医 学 影 像 分 析（实验四）

题目：实验四：实现一个基于深度学习的医学图像检测算法

学生姓名：王天也

学　　号：21S010051

所在学院：经济与管理学院

专业名称：管理科学与工程

2022 年 5 月

目 录

1 实验四，实现一个基于深度学习的医学图像检测算法

1.1 实验目的

熟悉 Faster-RCNN、Fast-RCNN 等基于深度学习的医学图像检测网络

1.2 用于对象检测的不同 R-CNN 算法的简要概述

R-CNN 使用选择性搜索从给定图像中提取一堆区域，然后检查这些框是否包含对象。我们首先提取这些区域，对于每个区域，CNN 用于提取特定的特征。最后，这些特征随后被用于检测对象。不幸的是，由于过程中涉及的多个步骤，R-CNN 变得相当慢。

另一方面，Fast R-CNN 将整个图像传递给生成感兴趣区域的 ConvNet（而不是传递从图像中提取的区域）。此外，它没有使用三个不同的模型（如我们在 R-CNN 中看到的），而是使用一个模型从区域中提取特征，将它们分类为不同的类，并返回边界框。

所有这些步骤都是同时完成的，因此与 R-CNN 相比，它的执行速度更快。然而，Fast R-CNN 在应用于大型数据集时不够快，因为它还使用选择性搜索来提取区域。

Faster R-CNN 通过将选择性搜索替换为区域建议网络 (RPN) 来解决选择性搜索的问题。我们首先使用 ConvNet 从输入图像中提取特征图，然后将这些图通过 RPN 传递，该 RPN 返回对象建议。最后，对这些地图进行分类并预测边界框。

我总结了以下步骤，然后是 Faster R-CNN 算法来检测图像中的对象：

获取输入图像并将其传递给 ConvNet，后者返回图像的特征图在这些特征图上应用区域建议网络（RPN）并获得对象建议应用 ROI 池化层以将所有提案降低到相同的大小最后，将这些建议传递给全连接层，以便对任何预测图像的边界框进行分类

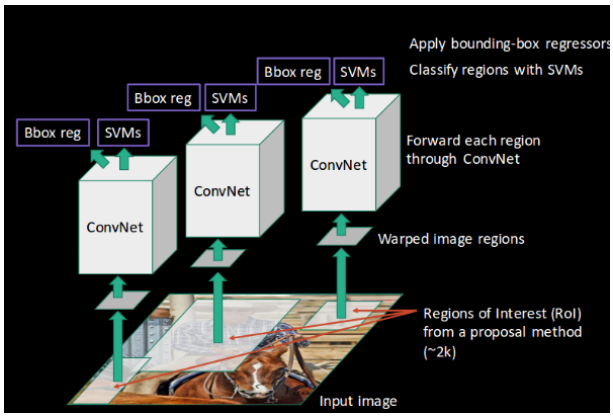

图 1.1　卷积神经网络

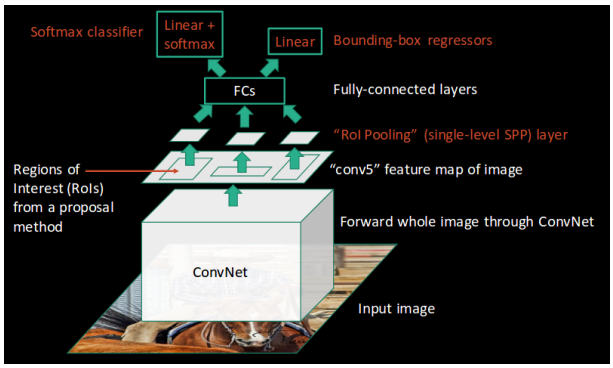

图 1.2　Fast R-CNN

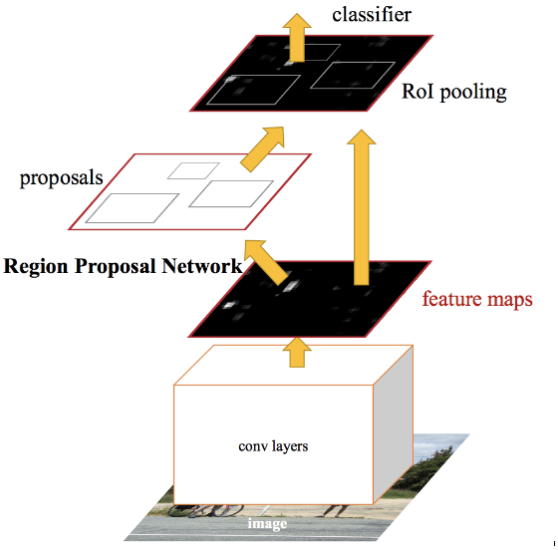

图 1.3　Faster R-CNN

算法	特征	预测时间/图像	局限性
CNN	将图像划分为多个区域，然后将每个区域分类为各种类别。	-	需要很多区域才能准确预测，因此计算时间长。
R-CNN	使用选择性搜索来生成区域。从每张图像中提取大约 2000 个区域。	40-50 秒	由于每个区域分别传递给 CNN，因此计算时间长。此外，它使用三种不同的模型进行预测。
Fast-RCNN	每个图像只传递给 CNN 一次，然后提取特征图。在这些地图上使用选择性搜索[2]来生成预测。将 R-CNN 中使用的所有三个模型组合	2 秒	选择性搜索很慢，因此计算时间仍然很高。

