevY, w)  
 if (S > 0) return 1  
 if ((Math.abs(S) < 1e-10)) return prevY[t]  
 if (S < 0) return -1  
 }  
 )  
  
  
} while (checkEpsilon(prevY, nextY))  
setCountOperations(countOperations)  
return nextY

Запуск на примере из лекции

Для проверки корректности реализованного алгоритма был осуществлен запуск на примере из лекционного материала.



Результат одной итерации прохождения нейронной сети

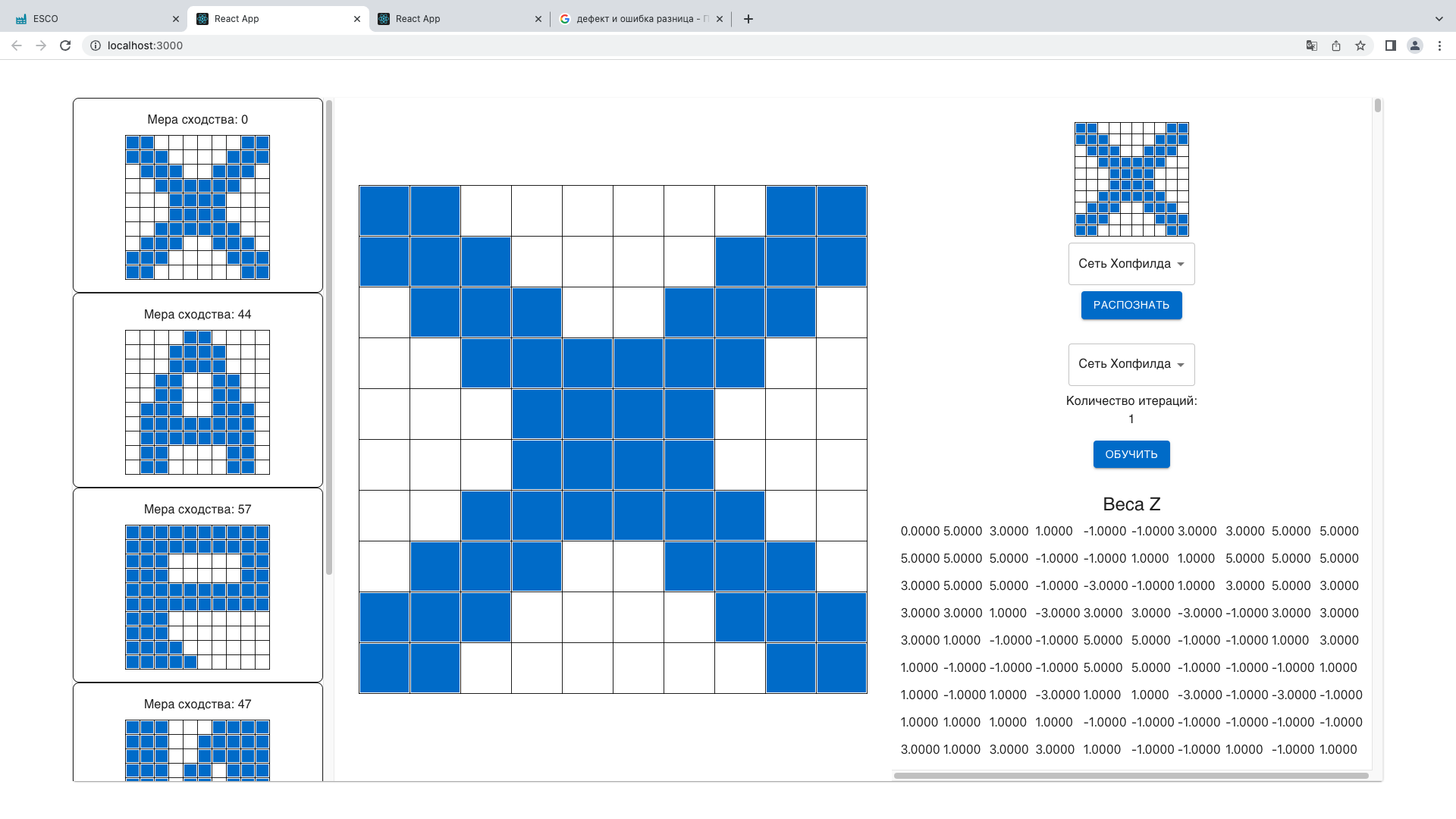


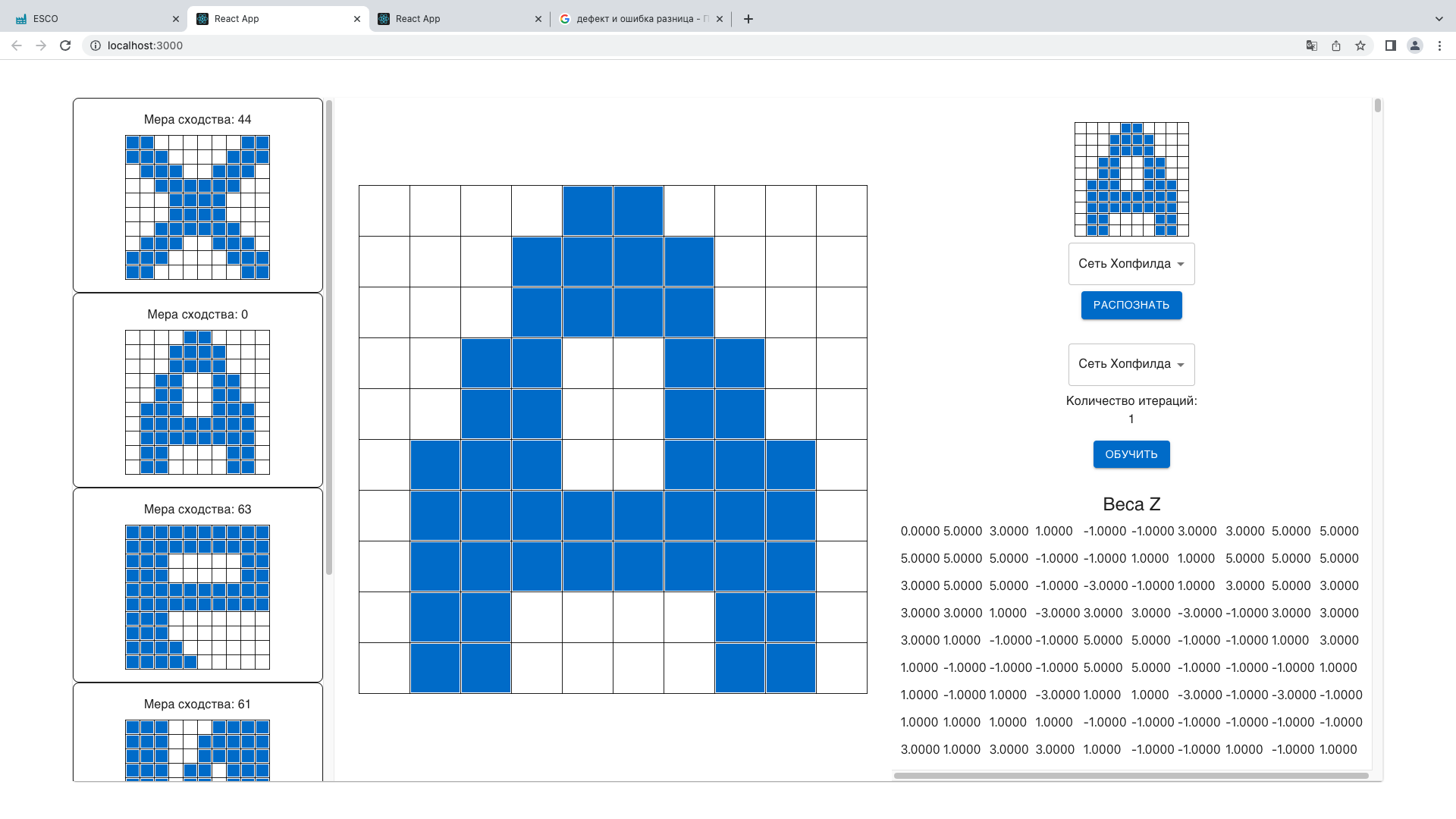
Отображение в интерфейсе

Как видно из скриншотов в результате 2х итераций (1 для приведение к прототипу, 2 для удостоверения что выход не меняется) результат аналогичен приведенному в лекционном материале, что свидетельствует о верности реализации, проинициализированные веса так же равны весам из примера.

