CREATE A PROGRAM THAT CAN RECOGNIZE DRAWINGS DRAWN BY THE USER USING A LEARNING NEURAL NETWORK IN THE OBJECT-ORIENTED JAVA LANGUAGE

Sinelnik K.E.[1], Limanova N.I.[2] (Russian Federation)

[1]Sinelnik Karim Eldarovich – student;
[2]Limanova Natalia Igorevna – Scientific Supervisor, Head of the Department,
DEPARTMENT OF INFORMATION SYSTEMS AND TECHNOLOGY,
VOLGA REGION STATE UNIVERSITY OF TELECOMMUNICATIONS AND INFORMATICS,
SAMARA, RUSSIAN FEDERATION

Abstract: *this article describes how using the object-oriented Java language, as well as using a library with a neural network, a program was created that can recognize the drawings that the user drew in the program window. The essence lies precisely in the learning process of the program, how and with what it determines what is drawn, the article pays much attention to the learning process of a neural network. The program code and screenshots also presented, which show how the application works. In conclusion, experiments presented in which it is seen how the neural network copes with the difficulties encountered during the issuance of the result to the user of this program.*
Keywords: *neural network, drawings, learning, programming language, java, development.*

СОЗДАТЬ ПРОГРАММУ, КОТОРАЯ СМОЖЕТ РАСПОЗНАВАТЬ РИСУНКИ, НАРИСОВАННЫЕ ПОЛЬЗОВАТЕЛЕМ С ПОМОЩЬЮ ОБУЧАЮЩЕЙСЯ НЕЙРОННОЙ СЕТИ НА ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ЯЗЫКЕ JAVA

Синельник К.Э.[1], Лиманова Н.И.[2] (Российская Федерация)

[1]Синельник Карим Эльдарович – студент,
Институт информатики и кибернетики;
[2]Лиманова Наталья Игоревна – научный руководитель, заведующая кафедрой,
кафедра Информационных систем и технологии,
Поволжский государственный университет телекоммуникаций и информатики,
г. Самара

Аннотация: *в данной статье расказано, как с помощью объектно-ориентированного языка Java, а так же использования библиотеки с нейронной сетью, была созданна программа, которая может распознавать рисунки, которые нарисовал пользователь в окне программы. Суть заключается именно в процессе обучения программы, как и с помощью чего она определяет что нарисованно, в статье уделено много внимания процессу*

обучения нейронной сети. Так же представлен код программы и скриншоты на которых виден принцып работы приложения. В заключении представленны эксперементы в которых видно, как нейросеть справляется с возникшими трудностями во время выдачи результата пользователю данной программы.
Ключевые слова: *нейронная сеть, рисунки, обучение, язык программирования, java, разработка.*

Программа была выполнена с помощью языка программирования java script для большей скорости работы и производительности, поддержке скриптов всеми популярными браузерами, понятному синтаксису для новичка.

В ходе работы мы использовали brain js-**это** библиотека с открытым кодом на языке **JavaScript**, используемая для запуска и обработки нейронных сетей.

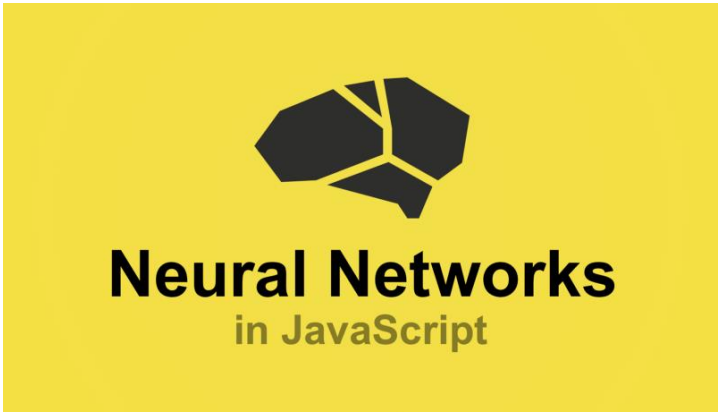

Рис.1 библиотека Brain js.

Brain.js используется, как правило, с Node.js или с браузером на стороне клиента для тренировки моделей машинного обучения. Она позволяет легко создавать нейронные сети, а затем обучать их на основе входных / выходных данных.

Также имеются другие библиотеки для нейронной сети такие как:

- **Limdu.js**

Это платформа машинного обучения, используемая для Node.js.

Limdu.js идеально подходит для виртуальных собеседников (чат-ботов), обработки естественного языка и других диалоговых систем.stdLib.

Рис.2. библиотека stdLib.

Эта библиотека JavaScript используется для создания продвинутых статистических моделей и библиотек машинного обучения. Кроме того, она может использоваться в графических средствах отображения информации для разведочного анализа данных, а также визуализации данных. Но для нашей работы была выбрана brain js.

Что такое нейронные сети?

Нейронные сети возникли из исследований в области искусственного интеллекта, а именно, из попыток воспроизвести способность биологических нервных систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга. В простейшем случае она состоит из нескольких соединенных между собой нейронов.

Математический нейрон

Несложный автомат, преобразующий входные сигналы в результирующий выходной сигнал.

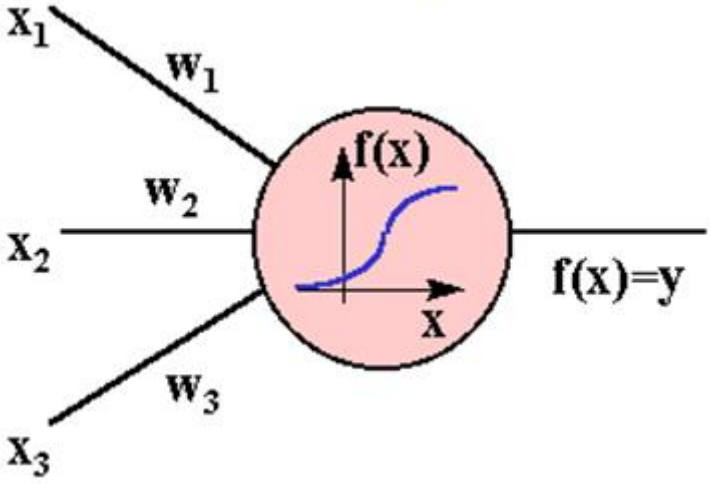

Рис.3. Модель математического нейрона.

Сигналы x1, x2, x3 … xn, поступая на вход, преобразуются линейным образом, т.е. к телу нейрона поступают силы: w1x1, w2x2, w3x3 … wnxn, где wi — веса соответствующих сигналов. Нейрон суммирует эти сигналы, затем применяет к сумме некоторую функцию f(x) и выдаёт полученный выходной сигнал. y. В качестве функции f(x) чаще всего используется сигмоидная или пороговая функции.

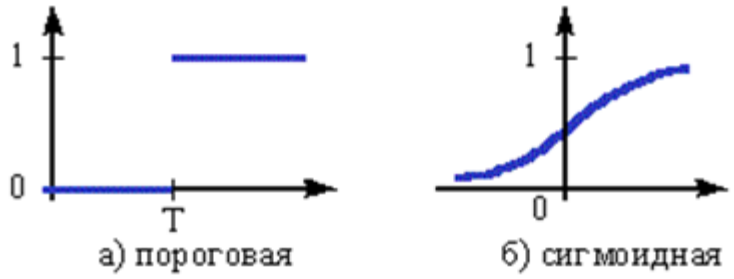

Рис.4. Виды функций полученного выходного сигнала.

