Lab6Part1实验报告

吴优 22307130158

实验目标

在 Java 中实现一个张量（tensor）类，并实现张量所支持的各种运算。

张量类的实现

本模块中，要求实现一个张量类，并支持如下运算：

- 张量的初始化：能够创建不同维度和数据类型的张量。
- 基本运算：支持加法、减法基本算术运算。不要求实现点积、张量乘法与除法。
- 索引：能够对张量进行索引操作，以获取特定的子张量。
- 填充：对张量中最后两个维度（图像高度和图像宽度）对应的矩阵周围填充 0。
- 拉伸：将一个张量的后两个维度拉伸到指定大小。

代码解析

采用助教所提供的大致框架，在框架下实现细节。

构造函数的实现

- 构造函数：`public Tensor(int... dimensions)`: 该构造函数用于创建指定维度的张量对象，通过递归调用 `createData` 方法来初始化数据。
- 数据创建方法：`private Object createData(int[] dimensions, int index)`: 递归方法，用于创建多维数组数据。

```
    public Tensor(int... dimensions) {
        this.dimensions = dimensions;
        this.data = createData(dimensions, 0);
    }
        private Object createData(int[] dimensions, int index) {
        if (index == dimensions.length - 1) {
            // 创建最内层的数组
            return new Object[dimensions[index]];
        } else {
            // 递归创建多维数组
            Object[] dataArray = new Object[dimensions[index]];
            for (int i = 0; i < dimensions[index]; i++) {
                dataArray[i] = createData(dimensions, index + 1);
            }
            return dataArray;
        }
    }
```

解析：实现了最基本的构造函数，输入指定维度的张量对象，递归调用`createData`来初始化数据，其中createData实现为对每一维度递归，创建指定维度的多维数组。

- `public Tensor(int[][][] data)`: 该构造函数接受一个三维数组作为参数，直接初始化数据，并获取维度信息。 解析：为了后续卷积实现处理的便利，新增一个可以指定张量存储内容的构造函数

获取数据方法

- `public Object getData()`: 用于获取张量的数据对象。

获取元素方法

- `public <T> T get(int... indices)`: 用于获取指定索引处的元素。

```
public <T> T get(int... indices) {
    Object current = data;
    for (int index : indices) {
        if (current instanceof Object[]) {
            current = ((Object[]) current)[index];
        } else if (current instanceof int[]) {
            current = ((int[]) current)[index];
        } else {
            throw new IllegalStateException("Unsupported data type");
        }
    }
    // 强制类型转换并返回
    return (T) current;
}
```

解析：使用泛型，便于对不同的数据类型处理，在后续实践中发现对于Object[]类型和int[]类型处理出现冲突，需要返回灵活的数据类型。

设置元素方法

- `public void set(Object value, int... indices)`: 用于设置指定索引处的元素值。

```
public void set(Object value, int... indices) {
    Object current = data;
    for (int i = 0; i < indices.length - 1; i++) {
        current = ((Object[]) current)[indices[i]];
    }
    ((Object[]) current)[indices[indices.length - 1]] = value;
}
```

解析：循环到指定维度中，设置current值为value

加减法操作方法

- `public Tensor add(Tensor other)`: 用于实现两个张量的加法操作。

```
    public Tensor add(Tensor other) {
        int[] dims = getDimensions();
        Tensor result = new Tensor(dims);
        addRecursive(this.data, other.data, result.data, 0);
        return result;
    }
    private void addRecursive(Object data1, Object data2, Object result, int index) {
        if (index == dimensions.length - 1) {
            int length = ((Object[]) data1).length;
            for (int i = 0; i < length; i++) {
                ((Object[]) result)[i] = (int)((Object[]) data1)[i] + (int)((Object[]) data2)
[i];
             }
        } else {
            int length = ((Object[]) data1).length;
            for (int i = 0; i < length; i++) {
                addRecursive(((Object[]) data1)[i], ((Object[]) data2)[i], ((Object[])
result)[i], index + 1);
            }
        }
    }
```