Экранные копии изображений

Для проведения исследований были разработаны экранные формы букв размером 10 на 10 пикселей: Х,А,Р,И,Т



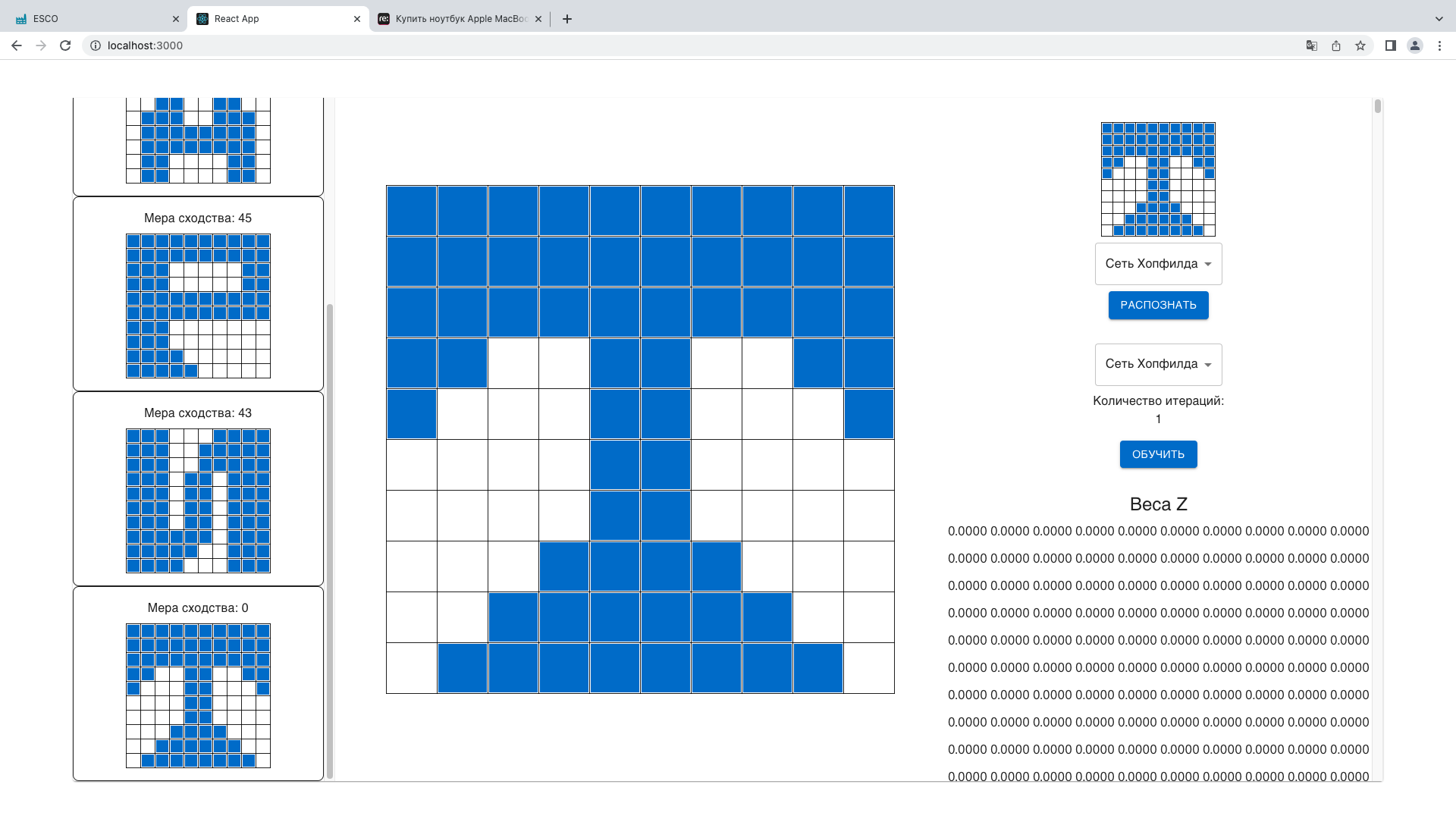


Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание



Исследование результатов работы сети

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемый образ | Процент зашумления образа | Вид искаженного образа | Результат востановления |
|  | 10% | Изображение выглядит как синий, с плиткой  Автоматически созданное описание |  |
| 20% | Изображение выглядит как внутренний, с плиткой, синий  Автоматически созданное описание |  |
| 30% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание |  |
| 35% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание |  |
| 40% |  |  |
| 45% |  |  |
| 50% |  | Изображение выглядит как седзи  Автоматически созданное описание |
| 60% |  |  |
| 70% |  |  |
| 80% | Изображение выглядит как с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, синий, кроссворд, с плиткой  Автоматически созданное описание |
| 90% |  |  |
| 100% |  |  |

Вывод: как видно из результатов исследования до 45% зашумленности нейронная сеть верно определяло оригинал изображения, что является достаточно высоким результатом. Так же после перехода границы в 50% зашумленности, нейронная сеть начала выдавать на выходе инвертированное изображение оригинала.

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемый образ | Процент зашумления образа | Вид искаженного образа | Результат востановления |
|  | 10% |  | Изображение выглядит как седзи, кроссворд, здание  Автоматически созданное описание |
| 20% |  | Изображение выглядит как седзи, кроссворд, здание  Автоматически созданное описание |
| 30% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, кроссворд, здание  Автоматически созданное описание |
| 35% | Изображение выглядит как с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи, здание, кроссворд, с плиткой  Автоматически созданное описание |
| 40% | Изображение выглядит как с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи, здание, коллекция картинок  Автоматически созданное описание |
| 45% | Изображение выглядит как с плиткой, синий, плитка  Автоматически созданное описание |  |
| 50% |  |  |
| 60% |  | Изображение выглядит как седзи, кроссворд, здание, с плиткой  Автоматически созданное описание |
| 70% |  | Изображение выглядит как седзи, здание  Автоматически созданное описание |
| 80% |  | Изображение выглядит как седзи, здание, кроссворд, с плиткой  Автоматически созданное описание |
| 90% |  | Изображение выглядит как седзи, кроссворд, здание, с плиткой  Автоматически созданное описание |
| 100% |  | Изображение выглядит как седзи, здание, кроссворд  Автоматически созданное описание |