1.3 实验步骤

我们将研究与医疗保健相关的数据集，目的是解决血细胞检测问题。我们的任务是检测通过显微图像读数拍摄的每张图像中的所有红细胞 (RBC)、白细胞 (WBC) 和血小板。以下是我们最终预测的示例：

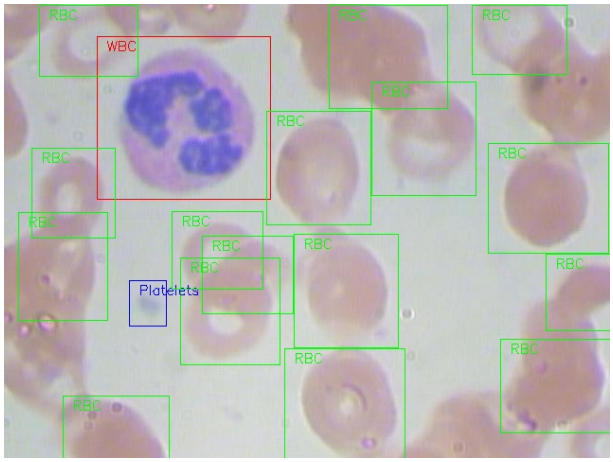

图 1.4　示例

选择这个数据集的原因是我们血液中红细胞、白细胞和血小板的密度提供了大量关于免疫系统和血红蛋白的信息。这可以帮助我们潜在地确定一个人是否健康，如果在他们的血液中发现任何差异，可以迅速采取行动进行诊断。

通过显微镜手动观察样品是一个繁琐的过程。这就是深度学习模型发挥如此重要作用的地方。他们可以从显微图像中以令人印象深刻的精度对血细胞进行分类和检测。

在本文的范围内，我对数据进行了一些修改：

边界框已从给定的.xml 格式转换为.csv 格式我还通过随机挑选图像进行分割，在整个数据集上创建了训练集和测试集分割请注意，我们将使用流行的 Keras 框架和 Python 中的 TensorFlow 后端来训练和构建我们的模型。

数据探索

首先探索我们拥有的数据总是一个好主意（坦率地说，这是一个强制性步骤）。这不仅可以帮助我们发掘隐藏的模式，还可以帮助我们全面了解我们正在使用的内容。我从整个数据集中创建的三个文件是：

1.train_images：我们将用于训练模型的图像。我们在这个文件夹中有每个类的类和实际边界框。

2.test_images：此文件夹中的图像将用于使用经过训练的模型进行预测。该集合缺少这些类的类和边界框。

3.train.csv：包含每个图像的名称、类和边界框坐标。一张图片可以有多行，因为一张图片可以有多个对象。

	image_names	cell_type	xmin	xmax	ymin	ymax
0	1.jpg	RBC	68	165	154	249
1	1.jpg	RBC	1	66	145	260
2	1.jpg	RBC	207	334	160	270
3	1.jpg	RBC	435	540	347	437
4	1.jpg	RBC	535	639	356	464

图 1.5　数据集

train 文件中有 6 列。

1.image_names：包含图像的名称

2. cell_type：表示单元格的类型

3. xmin：图像左下部分的 x 坐标

4. xmax：图像右上角的 x 坐标

5.ymin：图像左下部分的 y 坐标

6.ymax：图像右上角的 y 坐标

可视化我们的图像内容：

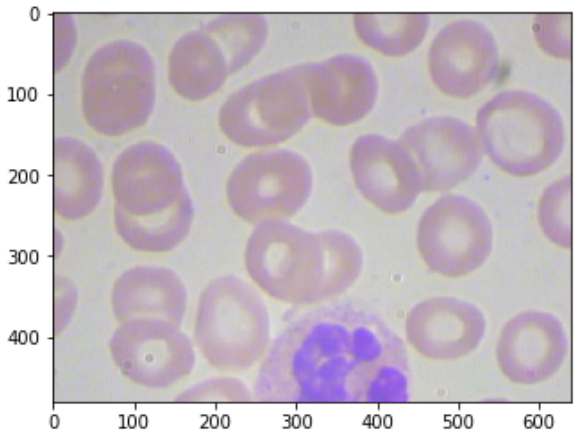

图 1.6　数据集

这就是血细胞图像的样子。这里，蓝色部分代表 WBC，略带红色的部分代表 RBC。我们看看我们的训练集中有多少张图像以及不同类型的类。

我们有三种不同类别的细胞，即红细胞、白细胞和血小板。

我们看看检测到物体的图像会是什么样子

为了实现 Faster R-CNN 算法，我们将遵循这个 Github 存储库中提到的步骤。因此，作为第一步，请确保克隆此存储库。打开一个新的终端窗口并键入以下内容来执行此操作：

```
RBC          2909
WBC           262
Platelets     252
Name: cell type, dtype: int64
```

