УДК 004

Приложение для распознания рукописных цифр на Java с использованием нейронных сетей

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут представлен процесс разработки приложения. Основной целью приложения является возможность распознания нарисованных цифр с помощью нейронных сетей. Нейросеть будет обучена на большом количестве данных и интегрирована в приложения для рисования. Итоговым результатом является рабочее приложения распознавания.
Ключевые слова: Java, Нейросеть, Цифры

Java handwritten digit recognition application using neural networks

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will introduce the application development process. The main purpose of the application is to be able to recognize drawn numbers using neural networks. The neural network will be trained on a large amount of data and integrated into drawing applications. The final result is a working recognition application.
Keywords: Java, Neural network, Numbers

Нейронная сеть, как следует из их названия, основана на биологических нейронах мозга. Хотя мозг — очень сложный орган, и некоторые его функции до сих пор остаются для нас загадкой, клетки, состоящие из нейронов, довольно просты.

Нейрон делится на две основные области:

- Область приема и обработки поступающей информации от других клеток.
- Область проведения и передачи информации другим клеткам.

Тип информации, которую получает, обрабатывает и передает нейрон, зависит от его расположения в нервной системе. Например, нейроны, расположенные в затылочной доле, обрабатывают зрительную информацию, тогда как нейроны в двигательных путях обрабатывают и передают информацию, которая управляет движением мышц. Однако, независимо от типа информации, все нейроны имеют одинаковую базовую анатомическую структуру.

В основном нейроны состоят из входа (дендриты), вычислительной единицы (тела клетки, ядра) и выхода (аксона). Таким образом, сигналы (электрические импульсы) поступают от аксонов других нейронов к некоторым дендритам, затем нейрон обрабатывает сигналы и, наконец, передает сигналы через свой аксон некоторым дендритам других нейронов.

Цель данной работы - разработать нейронную сеть с Java для обучения и обнаружения рукописных цифр.

Разработкой в данной области занимались: С.В. Прокопчина рассмотрела подход создания нового типа нейросети, названого байесовскими измерительными сетями [1]. Так же Л.С. Звягин рассмотрел моделирование в различных типах нейросетей [2]. А.А. Завитаев разработал программу для генерации паролей, где в основе лежит нейросеть [3]. В своей работе И.Ю. Каширин рассмотрел подход смежного наследования для анализа больших данных в нейросетях [4]. С.А. Скрипникова и А.В. Григорьевых разработали информационную систему для найма сотрудников с использованием нейронных сетей [5]. В.Ю. Игнатьев, А.Н. Трекин, П.Ю. Якубовский предложили новый метод для оценки высоты зданий на основе однократного изображения дистанционного зондирования Земли, снятого не в надир [6]. С.А. Орлов, А.С. Бордаков, А.С. Смагин, Т.В. Кожевникова рассмотрели инструментальные среды для разметки объектов на изображениях, обсудили их достоинства и недостатки [7].

Для дальнейшего обучения и тестирования возьмем готовые данные с базы данных MNIST. Она содержит 60 000 обучающих данных и 10 000 тестовых данных. Данные содержат черно-белые рукописные цифровые изображения размером 28X28 пикселей. Каждый пиксель содержит число от 0 до 255.

Способ организации данных не соответствует ни одному из стандартных форматов изображений. Но уже существует решение, читающее данные идеально и быстро. Для каждой записи создадим java-бин «LabeledImage» (рис.1).

```java
public class LabeledImage implements Serializable {
    private final double[] meanNormalizedPixel;
    private final double[] pixels;
    private double label;
    private Vector features;

    public LabeledImage(int label, double[] pixels) {
        meanNormalizedPixel = meanNormalizeFeatures(pixels);
        this.pixels = pixels;
        features = Vectors.dense(meanNormalizedPixel);
        this.label = label;
    }

    public double[] getPixels() { return pixels; }

    private double[] meanNormalizeFeatures(double[] pixels) {
        double min = Double.MAX_VALUE;
        double max = Double.MIN_VALUE;
        double sum = 0;
        for (double pixel : pixels) {
            sum = sum + pixel;
            if (pixel > max) {
                max = pixel;
            }
            if (pixel < min) {
                min = pixel;
            }
        }
        double mean = sum / pixels.length;

        double[] pixelsNorm = new double[pixels.length];
        for (int i = 0; i < pixels.length; i++) {
            pixelsNorm[i] = (pixels[i] - mean) / (max - min);
        }
        return pixelsNorm;
    }

    public Vector getFeatures() { return features; }

    public double getLabel() { return label; }

    public void setLabel(double label) { this.label = label; }
```