Пороговая функция может принимать только два дискретных значения 0 или 1. Смена значения функции происходит при переходе через заданный порог T.+

Сигмоидная — непрерывная функция, может принимать бесконечно много значений в диапазоне от 0 до 1.

Типы нейронных сетей

За период развития, нейронные сети поделились на множество типов, которые переплетаются между собой в различных задачах. На данный момент сложно классифицировать какую-либо сеть только по одному признаку. Это можно сделать по принципу применения, типу входной информации, характеру обучения, характеру связей, сфере применения.

Сверточные

Один из популярнейших типов сети, часто используемый для распознавания той или иной информации в фотографиях и видео, обработке языка, системах для рекомендаций. Основные характеристики:

1. Отличная масштабируемость – проводят распознания образов любого разрешения (какое бы не было оно большое).
2. Использование объемных трехмерных нейронов – внутри слоя, нейроны связаны малым полем, именуемы рецептивным слоем.
3. Механизм пространственной локализации – соседние слои нейронов связаны таким механизмом, за счет чего обеспечивается работа нелинейных фильтров и охват все большего числа пикселей графического изображения.

Идея сложной системы этого типа нейросети возникла при тщательном изучении зрительной коры, которая в больших полушариях мозга отвечает за обработку визуальной составляющей. Основной критерий выбора в пользу свёрточного типа – она в составе технологий глубокого обучения. Схожий тип с перцептроном, но разница в том, что здесь используется ограниченная матрица весов, сдвигаемая по обрабатываемому слою, вместо полносвязной нейронной сети.

Рекуррентные

Этот тип нейросети, в котором связи между элементами могут обрабатывать серии различных событий во времени или работать с последовательными цепочками в пространстве. Такой тип часто применяют там, где что-то целое разбито на куски. Например, распознавание речи или рукописного текста. От нее пошло множество видов сетей, в том числе Хопфилда, Элмана и Джордана.

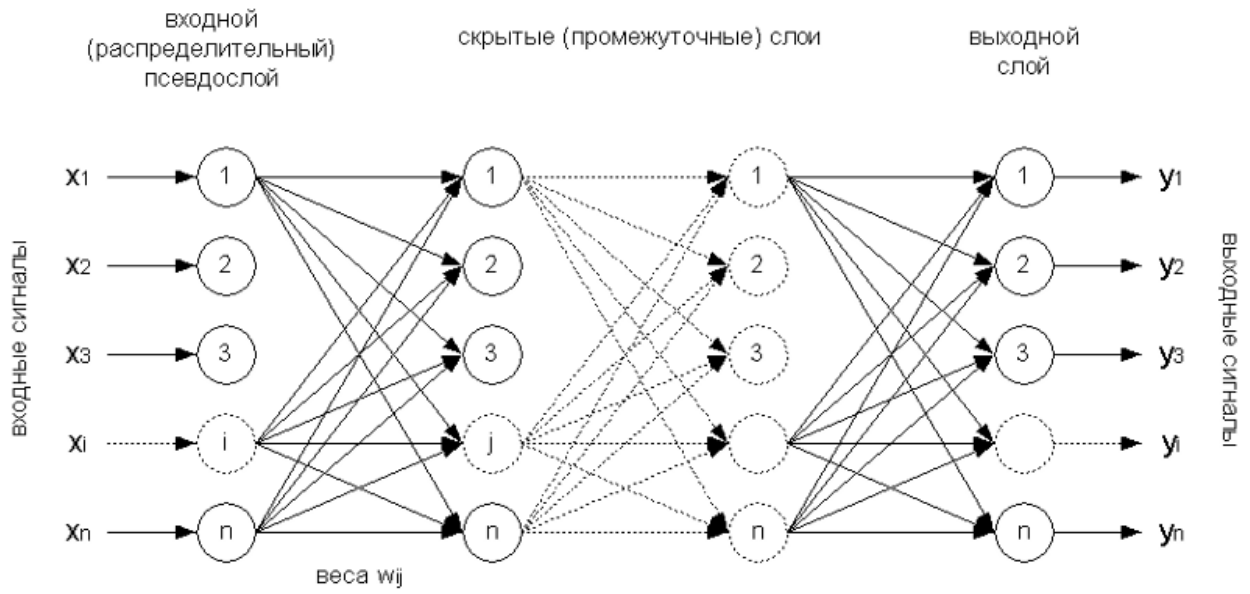

Рис.5. Архитектура нейроной сети - Perceptron.

Архитектура нейронной сети может быть разной, мы рассмотрим одну из простых реализаций нейронной сети – **Perceptron.**

Есть слой входных нейронов (где информация поступает извне), слой выходных нейронов (откуда можно взять результат) и ряд, так называемых, скрытых слоев между ними. Нейроны могут быть расположены в несколько слоёв. Каждая связь между нейронами имеет свой вес **Wij**.

Входные и выходные сигналы

Перед тем, как подавать сигналы на нейроны входящего слоя сети нам их нужно нормализовать. Нормализация входных данных — это процесс, при котором все входные данные проходят процесс «выравнивания», т.е. приведения к интервалу [0,1] или [-1,1]. Если не провести нормализацию, то входные данные будут оказывать дополнительное влияние на нейрон, что приведет к неверным решениям. Другими словами, как можно сравнивать величины разных порядков?

На нейронах выходного слоя у нас тоже не будет чистой «1» или «0», это нормально. Есть некий порог, при котором мы будем считать, что получили «1» или «0». Про интерпретацию результатов поговорим позже.

Обучение нейронной сети

Для конструирования процесса обучения, прежде всего, необходимо иметь модель внешней среды, в которой функционирует нейронная сеть - знать доступную для сети информацию.

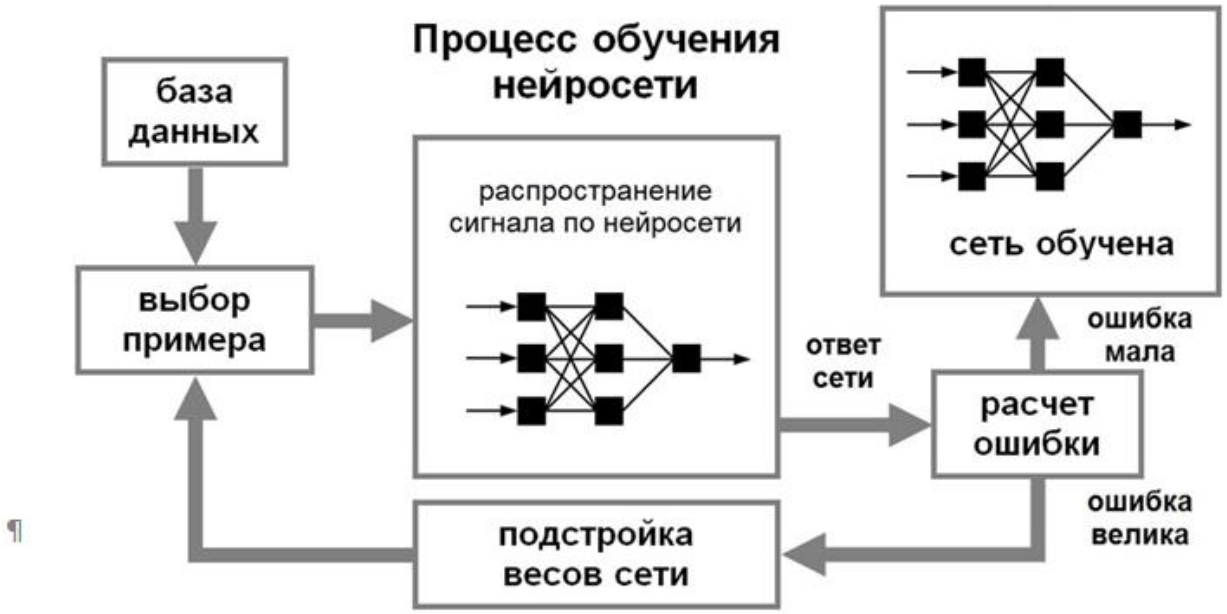

Рис.6. Схема процесса обучения нейронной сети.

Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети - какие правила обучения управляют процессом настройки. Алгоритм обучения означает процедуру, в которой используются правила обучения для настройки весов.

Существуют три парадигмы обучения: "с учителем", "без учителя" (самообучение) и смешанная. В первом случае нейронная сеть располагает правильными ответами (выходами сети) на каждый входной пример. Веса настраиваются так, чтобы сеть производила ответы как можно более близкие к известным правильным ответам. Усиленный вариант обучения с учителем предполагает, что известна только критическая оценка правильности выхода нейронной сети, но не сами правильные значения выхода. Обучение без учителя не требует знания правильных ответов на каждый пример обучающей выборки. В этом случае раскрывается внутренняя структура данных или Корреляции между образцами в системе данных, что позволяет распределить образцы по категориям. При смешанном обучении часть весов определяется посредством обучения с учителем, в то время как остальная получается с помощью самообучения.

Теория обучения рассматривает три фундаментальных свойства, связанных с обучением по примерам: емкость, сложность образцов и вычислительная сложность. Под емкостью понимается, сколько образцов может запомнить сеть, и какие функции и границы принятия решений могут быть на ней сформированы. Сложность образцов определяет число обучающих примеров, необходимых для достижения способности сети к обобщению. Слишком малое число примеров может вызвать «переобученность» сети, когда она хорошо функционирует на примерах обучающей выборки, но плохо – на тестовых примерах, подчиненных тому же статистическому распределению. Известны 4 основных типа правил

обучения: коррекция по ошибке, машина Больцмана, правило Хебба и обучение методом соревнования.

Правило коррекции по ошибке. При обучении с учителем для каждого входного примера задан желаемый выход d. Реальный выход сети y может не совпадать с желаемым. Принцип коррекции по ошибке при обучении состоит в использовании сигнала (d-y) для модификации весов, обеспечивающей постепенное уменьшение ошибки. Обучение имеет место только в случае, когда перцептрон ошибается. Известны различные модификации этого алгоритма обучения.

Обучение Больцмана. Представляет собой стохастическое правило обучения, которое следует из информационных теоретических и термодинамических принципов. Целью обучения Больцмана является такая настройка весовых коэффициентов, при которой состояния видимых нейронов удовлетворяют желаемому распределению вероятностей. Обучение Больцмана может рассматриваться как специальный случай коррекции по ошибке, в котором под ошибкой понимается расхождение Корреляций состояний в двух режимах Правило Хебба. Самым старым обучающим правилом является постулат обучения Хебба. Хебб опирался на следующие нейрофизиологические наблюдения: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи возрастает. Важной особенностью этого правила является то, что изменение синаптического веса зависит только от активности нейронов, которые связаны данным синапсом. Это существенно упрощает цепи обучения в реализации VLSI.

Обучение методом соревнования. В отличие от обучения Хебба, в котором множество выходных нейронов могут возбуждаться одновременно, при соревновательном обучении выходные нейроны соревнуются между собой за активизацию. Это явление известно как правило "победитель берет все". Подобное обучение имеет место в биологических нейронных сетях.

Обучение посредством соревнования позволяет кластеризовать входные данные: подобные примеры группируются сетью в соответствии с корреляциями и представляются одним элементом. При обучении модифицируются только веса "победившего" нейрона. Эффект этого правила достигается за счет такого изменения сохраненного в сети образца (вектора весов связей победившего нейрона), при котором он становится чуть ближе к входному примеру.

Код программы

Для работы с нейронной сетью были созданы следующие файлы.

- **Index.html**- в данной файле будет происходить основная работа.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Нейросеть</title>
    <link rel="stylesheet" href="style.css">// подключение стилей
  </head>
  <body>
     //подключение canvas для создания рисунков
    <canvas id="canvas"></canvas>
     //использование библиотеки brain.js
    <script src="brain.js"></script>
     // подключение app.js для работы кода
    <script src="app.js"></script>
  </body>
</html>
```

Рис.7. Index.html

- **Style.css** - для создания внешнего вида и цвета полотна.

```css
body {
  background-color: #000000;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

#canvas {
  background-color: #fff;
}
```

Рис.8. Style.css

- **App.js** – код java script.

```
//создание элемента canvas
const canvas = document.getElementById( elementId: 'canvas')
// создание класса paint для рисования
class Paint {
// создание конструктора для определения размеров полотна
  constructor(width, height, pixelSize, color) {
    canvas.width = width
    canvas.height = height
    this.canvas = canvas
    this.ctx = this.canvas.getContext( contextId: '2d')
// отвечают за цвет и ширину линии
    this.pixelSize = pixelSize
    this.color = color
    this.isMouseDown = false
  }
//данный метод создан для того,чтобы линия рисовалась окружностями
  drawCircle(x, y, diameter = this.pixelSize, color = this.color) {
    this.ctx.beginPath()
    this.ctx.fillStyle = color
    this.ctx.arc(x, y,  radius: diameter / 2,  startAngle: 0,  endAngle: Math.PI * 2)
    this.ctx.fill()
  }
// данный метод -это линия которая связывает окружности
  drawConnectingLine(x, y, lineWidth = this.pixelSize, color = this.color) {
    this.ctx.lineTo(x, y)
    this.ctx.lineWidth = lineWidth
    this.ctx.strokeStyle = color
    this.ctx.stroke()
  }
//запоминание позиции
  storePosition(x, y) {
    this.ctx.beginPath()
    this.ctx.moveTo(x, y)
  }
// создание линии для сетки,чтобы нейросеть запоминала изображение
  drawLine(x1, y1, x2, y2, lineWidth : number  = 2, color : string  = '#00bfff') {
    this.ctx.beginPath()
    this.ctx.moveTo(x1, y1)
    this.ctx.lineWidth = lineWidth
    this.ctx.strokeStyle = color
    this.ctx.lineTo(x2, y2)
    this.ctx.stroke()
```

Рис.9. App.js

