应用案例深度解析

5.1 基于卷积神经网络的图像分类

《史记》有记载：赵高指鹿为马，混淆是非；《艾子杂说》中，有人欲以鹘猎兔而不识鹘，买凫捉兔，成为笑谈。上述两个历史典故，都与“分类”相关。千年之后的今天，深度学习技术极大地提高了图像分类的准确性并将其广泛地应用于生活中，如网络图像检索、人脸识别等。本节就来看一下，那些经典的卷积神经网络是否能分得清“鹿”和“马”，是否能认得“鹘”和“凫”。

本节将重点讲解如下内容：

- 图像分类的概念及其评价指标；
- 基于特征提取的分类方法的不足；
- 通过编程体验基于卷积神经网络的图像分类；
- 体验噪声干扰对基于卷积神经网络的图像分类的影响。

5.1.1 什么是图像分类

与文字相比，图像能够提供更加生动、更易理解、更加直观的信息，是人们传递与交换信息的重要来源。本节专注于计算机视觉和人工智能的一个重要研究领域的一个重要问题（关于计算机视觉及其发展，详见本章的“扩展阅读”），即图像分

类。图像分类，顾名思义，是一个输入图像，输出对该图像内容分类的描述；或者是判断图像中是否包含一个已知类别的物体（见图 5.1.1）。图像分类核心是从给定的分类集合中给图像分配一个标签，标签来自预定义的可能类别集。图像分类是目标检测、图像分割、物体跟踪、行为分析等其他高层次计算机视觉任务的基础；它在很多领域都有应用，如互联网图像检索、无人驾驶车辆交通标志识别、医疗图像判读等。

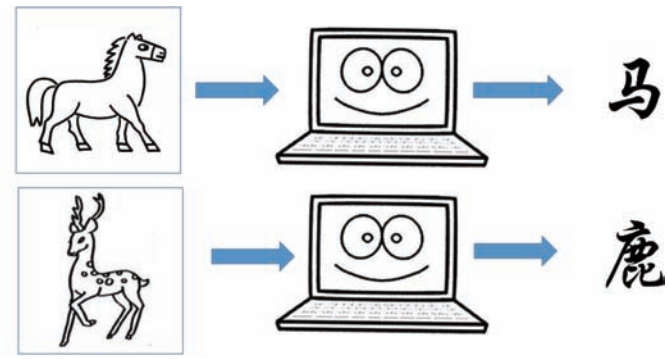

图 5.1.1　图像分类示意图

那么如何对图像分类呢？传统的方法是对输入的图像提取特征（常见的特征包括颜色特征、角点特征、边缘特征、轮廓特征、纹理特征、统计特征等），并对这些特征进行编码（常见的编码方法包括向量量化编码、稀疏编码等），然后，采用特征分类器（常见的特征分类器为支持向量机，对支持向量机的介绍详见 1.3 节）对编码后的特征进行分门别类，以判断图像的类别。在上述过程中，存在如下问题：

问题 1，如何选择区分度高的特征？如果要让计算机识别“马”，仅把棕色作为区分特征显然是不行的，因为不同品种的马，颜色可能不同，即使毛色或皮肤的颜色是棕色的动物，也未必是马；就算联合使用多个特征，也未必能达到高区分度的效果。

问题 2，如何使特征具有稳定性？正所谓“横看成岭侧成峰，远近高低各不同”，对于同一个物体来说，观察的角度不同、位置不同、环境不同，所呈现的图像也是不同的，当然其表现出的特征也不尽相同（见图 5.1.2）。如果把特征分为底层特征（如“马的颜色”“鹿的轮廓”等）和抽象特征（如“有没有犄角”“尾巴的长短”等），越是底层的特征，稳定性越差，如“马的颜色”可能受环境的影响而略有差异，“鹿的轮廓”随观察的角度不同而不同。

问题 3，如何表达抽象性特征？可以通过“有没有犄角”“尾巴的长短”等这样抽象的特征来区分麋鹿和骏马，对于人类而言，这个过程非常简单，只要看一眼图片，大脑就可以获取这些特征。但对于计算机而言，一幅图像就是以特定方式存储的数字（关于数字图像的讨论，详见 3.1 节的“扩展阅读”），让计算机通过一系列计算，从这些数字中提取“有没有犄角”“尾巴的长短”这样的抽象特征，难度极大。