解析：递归调用addRecursive函数，对每个维度的对应位置相加。

填充方法

- `public Tensor pad(int padHeight, int padWidth)`: 用于在张量周围填充指定数量的零值。

```
public Tensor pad(int padHeight, int padWidth)
{
    int[] originalDimensions = getDimensions();
    int originalHeight = originalDimensions[originalDimensions.length - 2];
    int originalWidth = originalDimensions[originalDimensions.length - 1];
    // 计算填充后的尺寸
    int paddedHeight = originalHeight + 2 * padHeight;
    int paddedWidth = originalWidth + 2 * padWidth;
    // 创建填充后的张量
    int[] paddedDimensions = Arrays.copyOf(originalDimensions, originalDimensions.length);
    paddedDimensions[originalDimensions.length - 2] = paddedHeight;
    paddedDimensions[originalDimensions.length - 1] = paddedWidth;
    Tensor paddedTensor = new Tensor(paddedDimensions);
    for(int m=0;m<paddedTensor.dimensions[0];m++)
    {
        for(int n=0;n<paddedTensor.dimensions[1];n++)
        {
            // 填充操作
            for (int i = 0; i < originalHeight; i++) {
                for (int j = 0; j < originalWidth; j++) {
                    // 复制原始张量的数据到填充后的张量中心区域
                    paddedTensor.set(get(m,n,i, j),m,n, i + padHeight, j + padWidth);
                }
            }

            // 左侧和右侧填充
            for (int i = 0; i < paddedHeight; i++)
            {
                for (int j = 0; j < padWidth; j++) {
                    paddedTensor.set(0,m,n, i, j); // 左侧填充
                    paddedTensor.set(0, m,n,i, paddedWidth - 1 - j); // 右侧填充
                }
            }

            // 上侧和下侧填充
            for (int i = 0; i < padHeight; i++) {
                for (int j = 0; j < paddedWidth; j++) {
                    paddedTensor.set(0,m,n, i, j); // 上侧填充
                    paddedTensor.set(0, m,n,paddedHeight - 1 - i, j); // 下侧填充
                }
            }
        }
    }
    return paddedTensor;
}
```