Вывод: аналогично букве Х трудности в распознавании начались около 40%-45%, однако в отличии от нее в переходный период около 50% нейронная сеть определяла входное изображение как букву Х, затем как и в предыдущем случае начало проявляться инвертированное изображение

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемый образ | Процент зашумления образа | Вид искаженного образа | Результат востановления |
|  | 10% |  | Изображение выглядит как седзи  Автоматически созданное описание |
| 20% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, кроссворд  Автоматически созданное описание |
| 30% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи  Автоматически созданное описание |
| 35% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, кроссворд, синий, с плиткой  Автоматически созданное описание |
| 40% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи  Автоматически созданное описание |
| 45% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как стол  Автоматически созданное описание |
| 50% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как стол  Автоматически созданное описание |
| 60% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи  Автоматически созданное описание |
| 70% |  | Изображение выглядит как седзи  Автоматически созданное описание |
| 80% |  |  |
| 90% |  |  |
| 100% |  |  |

Вывод: В отличии от предыдущих изображений проблемы с качеством распознавания начались уже на 35%, соответственно сдвинулась и граница приведения к инверсированному изображению до 80%.

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемый образ | Процент зашумления образа | Вид искаженного образа | Результат востановления |
|  | 10% |  | Изображение выглядит как стол  Автоматически созданное описание |
| 20% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как стол  Автоматически созданное описание |
| 30% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как стол  Автоматически созданное описание |
| 35% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, кроссворд  Автоматически созданное описание |
| 40% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как стол  Автоматически созданное описание |
| 45% |  | Изображение выглядит как кроссворд, с плиткой  Автоматически созданное описание |
| 50% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, кроссворд  Автоматически созданное описание |
| 60% | Изображение выглядит как с плиткой  Автоматически созданное описание | Изображение выглядит как седзи  Автоматически созданное описание |
| 70% |  | Изображение выглядит как седзи, кроссворд, коллекция картинок  Автоматически созданное описание |
| 80% |  | Изображение выглядит как седзи, кроссворд, здание, коллекция картинок  Автоматически созданное описание |
| 90% |  | Изображение выглядит как седзи, кроссворд, коллекция картинок  Автоматически созданное описание |
| 100% |  | Изображение выглядит как седзи, кроссворд, коллекция картинок  Автоматически созданное описание |