```
    this.ctx.stroke()
  }
 // рисование ячейки
  drawCell(x1, y1, size = this.pixelSize, color : string  = '#00bfff') {
    this.ctx.beginPath()
    this.ctx.fillStyle = color
    this.ctx.rect(x1, y1, size, size)
    this.ctx.fill()
  }
// создание числового представления
 //если в сетке клетки не закрашены,то ставится 0
//если в сетке клекта закрашена,то ставится 1
  toNumericalRepresentation(shouldBeDrawn) {
    const numericalRepresentation = []
    let cellsToDraw = []

    for (let x = 0; x < this.canvas.width; x += this.pixelSize) {
      for (let y = 0; y < this.canvas.height; y += this.pixelSize) {
        const cell = this.ctx.getImageData(x, y, this.pixelSize, this.pixelSize)
        let isFilled = false

        for(let i = 0; i < cell.data.length; i += 10) {
          if (cell.data[i] !== 0) {
            isFilled = true
            break
          }
        }

        if (isFilled) {
          numericalRepresentation.push(1)
          cellsToDraw.push({ x, y })
        } else {
          numericalRepresentation.push(0)
        }
      }
    }

    if (shouldBeDrawn) {
      cellsToDraw.forEach((cell) => this.drawCell(cell.x, cell.y))
    }

    return numericalRepresentation
```

Рис.10. App.js

```
    if (shouldBeDrawn) {
      cellsToDraw.forEach((cell) => this.drawCell(cell.x, cell.y))
    }

    return numericalRepresentation
  }
// создание сетки
  drawGrid() {
    for (let x = 0; x < this.canvas.width; x += this.pixelSize) {
      this.drawLine(x,  y1: 0, x, this.canvas.height)
    }

    for (let y = 0; y < this.canvas.height; y += this.pixelSize) {
      this.drawLine( x1: 0, y, this.canvas.width, y)
    }
  }

  clear() {
    this.ctx.clearRect( x: 0,  y: 0, this.canvas.width, this.canvas.height)
  }
// задание параметров для рисования
  run() {
    this.canvas.addEventListener( type: 'mousedown',  listener: () => {
      this.isMouseDown = true
      this.ctx.beginPath()
    })

    this.canvas.addEventListener( type: 'mouseup',  listener: () => {
      this.isMouseDown = false
    })

    this.canvas.addEventListener( type: 'mousemove',  listener: (e : MouseEvent ) => {
      if (this.isMouseDown) {
        const x = e.offsetX
        const y = e.offsetY

        this.drawConnectingLine(x, y)
        this.drawCircle(x, y)
        this.storePosition(x, y)
      }
    })
  }
```

Рис.11. App.js

```
}
// присвоение размеров полотна и ширину,цвета линии
const paint = new Paint( width: 600,  height: 400,  pixelSize: 15,  color: 'purple')
paint.run()
//вывести ответ
const statistic = []
const addToStatistic = (name, numericalRepresentation) => {
  statistic.push({
    input: numericalRepresentation,
    output: {
      [name]: 1
    }
  })
}
// нажатие на кнопку C - стирается изображение
document.addEventListener( type: 'keypress',  listener: ({ code : string }) => {
  if (code === 'KeyC') {
    paint.clear()
  }
// нажатие на кнопку V - нейросеть запоминает изоображение
  if (code === 'KeyV') {
    const numericalRepresentation = paint.toNumericalRepresentation( shouldBeDrawn: true)
    addToStatistic(prompt( message: 'ЧТО ИЗОБРАЖЕНО?'), numericalRepresentation)
  }
// нажатие на кнопку B -запускается нейросеть и угадывает изоображение
  if (code === 'KeyB') {
    const neuralNetwork = new brain.NeuralNetwork()
    neuralNetwork.train(statistic, { log: true })
    const numericalRepresentation = paint.toNumericalRepresentation()
    const result = brain.likely(numericalRepresentation, neuralNetwork)

    if (confirm(`НА РИСУНКЕ ИЗООБРАЖЕН  ${result}. Я ПРАВ?`)) {
      addToStatistic(result, numericalRepresentation)
    } else {
      addToStatistic(prompt( message: 'ЕСЛИ НЕТ,ТО ЧТО ИЗОБРАЖЕНО?'), numericalRepresentation)
    }

    console.log(statistic)
  }
})
```

Рис.12. App.js

Для анализа статистических данных и последующем угадыванием изображения, самым верным будет разбиения изображения(полотна) на сетку, которая будет проверять каждый пиксель, если пиксель закрашен, то ставится 1, если пиксель пустой, то ставится 0.

Для этого в программе создан метод DrawGrid, который будет рисовать данную сетку, чтобы не создавать код на черчение линий, можно сделать функцию, которая названа DrawLine, которая будет принимать начальные и конечные координаты точки, ширину линии и цвет.

```
// создание сетки
drawGrid() {
  for (let x = 0; x < this.canvas.width; x += this.pixelSize) {
    this.drawLine(x,  y1: 0, x, this.canvas.height)
  }

  for (let y = 0; y < this.canvas.height; y += this.pixelSize) {
    this.drawLine( x1: 0, y, this.canvas.width, y)
  }
}

clear() {
  this.ctx.clearRect( x: 0,  y: 0, this.canvas.width, this.canvas.height)
}
```

Рис.13. Создание сетки

```
// создание линии для сетки,чтобы нейросеть запоминала изображение
 drawLine(x1, y1, x2, y2, lineWidth : number  = 2, color : string  = '#00bfff') {
   this.ctx.beginPath()
   this.ctx.moveTo(x1, y1)
   this.ctx.lineWidth = lineWidth
   this.ctx.strokeStyle = color
   this.ctx.lineTo(x2, y2)
   this.ctx.stroke()
 }
```

Рис.14. Создание линии для сетки

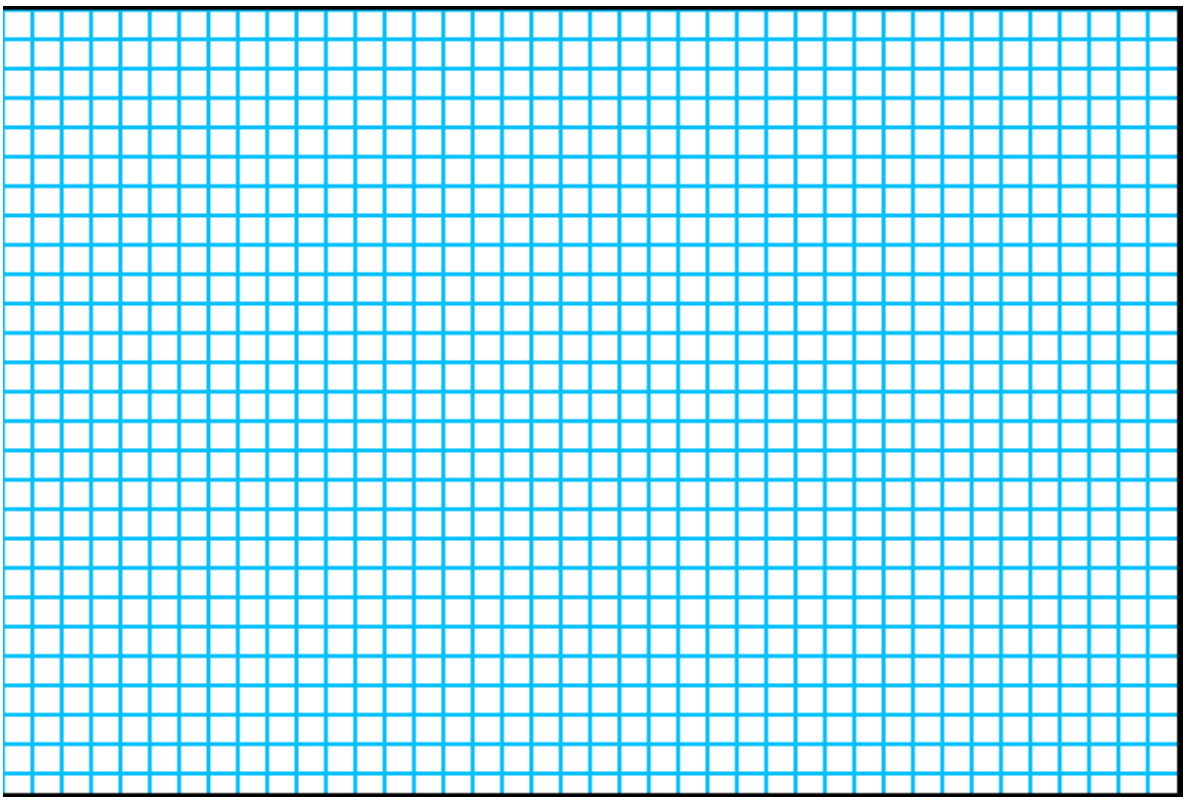

Рис.15. Вид полученной сетки

Так выглядит разбиение палитры, где каждая клетка является пикселем и будет закрашивать зарисованные пиксели для сохранения статистических данных.

Дальше, если клетка закрашена пользователем, то чтобы нейросеть понимала, что она закрашена, вся клетка будет закрашена синим и ставится 1, если клетка пуста, то 0.

Для этого был создан метод DrawCell, который будет закрашивать ячейку. Так же создан метод ToNumericalRepresentation,- это является числовым представлением ячейки, было описано ранее (палитра переводится в 1 и 0).