图 5.1.2 图像分类的难点问题

5.1.2 评价分类的指标

常用的评价图像分类性能的指标有准确率、精确率和召回率。准确率是用分类正确的样本数除以所有的样本数,这个概念很好理解。下面着重介绍精确率和召回率。

假设现在有这样一个测试集,测试集中的图片只由鹿和马两种图像组成,假设你的分类系统最终的目的是:能取出测试集中所有鹿的图像,而不是马的图像,测试集中鹿的图像为正样本、马的图像为负样本,进行如下定义:

- tp(true positives)——正样本被正确识别为正样本数量,即鹿的图像被正确地识别成了鹿的图像数量。
- tn (true negatives)——负样本被正确识别为负样本数量,即:马的图像被正确地识别成了马的图像数量。
- fp(false positives)——负样本被错误识别为正样本数量,即:马的图像被错误地识别成了鹿的图像数量。
- fn (false negatives)——正样本被错误识别为负样本数量,即:鹿的图像被错误地识别成了马的图像。

精确率(precision)就是在识别出来的图片中,tp 所占的比率,即被计算机程序识别出来的鹿中,真正的鹿的图像所占的比例。

$$\text{精确率} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

召回率(recall)是测试集中所有正样本中,被正确识别为正样本的比例。也就是本假设中,被计算机程序正确识别出来的鹿的图像个数与测试集中所有鹿的图像个数的比值。

$$\text{召回率} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

5.1.3　基于深度学习和数据驱动的图像分类

随着大数据和深度学习技术的发展，基于深度学习和数据驱动的图像分类方法粉墨登场，并大显身手；特别是在 2012 年的 ImageNet 挑战赛上的优异表现，更是奠定了其“霸主”的地位。基于深度学习和数据驱动的图像分类的主要步骤如下：

步骤 1，构建数据集，用于对深度神经网络模型进行训练和验证。

步骤 2，采用数据集，对深度神经网络进行训练和验证；在训练过程中，深度网络自动提取数据集的特征并与分类标签相对应。

步骤 3，采用新的数据，对训练好的深度神经网络进行测试。

基于深度学习和数据驱动的图像分类过程示意图如图 5.1.3 所示。

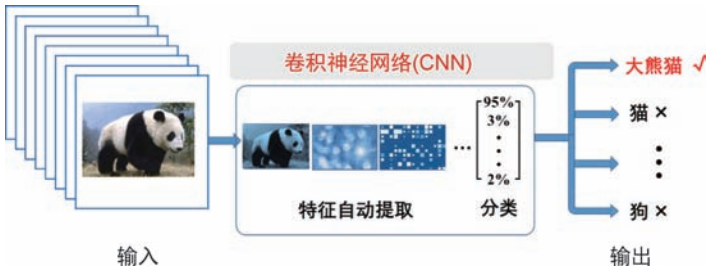

图 5.1.3　基于深度学习和数据驱动的图像分类过程示意图

5.1.4　传统的图像分类与基于深度学习的图像分类的区别

基于深度学习的图像分类与传统的图像分类方法相比，其最大的特点就是它所采用的特征是从大量数据样本中自动学习得到的，并非人为设计；同时，卷积神经网络实现了特征提取与分类的一体化，最大限度地发挥了二者联合协作的性能优势。传统的图像分类方法将特征提取与表示和分类分开进行，效率较低，同时，特征提取与表示也依赖于先验知识。传统的图像分类与基于深度学习的图像分类的区别如图 5.1.4 所示。

5.1.5　基于 AlexNet 的图像分类

例程 5.1.1 是基于 AlexNet 卷积神经网络对 MATLAB 自带图像进行分类的程序，其运行效果如图 5.1.5 所示。

例程 5.1.1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 实例 5.1.1
% 功能：基于 AlexNet 卷积神经网络对 MATLAB 自带图像进行分类
% 作者：zhaoxch_mail@sina.com
```

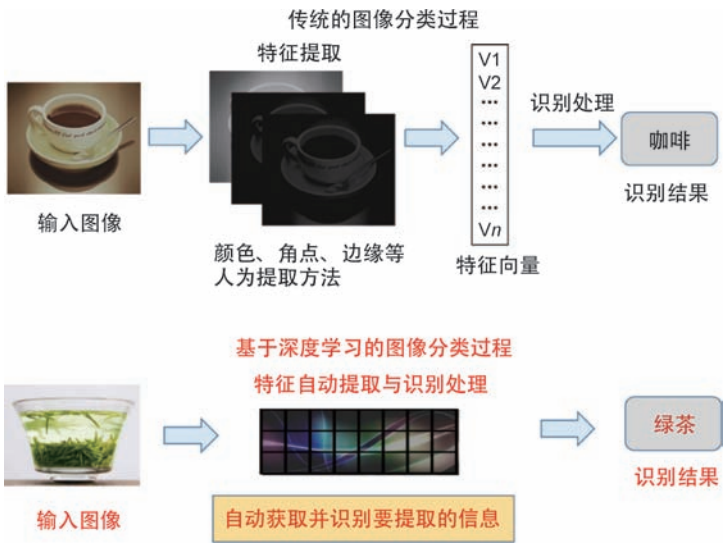

图 5.1.4 传统的图像分类与基于深度学习的图像分类的区别示意图