Вывод: Результаты такие же как для буквы Р

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемый образ | Процент зашумления образа | Вид искаженного образа | Результат востановления |
|  | 10% |  | Изображение выглядит как седзи, кроссворд  Автоматически созданное описание |
| 20% | Изображение выглядит как внутренний, синий, с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи, кроссворд  Автоматически созданное описание |
| 30% | Изображение выглядит как внутренний, синий, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, здание, с плиткой, синий  Автоматически созданное описание |
| 35% | Изображение выглядит как внутренний, синий, с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи  Автоматически созданное описание |
| 40% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как седзи, с плиткой  Автоматически созданное описание |
| 45% | Изображение выглядит как внутренний, с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи, здание  Автоматически созданное описание |
| 50% | Изображение выглядит как внутренний, с плиткой  Автоматически созданное описание | Изображение выглядит как стол  Автоматически созданное описание |
| 60% | Изображение выглядит как внутренний, с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи, здание  Автоматически созданное описание |
| 70% | Изображение выглядит как внутренний, с плиткой, плитка  Автоматически созданное описание | Изображение выглядит как седзи, здание  Автоматически созданное описание |
| 80% |  | Изображение выглядит как седзи, здание  Автоматически созданное описание |
| 90% |  | Изображение выглядит как седзи, здание  Автоматически созданное описание |
| 100% |  |  |

Вывод: Результат аналогичен с предыдущими буквами

Вывод: в результате выполнения работы было выяснено, что сеть Хопфилда вполне эффективный инструмент для восстановления образов, на исследуемой выборке при зашумлении до 30-45% восстанавливает входное изображение до оригинала, при 50% зашумленности зачастую выбирает некорректный оригинальный образ для восстановления, после 60-70% изображение восстанавливается до обратного оригиналу.

Листинг 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 kohTeacher = ({  
 id: 1,  
 name: 'Сеть Хопфилда',  
})  
  
export const teachers = [kohTeacher]  
  
export const $teacher = *createStore*(kohTeacher)  
  
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";  
import {setCountOperations} from "../info";  
  
const handleChangeTeacher = (\_, teacher) => teacher  
const handleTeach = (teacher, {M, weight, recognize}) => {  
 const Xs = M.map(({x}) => x);  
 const newW = weight.map(  
 (wRow, i) =>  
 wRow.map(  
 (\_, j) => {  
 if (i === j)  
 return 0.0  
 else  
 return Xs.reduce(  
 (w, x) => w + x[i] \* x[j],  
 0.0  
 )  
 }  
 )  
 )  
 setWeights(newW)  
}  
  
$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) => {  
  
 let prevY = [...xs];  
 let nextY = [...xs];  
  
 let countOperations = 0;  
  
 const checkEpsilon = (prevW, nextW) => {  
 return prevW.reduce(  
 (result, prevWr, indexR) =>  
 result || (prevWr !== nextW[indexR]),  
 false  
 )  
 }  
  
 do {  
 countOperations++  
 prevY = [...nextY]  
 nextY = ws.map((w, t) =>{  
 const S = *getS*(prevY, w)  
 if (S > 0) return 1  
 if ((Math.abs(S) < 1e-10)) return prevY[t]  
 if (S < 0) return -1  
 }  
 )  
  
  
 } while (checkEpsilon(prevY, nextY))  
 setCountOperations(countOperations)  
 return nextY  
 },  
 id: 1,  
 name: 'Сеть Хопфилда'  
 },  
  
]  
  
export const $recognize = *createStore*(recognizeFunctions[0])  
  
export const setRecognize = *createEvent*()

import {*createEvent*, *createStore*} from "effector";  
  
const getRandomW = () => (Math.random() - 0.5)  
const getRandomWs = () => [  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
]  
  
const getRandomExWs = () => [  
 0, 0, 0,  
 0, 0, 0,  
 0, 0, 0,  
]  
  
export const zeroWeight = [  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
].map(getRandomWs)  
export const $weight = *createStore*(zeroWeight)  
  
export const setWeights = *createEvent*()

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