解析：首先复制dimension数组并进行最后两维的计算修改dimension数组。然后复制每一维度的数据到新建张量的中心，并对该新建张量上下左右填充0

拉伸方法

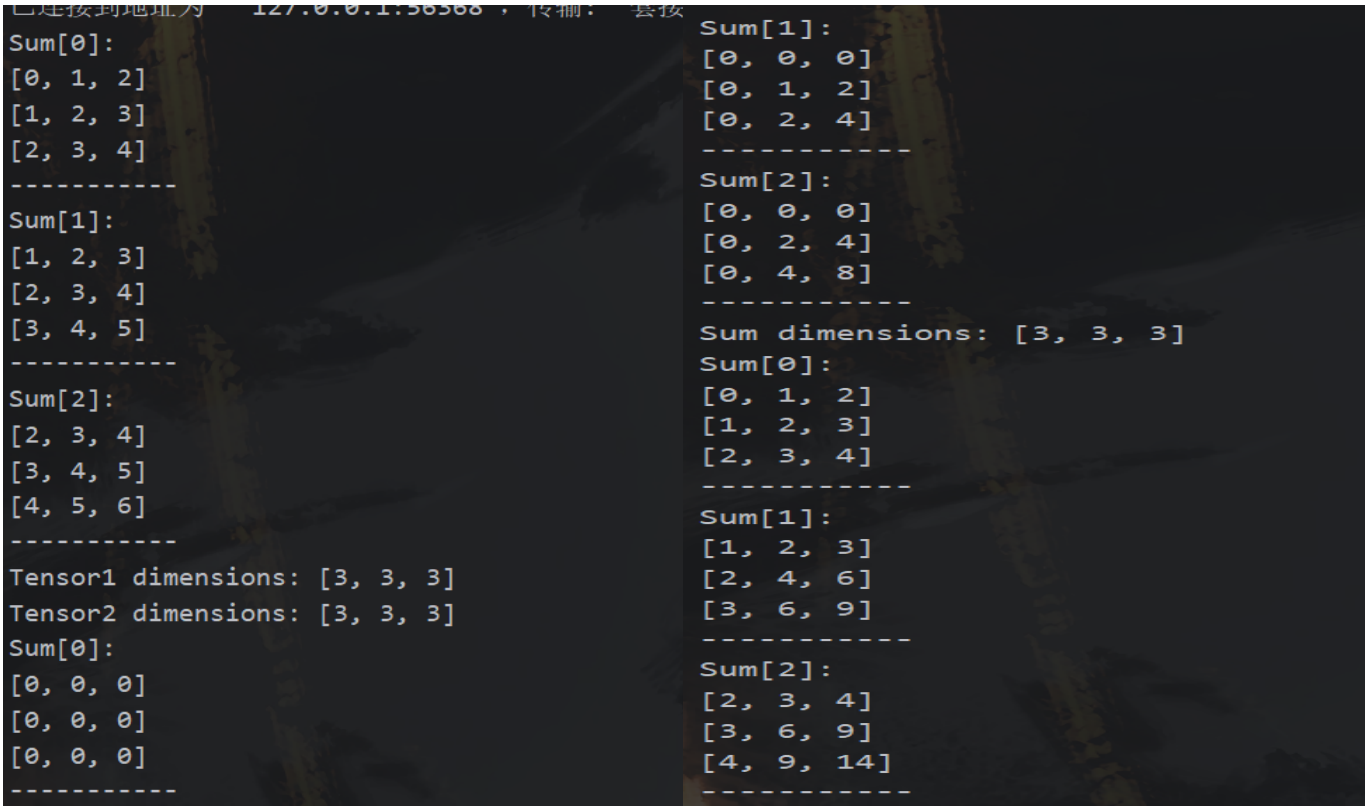
- `public Tensor stretch(int newHeight, int newWidth)`：用于将张量在高度和宽度方向上拉伸至指定尺寸。

```
    public Tensor stretch(int newHeight, int newWidth) {
        int[] originalDimensions = getDimensions();
        int originalHeight = originalDimensions[originalDimensions.length - 2];
        int originalWidth = originalDimensions[originalDimensions.length - 1];
        // 创建拉伸后的张量
        int[] stretchedDimensions = Arrays.copyOf(originalDimensions,
originalDimensions.length);
        stretchedDimensions[originalDimensions.length - 2] = newHeight;
        stretchedDimensions[originalDimensions.length - 1] = newWidth;
        Tensor stretchedTensor = new Tensor(stretchedDimensions);
        // 计算拉伸比例
        double heightRatio = (double)newHeight / originalHeight;
        double widthRatio = (double)newWidth / originalWidth;

        // 根据拉伸比例进行像素值映射
        Random random = new Random(); // 用于生成随机数
        for (int i = 0; i < newHeight; i++) {
            for (int j = 0; j < newWidth; j++) {
                // 计算在原始张量中对应的位置
                int originalRow = (int)(i / heightRatio);
                int originalCol = (int)(j / widthRatio);
                // 获取对应位置的像素值
                Object pixelValue;
                if (originalRow < originalHeight && originalCol < originalWidth) {
                    pixelValue = get(originalRow, originalCol);
                } else {
                    pixelValue = random.nextInt(); // 随机取值
                }
                // 将像素值设置到拉伸后的张量中
                stretchedTensor.set(pixelValue, i, j);
            }
        }
        return stretchedTensor;
    }
```

解析：计算拉伸比例后，对新张量中的每个位置计算原有张量对应的位置，如果不是整数则直接取整，获取像素值后放入新张量中，如果新张量的部分超出了原张量，采取随机取值的方法放到新张量中

运行截图


已连接到地址为 "127.0.0.1:56368"，传输: "套接
Sum[0]:
[0, 1, 2]
[1, 2, 3]
[2, 3, 4]
Sum[1]:
[1, 2, 3]
[2, 3, 4]
[3, 4, 5]
Sum[2]:
[2, 3, 4]
[3, 4, 5]
[4, 5, 6]
Tensor1 dimensions: [3, 3, 3]
Tensor2 dimensions: [3, 3, 3]
Sum[0]:
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
Sum[1]:
[0, 0, 0]
[0, 1, 2]
[0, 2, 4]
Sum[2]:
[0, 0, 0]
[0, 2, 4]
[0, 4, 8]
Sum dimensions: [3, 3, 3]
Sum[0]:
[0, 1, 2]
[1, 2, 3]
[2, 3, 4]
Sum[1]:
[1, 2, 3]
[2, 4, 6]
[3, 6, 9]
Sum[2]:
[2, 3, 4]
[3, 6, 9]
[4, 9, 14]