Рисунок 1 – Класс LabeledImage

У класса есть метка, которая представляет собой реальную цифру от 0 до 9, и вектор признаков, которые представляют пиксели в одном измерении. Так как входные данные 28x28 пикселей, которые содержат числа от 0 до 255, это будет означать, что есть один массив длиной 784, содержащий числа от 0 до 255. После чтения данных получится список «LabeledImage».

Входные данные для модели — это вектор признаков объекта «LabeledImage». В действии это будет выглядеть так: в представленном одним объектом «LabeledImage», внутри которого есть векторы признаков. Входными данными модели являются векторы признаков, поэтому размер входных данных равен 784. Это случай одного примера, поэтому для масштабирования большего количества примеров выполняем модель для каждого примера. Итак, чтобы полностью обучить модель, нужно будет вычислить функция стоимости, производная для каждого примера.

Выходные данные, с другой стороны, легче обосновать, потому что есть 10 цифр от 0 до 9, поэтому выход представляет собой одномерный вектор размера 10. Значения выходного вектора представляют собой вероятности того, что входные данные, вероятно, будут одной из этих цифр.

Итак, имеется модель с входным размером 784 и выходным размером 10. Теперь пришло время настроить другие слои. Остановимся на двух скрытых слоях: 128 и 64 нейрона. Теоретически, чем больше слой, тем лучше, но и обучение будет намного медленнее. Таким образом, решение о слоях в основном зависит от желаемой точности модели. Здесь есть возможности для улучшения, возможно, в будущем стоит попробовать различные конфигурации слоев и посмотреть, что будет работать лучше всего. Также это можно сделать автоматически с последующим выбором лучшей модели. Код для обучения выглядит следующим образом (рис.2).

```java
public class NeuralNetwork {

    private final static Logger LOGGER = LoggerFactory.getLogger(NeuralNetwork.class);

    private SparkSession sparkSession;
    private MultilayerPerceptronClassificationModel model;

    public void init() {
        initSparkSession();
        if (model == null) {
            LOGGER.info("Загрузка нейронной сети из сохраненной модели... ");
            model = MultilayerPerceptronClassificationModel.load("resources/nnTrainedModels/ModelWith60000");
            LOGGER.info("Загрузка из сохраненной модели выполнена");
        }
    }

    public void train(Integer trainData, Integer testFieldValue) {

        initSparkSession();

        List<LabeledImage> labeledImages = IdxReader.loadData(trainData);
        List<LabeledImage> testLabeledImages = IdxReader.loadTestData(testFieldValue);

        Dataset<Row> train = sparkSession.createDataFrame(labeledImages, LabeledImage.class).checkpoint();
        Dataset<Row> test = sparkSession.createDataFrame(testLabeledImages, LabeledImage.class).checkpoint();

        int[] layers = new int[]{784, 128, 64, 10};

        MultilayerPerceptronClassifier trainer = new MultilayerPerceptronClassifier()
                .setLayers(layers)
                .setBlockSize(128)
                .setSeed(1234L)
                .setMaxIter(100);

        model = trainer.fit(train);

        evalOnTest(test);
        evalOnTest(train);
    }
```

Рисунок 2 – Код нейросети

Для более однородных значений от 0 до 1 произведем нормализацию данных по формуле (рис.3).

$$X_i = \frac{X_i - \mu_i}{s_i}$$

Рисунок 3 – Формула нормализации

Где μi — среднее значение всех значений для признака (i), а si — диапазон значений (максимум — минимум) или стандартное отклонение.

