山东大学计算机科学与技术学院

可视化技术课程实验报告

学号：202300130205	姓名：李尚远	班级：23 级数据班
实验题目：cast 数据驱动图表动画		
实验学时：2	实验日期：2025. 11. 28	
实验目标：尝试使用 libra 交互式可视化工具，体会当前交互式可视化的工作方向与重点，为日后的项目提供可视化的参考		

实验步骤：

定义依赖和入口函数

```
const VIS = require("./staticVisualization"); // 导入自定义静态可视化模块

async function main() {
  await VIS.loadData(); // ① 异步加载 MNIST 数据（需等待数据就绪）
  VIS.renderStaticVisualization(); // ② 渲染静态基础视图（如坐标轴、图例等）
  const mainLayer = renderMainVisualization(); // ③ 创建交互图层并绘制散点
  mountInteraction(mainLayer); // ④ 为该图层挂载 hover 交互
}
```

散点图渲染

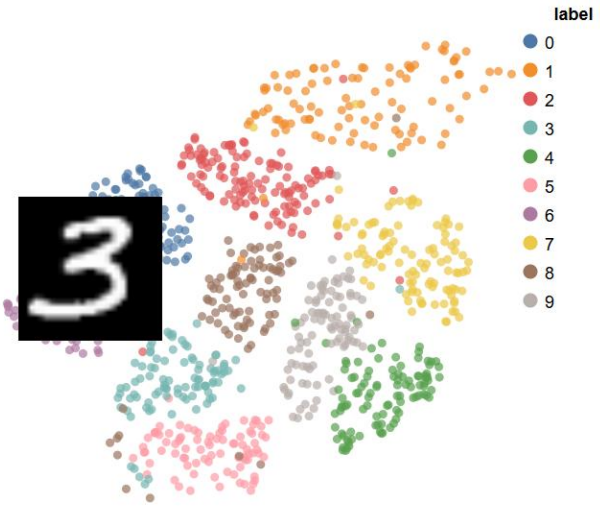
```
function renderMainVisualization() {
  // ① 选择页面中已存在的 SVG 容器（#LibraPlayground 是 Libra 的默认容器 ID）
  const svg = d3.select("#LibraPlayground svg");

  // ② 创建 Libra 交互图层（D3Layer 类型，适配 D3 的绘图逻辑）
  const mainLayer = Libra.Layer.initialize("D3Layer", {
    name: "mainLayer", // 图层名称（用于后续交互绑定）
    width: globalThis.WIDTH, // 图层宽度（继承全局配置）
    height: globalThis.HEIGHT, // 图层高度（继承全局配置）
    offset: { x: globalThis.MARGIN.left, y: globalThis.MARGIN.top }, // 偏移量（避开边距，与坐标轴对齐）
    container: svg.node(), // 图层挂载到 SVG 容器
  });

  // ③ 获取图层的绘图容器（D3 的<g>元素）
  const g = d3.select(mainLayer.getGraphic());

  // ④ D3 核心绘图逻辑：绑定数据并绘制散点
  g.selectAll("circle") // 选择所有 circle 元素（初始为空）
    .data(globalThis.data) // 绑定 MNIST 数据集（globalThis.data 由 VIS.loadData()加载）
    .join("circle") // 数据绑定：新增数据创建 circle，删除数据移除 circle
```

```
      .attr("class", "mark") // 给散点添加 class（用于样式控制或选择器定位）
      .attr("cx", (d) => globalThis.x(d[globalThis.FIELD_X])) // 散点 x 坐标（通过比例尺映射数据值）
      .attr("cy", (d) => globalThis.y(d[globalThis.FIELD_Y])) // 散点 y 坐标（同上）
      .attr("fill", (d) => globalThis.color(d[globalThis.FIELD_COLOR])) // 散点颜色（按类别映射）
      .attr("fill-opacity", 0.7) // 透明度（避免散点重叠遮挡）
      .attr("r", 3); // 散点半径（3px，平衡可见性与密度）

    return mainLayer; // 返回图层实例，供后续挂载交互
}
```

交互挂载

```
async function mountInteraction(layer) {
    // ① 构建 Hover 交互（继承 Libra 内置的 HoverInstrument）
    Libra.Interaction.build({
      inherit: "HoverInstrument", // 继承内置悬浮交互工具
      layers: [layer], // 绑定到之前创建的 mainLayer（只对该图层的元素生效）
      sharedVar: { // 交互配置参数（共享变量，控制 tooltip 行为）
        tooltip: {
          image: (d) => d.image, // tooltip 显示的图像：取数据中的 image 字段（MNIST 原始图像数据）
          offset: { // tooltip 相对于鼠标的偏移量（避免遮挡鼠标或散点）
            x: -70 - globalThis.MARGIN.left, // 向左偏移（70px + 左边距，确保不压散点）
            y: -100 - globalThis.MARGIN.top, // 向上偏移（100px + 上边距）
          },
        },
      },
    });

    // ② 创建交互历史跟踪（Libra 内置功能，记录用户交互行为，可选）
    await Libra.createHistoryTrrack();
}
```

效果展示



鼠标悬停在数据点上可以展示其对应的手写体图片

结论：

体会到了当今流行系统交互式可视化的功能。