```
drawCell(x1, y1, size = this.pixelSize, color : string = '#00bfff') {
  this.ctx.beginPath()
  this.ctx.fillStyle = color
  this.ctx.rect(x1, y1, size, size)
  this.ctx.fill()
}
```

Рис.16. Разбиение палитры на ячейки

```
// создание числового представления
// создание числового представления
//если в сетке клетки не закрашены,то ставится 0
//если в сетке клекта закрашена,то ставится 1
  toNumericalRepresentation(shouldBeDrawn) {
    const numericalRepresentation = []
    let cellsToDraw = []

    for (let x = 0; x < this.canvas.width; x += this.pixelSize) {
      for (let y = 0; y < this.canvas.height; y += this.pixelSize) {
        const cell = this.ctx.getImageData(x, y, this.pixelSize, this.pixelSize)
        let isFilled = false

        for(let i = 0; i < cell.data.length; i += 10) {
          if (cell.data[i] !== 0) {
            isFilled = true
            break
          }
        }

        if (isFilled) {
          numericalRepresentation.push(1)
          cellsToDraw.push({ x, y })
        } else {
          numericalRepresentation.push(0)
        }
      }
    }

    if (shouldBeDrawn) {
      cellsToDraw.forEach((cell) => this.drawCell(cell.x, cell.y))
    }