Итак, в данном случае каждое значение пикселя вычитаем из среднего значения всех пикселей для этого изображения и делим на разницу между максимальным значением пикселя и минимальным значением пикселя на этом изображении. Код выглядит следующим образом (рис.4).

```java
private double[] meanNormalizeFeatures(double[] pixels) {
    double min = Double.MAX_VALUE;
    double max = Double.MIN_VALUE;
    double sum = 0;
    for (double pixel : pixels) {
        sum = sum + pixel;
        if (pixel > max) {
            max = pixel;
        }
        if (pixel < min) {
            min = pixel;
        }
    }
    double mean = sum / pixels.length;

    double[] pixelsNorm = new double[pixels.length];
    for (int i = 0; i < pixels.length; i++) {
        pixelsNorm[i] = (pixels[i] - mean) / (max - min);
    }
    return pixelsNorm;
}
```

Рисунок 4 – Нормализация данных

Эта реализация не является полной в соответствии с формулой, которую указали в рисунке 3, потому что она нормализуется для каждого изображения, каждое среднее, максимальное и минимальное значения рассчитываются для одного пикселя изображения. А формула, которую указали в рисунке 3, требует вычисления среднего, максимального, минимального для всех пикселей изображений, а затем вычитания этого среднего и деления на этот максимальный-минимальный пиксель изображения. Поскольку данная работа представляет собой простое черно-белое изображение, эта нормализация работает правильно.

После применения нормализации точность возросла до 97%. Таким образом, только 3% были ошибочно обнаружены сетью. Это хороший результат, учитывая простоту модели и трудозатраты.

Теперь интегрируем данную нейросеть в приложение для рисования, и запустим для тестирования.

При запуске в консоли будет проходить процесс обучения сети (рис.5).

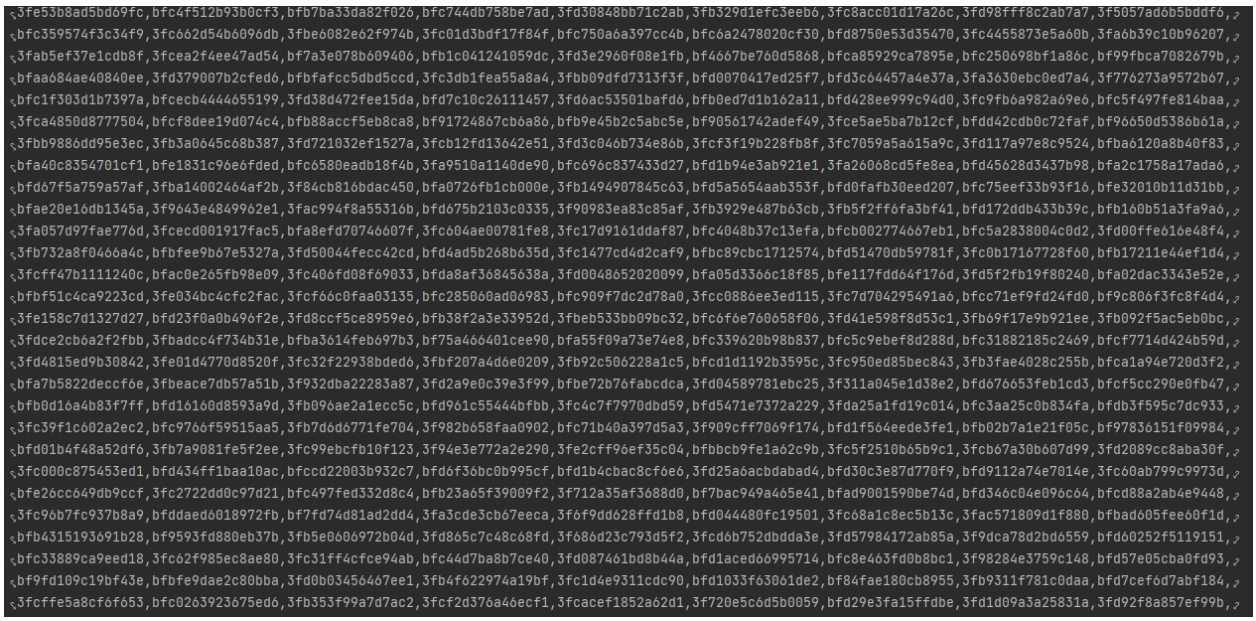

Рисунок 5 – Процесс обработки изображений в консоли

После запуска и обработки всех тестовых изображений представляется окно приложения (рис.6)