```
% 时间: 2020 年 3 月 1 日

%% 导入预训练好的 AlexNet,并确定该网络输入图像的大小以及分类种类的名称
net = alexnet; % 将 AlexNet 导入工作区
inputSize = net.Layers(1).InputSize; % 获取 AlexNet 输入层中输入图像的大小
classNames = net.Layers(end).ClassNames; % 获取 AlexNet 输出层中的分类

%% 读入两幅 MATLAB 自带的 RGB 图像,并将图像的大小变换成与 AlexNet 输入层中输入图
%% 像相同的大小
I = imread('peppers.png');
figure
imshow(I)
I = imresize(I,inputSize(1:2));

J = imread('peacock.jpg');
figure
imshow(J)
J = imresize(J,inputSize(1:2));

%% 基于 AlexNet 对两幅输入的图像进行分类
[label1,scores1] = classify(net,I);
[label2,scores2] = classify(net,J);

%% 在图像上显示分类结果及概率
figure
imshow(I)
title(string(label1) + ", " + num2str(100 * scores1(classNames == label1),
3) + "%");
```

```
figure
imshow(J)
title(string(label2) + ", " + num2str(100 * scores1(classNames == label2),3) +
"%");
**************************************************************************
```

图 5.1.5　例程 5.1.1 的运行效果

由图 5.1.5 可知，AlexNet 卷积神经网络对所输入的两幅图像分类正确（bell pepper：彩椒，peacock：孔雀）。

如图 5.1.6 所示，MATLAB 中还自带有多幅 RGB 图像，文件名称如下：

图 5.1.6　部分 MATLAB 自带的可供测试的 RGB 图像

```
'sherlock.jpg','car2.jpg','fabric.png','greens.jpg','hands1.jpg','kobi.png',
'lighthouse.png','micromarket.jpg','office_4.jpg','onion.png','pears.png',
'yellowlily.jpg','indiancorn.jpg','flamingos.jpg','sevilla.jpg','llama.jpg',
'parkavenue.jpg','peacock.jpg','car1.jpg','strawberries.jpg','wagon.jpg'
```

读者可根据需要调用、测试。

5.1.6 基于 GoogLeNet 的图像分类

例程 5.1.1 是对 MATLAB 自带的图像进行分类,下面通过例程 5.1.2 来实现对外部输入的 RGB 图像进行分类。首先,将本书配套资料中的图片 deer.jpg、horse.jpg 复制到 C:\我的文档\MATLAB 文件夹下(注:由于版本及安装路径不同,MATLAB 文件夹的路径也不相同,请读者按照自己计算机上安装 MATLAB 的实际情况进行操作,此处的路径为 C:\Users\zhao\Documents\MATLAB)。

例程 5.1.2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.1.1
% 功能: 基于 GoogLeNet 卷积神经网络对图像进行分类
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 3 月 15 日
% 版本: DLTEXC501 - V1

%% 导入预训练好的 GoogLeNet,并确定该网络输入图像的大小以及分类种类的名称
net = googLeNet;                          % 将 GoogLeNet 导入工作区
inputSize = net.Layers(1).InputSize;      % 获取 GoogLeNet 输入层中输入图像的大小
classNames = net.Layers(end).ClassNames;% 获取 GoogLeNet 输出层中的分类

%% 读入两幅 RGB 图像,并将图像的大小变换成与 GoogLeNet 输入层中输入图像相同的大小
I = imread('deer.jpg');
figure
imshow(I)
I = imresize(I,inputSize(1:2));

J = imread('horse.jpg');
figure
imshow(J)
J = imresize(J,inputSize(1:2));

%% 基于 GoogLeNet 对输入的图像进行分类
[label1,scores1] = classify(net,I);
[label2,scores2] = classify(net,J);

%% 在图像上显示分类结果及概率
```

```
figure
imshow(I)
title(string(label1) + ", " + num2str(100 * scores1(classNames == label1),
3) + "%");

figure
imshow(J)
title(string(label2) + ", " + num2str(100 * scores1(classNames == label2),
3) + "%");
**************************************************************
```