    return numericalRepresentation
  }
```

Рис.17. Числовое представление ячейки.

Эксперименты с рисунками

Сначала мы проводим краткое обучение нейронной сети, чтобы у него появились статистические данные рисунков, с которыми она должна будет сравнивать изображение нарисованное пользователем и затем выдавать ему результат (определить, что нарисовал пользователь).

Пример обучения:

1. Нас приветствует начальный экран, на котором ничего нет.

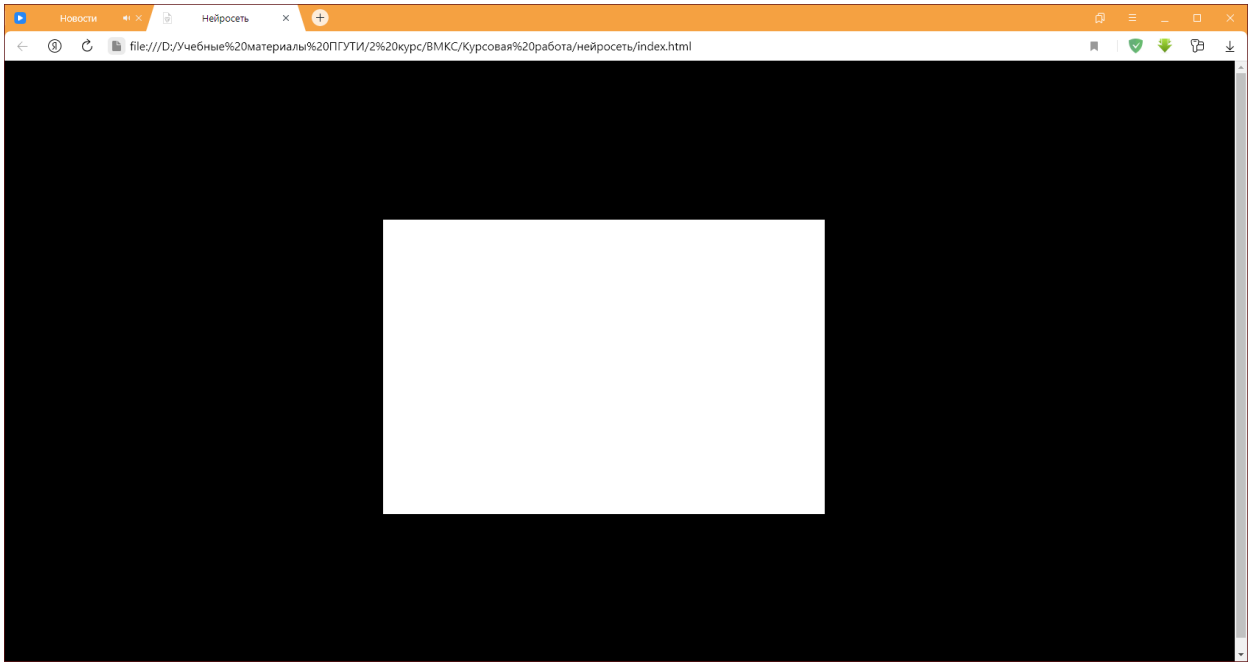

Рис.18. Начальный экран

2. Мы рисуем изображение на нём.

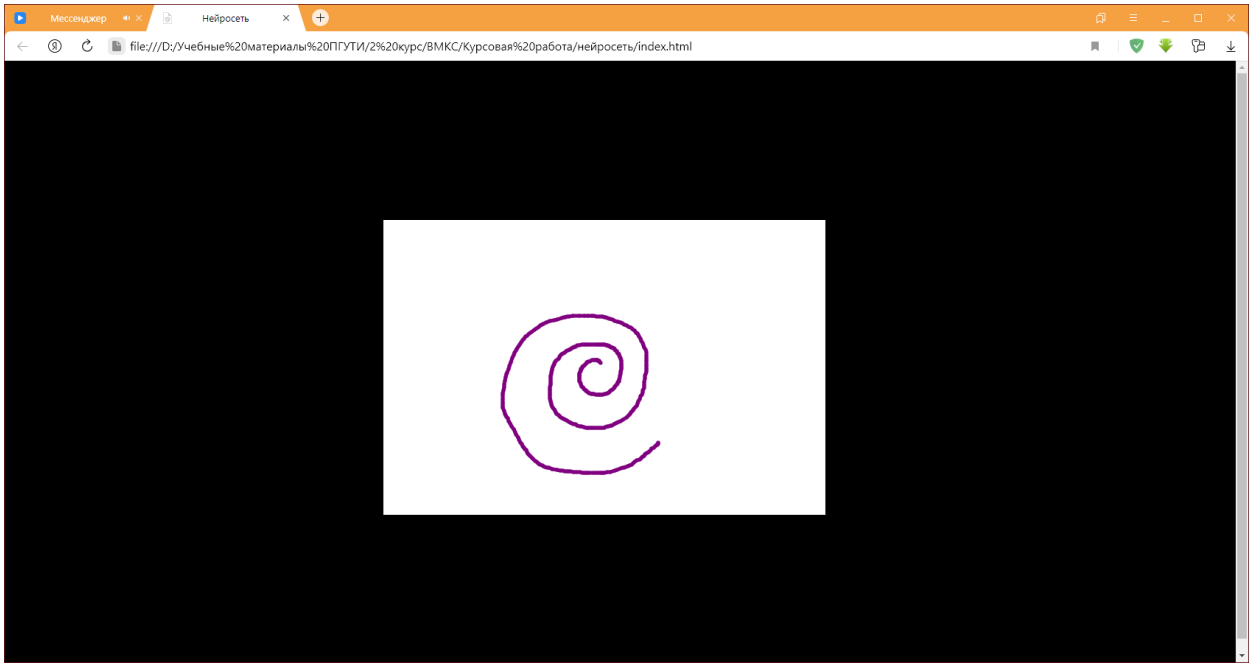

Рис.19. Нарисованное изображение на экране

3. Вносим наш рисунок в базу данных нейросети.

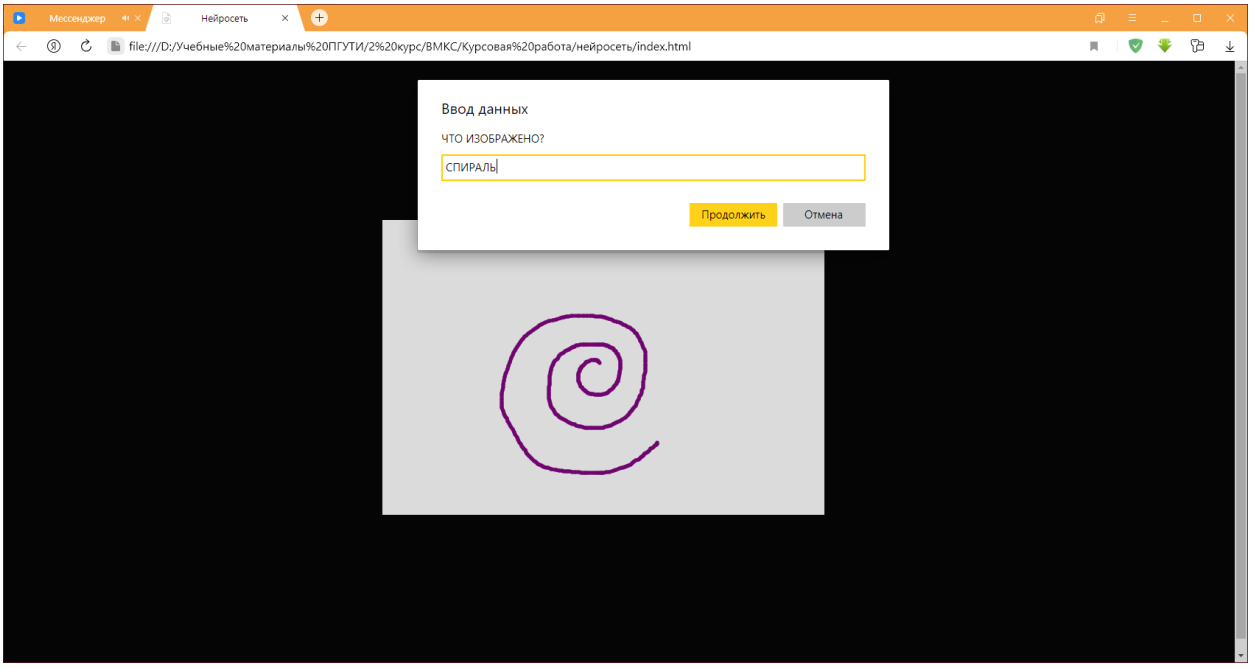

Рис.20. Внесение рисунка в базу данных нейросети

4. Видим, что нейросеть запомнила наш рисунок.

Рис.21. Нейросеть запоминает наш рисунок

5. Рисуем наш рисунок и просим у нейросети дать результат, и как видим она это делает без ошибки.

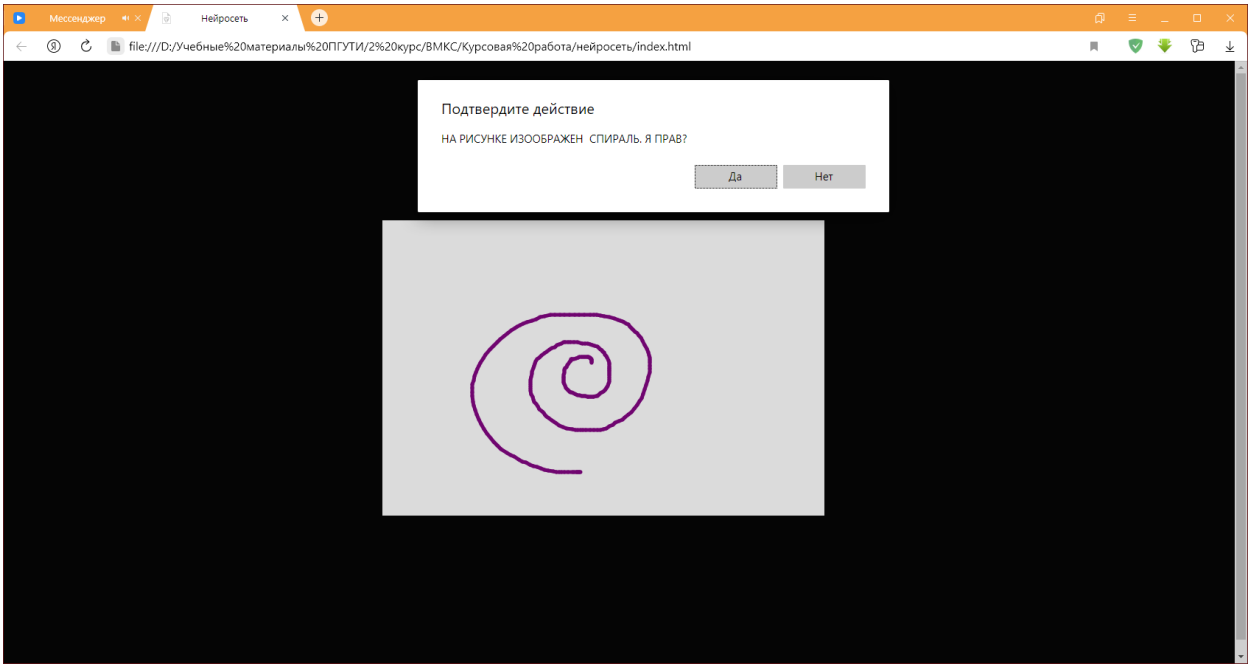

Рис.22. Нейросеть выдаёт результат

Теперь мы внесём в базу нейросети 3 рисунка, которые будут являться сложными, ибо простые рисунки нейросеть угадывает легко, после мы будем составлять статистику, которая покажет, насколько часто нейросеть будет совершать ошибки, при попытке угадывания рисунка нарисованным пользователем.