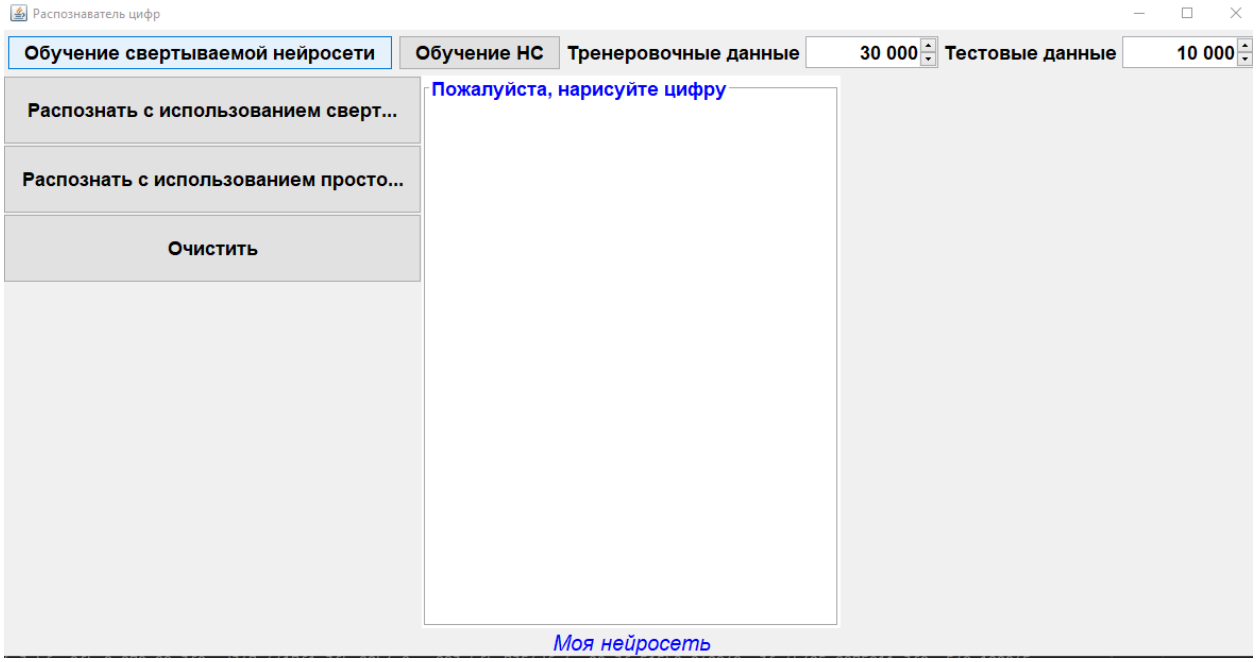

Рисунок 6 – Интерфейс приложения

Приложение уже загружает обучение по умолчанию, выполненное заранее с точностью 97%, протестированное на 10 000 тестовых данных и обученное с 60 000 изображений в два слоя 128 нейронов и 64 нейрона.

Теперь нарисуем в центральном окне цифру и проверим работоспособность (рис.7-8).

Рисунок 7 – Процесс работы приложения

Рисунок 8 - Процесс работы приложения

В данной статье был описан процесс создания приложения для распознания рукописных цифр с использованием нейронной сети в Java.

Библиографический список

1. Прокопчина С.В. Новый тип нейросетей: байесовские измерительные нейросети (бин) на базе методологии регуляризирующего байесовского подхода // Мягкие измерения и вычисления. 2020. №10. С. 17-24.
2. Звягин Л.С. Особенности моделирования в различных типах нейросетей // Мягкие измерения и вычисления. 2022. №3. С. 43-54.

3. Завитаев А.А. Программа генерации паролей с применением нейросетей // Мягкие измерения и вычисления. 2021. №7. С. 31-45.
4. Каширин И.Ю. Нейросети со знаниями для анализа больших данных // Вестник рязанского государственного радиотехнического университета 2021. №75. С. 71-84.
5. Скрипникова С.А., Григорьевых А.В. Найм сотрудников с использованием нейросети кадрового подбора // Информационные технологии в управлении и экономике. 2022. №1(26). С. 32-48.
6. Игнатьев В.Ю., Трекин А.Н., Якубовский П.Ю. Глубокие нейросети для вычисления параметров зданий по одномоментному космическому изображению // Известия российской академии наук. теория и системы управления. 2020. №5. С. 116-128.
7. Орлов С.А., Бордаков А.С., Смагин А.С., Кожевникова Т.В. Инструментальные среды для формирования обучающей выборки нейросети // Научно-техническое и экономическое сотрудничество стран АТР в XXI веке. 2021. №2 С. 282-285.