图 1.7　训练集

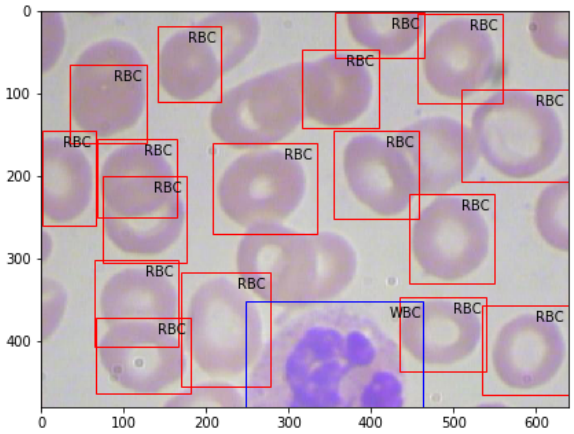

图 1.8　检测图像

$$gitclonehttps://github.com/kbardool/keras-frcnn.git$$

将 train_images 和 test_images 文件夹以及 train.csv 文件移动到克隆的存储库。为了在新数据集上训练模型，输入的格式应该是：

$$filepath, x1, y1, x2, y2, class_name$$

其中：

1.filepath 是训练图像的路径

2.x1 是边界框的 xmin 坐标

3.y1 是边界框的 ymin 坐标

4.x2 是边界框的 xmax 坐标

5.y2 是边界框的 ymax 坐标

6.class_name 是该边界框中类的名称

我们需要将.csv 格式转换为.txt 文件，该文件的格式与上述相同。创建一个新的数据框，按照格式将所有值填充到该数据框中，然后将其保存为.txt 文件。

下一步进行模型的训练

以下是我在实施 Faster R-CNN 后得到的一些预测示例：

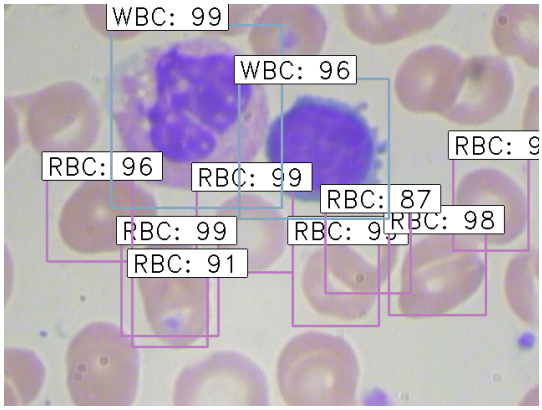

图 1.9　实例 1

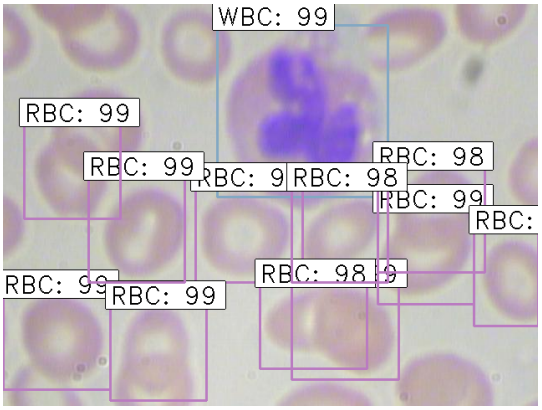

图 1.10　实例 2

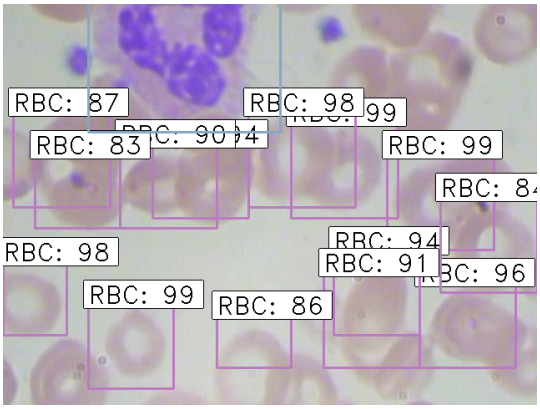

图 1.11　实例 3

这个实验的处理文件的代码都在“实验四的文件中”