1. Кот

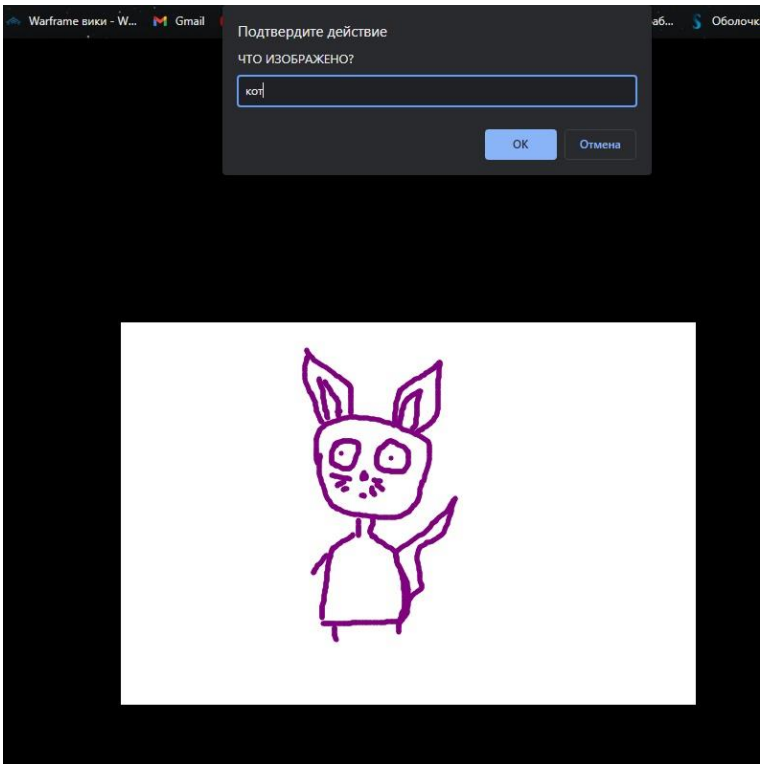

Рис.23. Внесение рисунка в базу нейросети

2. Дом

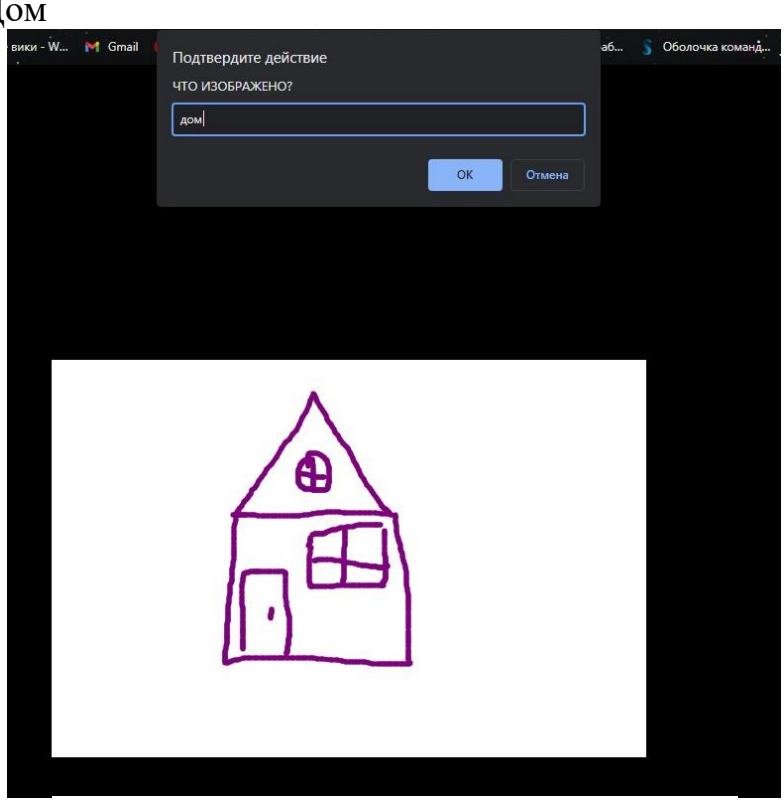

Рис.24. Внесение рисунка в базу нейросети

3. Пальмы

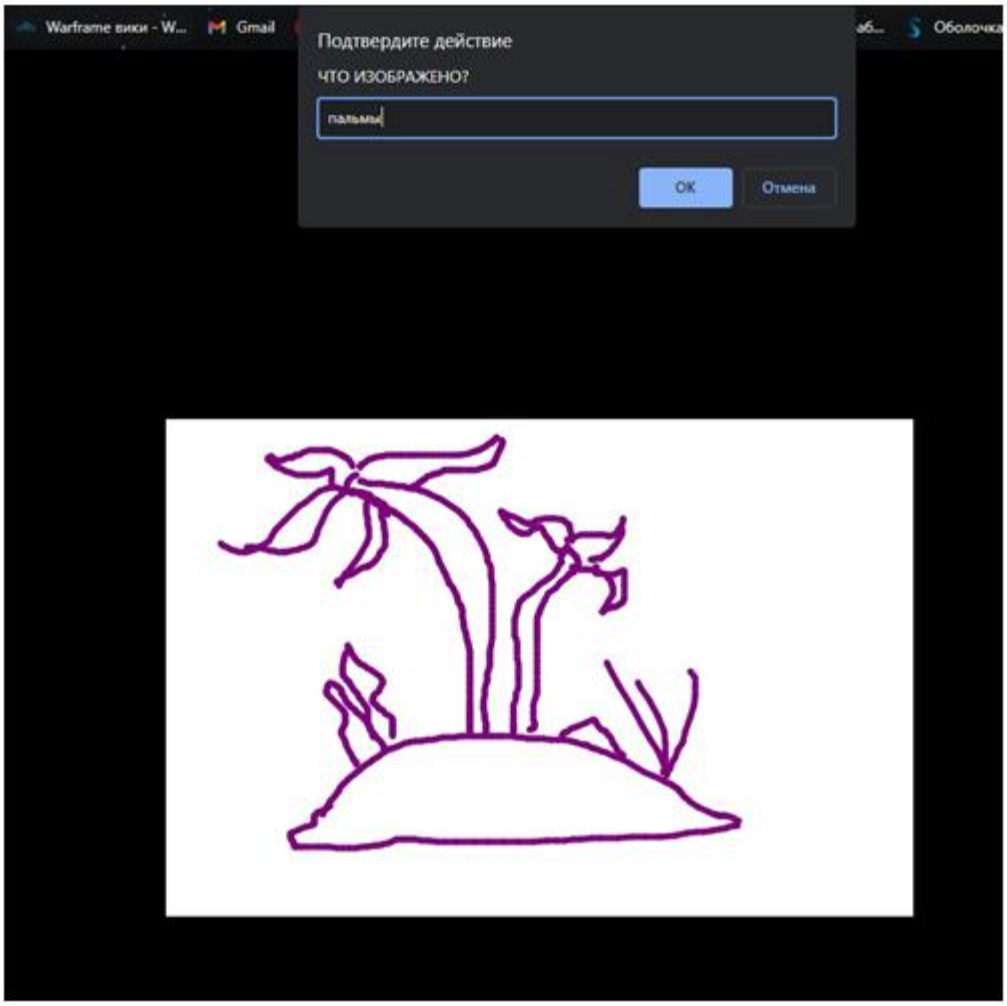

Рис.25. Внесение рисунка в базу нейросети

Мы внесли рисунки в базу нейросети и теперь можно начинать эксперимент. Нарисуем 100 рисунков и посчитаем количество неверных результатов. Отталкиваясь от статистики, составим график неверных результатов.

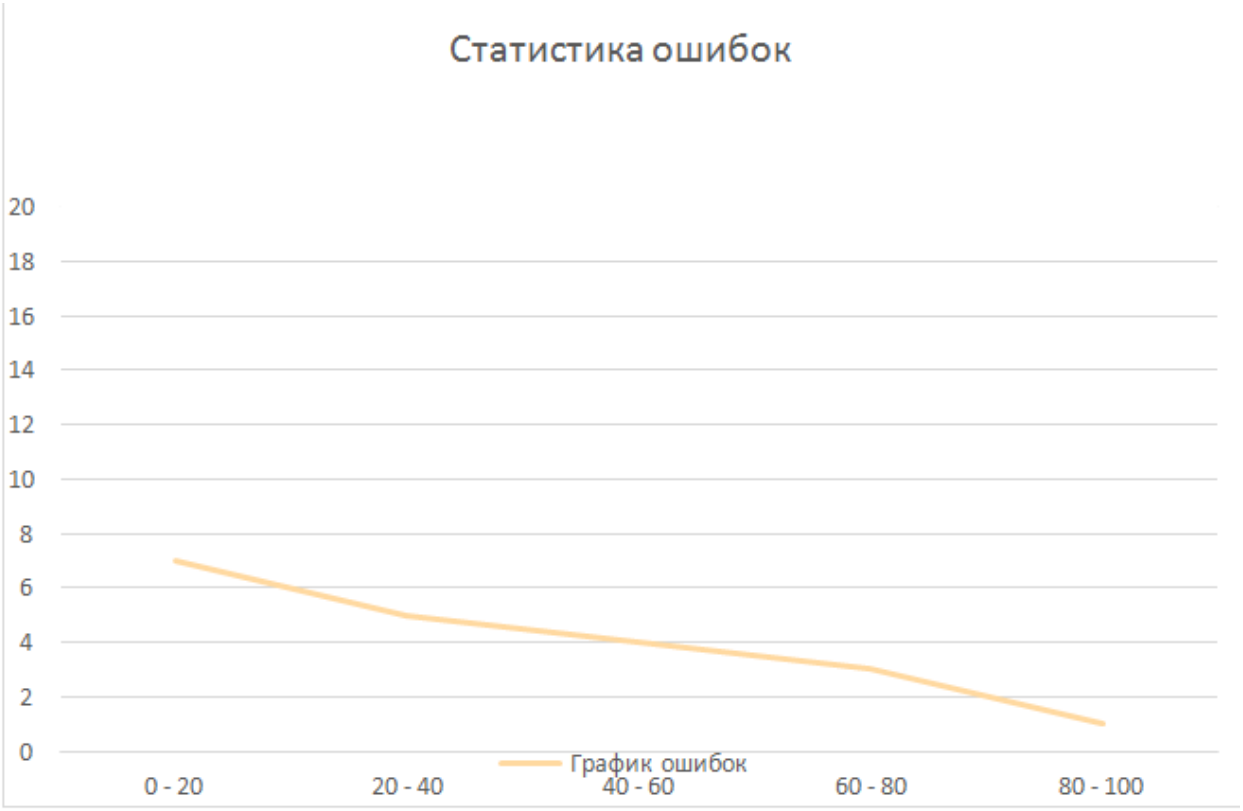

Рис.26. График ошибок нейросети №1.

Как показано на графике, нейросеть обучалась, и с каждым последующим рисунком вероятность получения неверного результата уменьшалась.

Теперь усложним задачу для нашей нейросети, мы нарисуем снова 100 рисунков, но уже не по центру начального экрана, а где-нибудь в углу, рисовать фигуры разного размера, прямые линии делать более кривыми, и посмотрим, как измениться график ошибок нейросети.

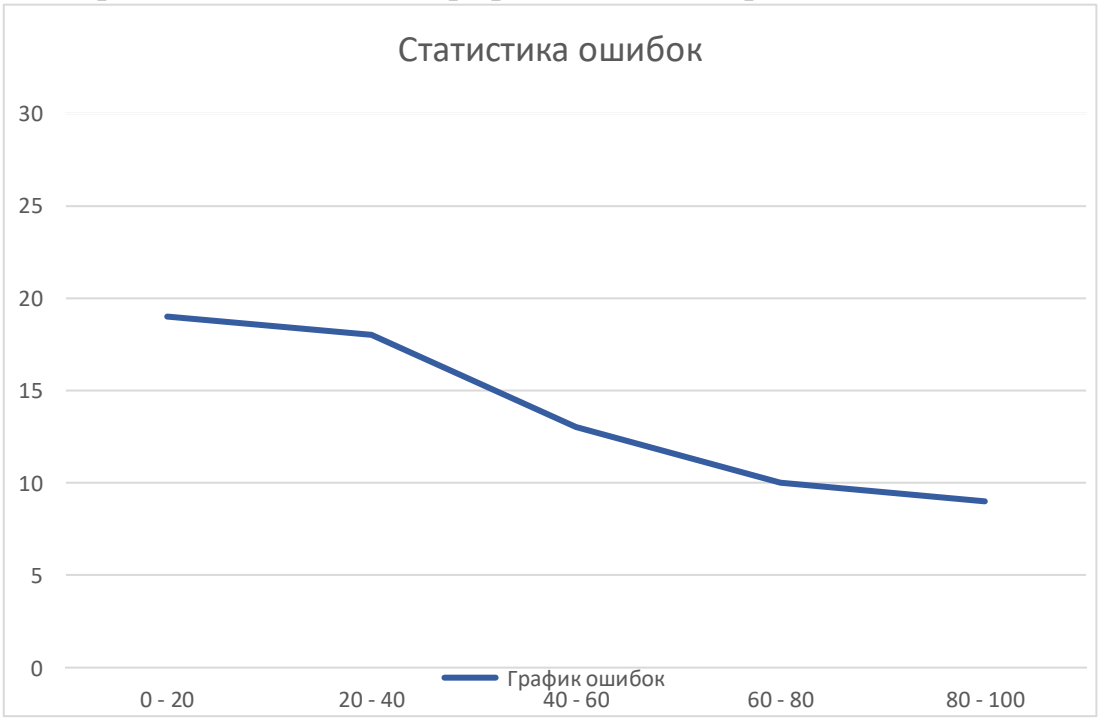

Рис.27. График ошибок нейросети №2

Как мы можем заметить, в первых двадцати рисунках количество ошибок резко увеличилось, как в последующих, но мы можем отследить положительную динамику, ибо видно, что нейросеть так же продолжает обучаться и ошибок с последующими рисунками становиться меньше.

Заключение

Вывод: благодаря, этим теоретическим данным, нейросети Brain.js и возможностям языка JavaScript, мы смогли создать программу, которая может с помощью нейронной сети научиться определять то, что нарисовал пользователь, запоминать это и выводить сообщение пользователю, в котором написано, что он нарисовал на экране.

Список используемой литературы

1. Скачать библиотеку. [Электронный ресурс]. Режим доступа: //github.com/BrainJS/brain.js (дата обращения: 28.11.2022).
2. Информация о нейросетях. [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/304414/ (дата обращения: 28.11.2022).
3. Принципы построения нейронных сетей. [Электронный ресурс]. Режим доступа: https://web-panda.ru/post/brainjs-simple (дата обращения: 28.11.2022).
4 Основные принципы работы [Электронный ресурс]. Режим доступа: https://medium.com/@stasonmars/основы-основ-в-javascr (дата обращения: 28.11.2022).
5. Как пользоваться библиотекой. [Электронный ресурс]. Режим доступа: https://www.youtube.com/watch?v=GUzhH6tTkv4 (дата обращения: 28.11.2022).