例程 5.1.2 的运行效果如图 5.1.7 所示，成功地对输入图像进行了分类。（注：分类结果中 impala 为黑斑羚，sorrel 为栗色的马。）

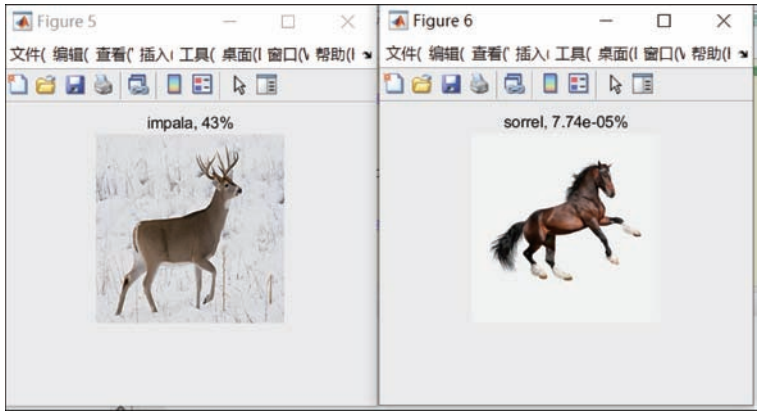

图 5.1.7　例程 5.1.2 的运行效果

5.1.7　基于卷积神经网络的图像分类抗干扰性分析

在例程 5.1.1 和例程 5.1.2 所采用的图像中，目标物体清晰，且不存在任何干扰，这是一种非常理想的情况，在实际生活中，图像中往往存在着某种干扰。下面就来看一下存在干扰时，是否会影响卷积神经网络图像分类的性能。

1. 抗装饰性干扰的分析

所谓装饰性干扰，就是图像中的目标物体被添加了种种装饰，如图 5.1.8 所示，图像中的狗被戴上了眼镜，而在实际生活中，狗是很少被戴眼镜的。下面就来测试一下卷积神经网络能否认得出“戴眼镜的小狗”。

例程 5.1.3 是用 VGG-16 卷积神经网络对如图 5.1.8 所示的图片进行分类的程序（注：在运行例程 5.1.3 时，请将本书配套资料中名为 glassdog.jpg 的图像复制到 C:\我的文档\ MATLAB 文件夹下，由于版本及安装路径不同，MATLAB 文件夹的路径也不相同，请读者按照自己计算机上安装 MATLAB 的实际情况进行操作），其分类结果如图 5.1.9 所示。

图 5.1.8 存在装饰性干扰的图片示例

例程 5.1.3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.1.3
% 功能: 基于 VGG16 卷积神经网络对图像进行分类
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 3 月 15 日

%% 导入预训练好的 VGG16 卷积神经网络,并确定该网络输入图像的大小以及分类种类的名称
net = vgg16;                                % 将 VGG16 卷积神经网络导入工作区
inputSize = net.Layers(1).InputSize;
                                  % 获取 VGG16 卷积神经网络输入层中输入图像的大小
classNames = net.Layers(end).ClassNames; % 获取 VGG16 卷积神经网络输出层中的分类

%% 读入 RGB 图像,并将图像变换成与 VGG16 卷积神经网络输入层中输入图像相同的大小
I = imread('glassdog.jpg');
figure
imshow(I)
I = imresize(I,inputSize(1:2));

%% 基于 VGG16 卷积神经网络对输入的图像进行分类
[label1,scores1] = classify(net,I);

%% 在图像上显示分类结果及概率
figure
imshow(I)
title(string(label1) + ", " + num2str(100 * scores1(classNames == label1),3) +
"%");
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

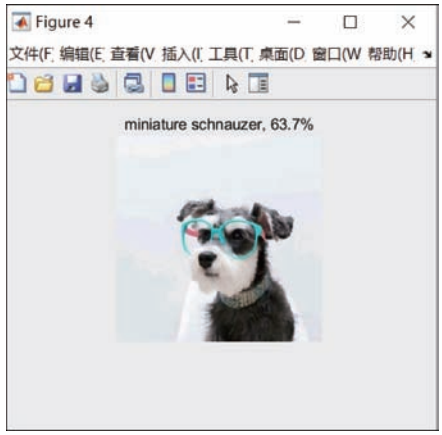

图 5.1.9　例程 5.1.3 的运行效果

由图 5.1.9 可知，VGG-16 网络对图 5.1.8 所示的具有装饰性干扰的图像分类正确(注：分类结果中 miniature schnauzer 为小型雪纳瑞犬)。

感兴趣的读者可以添加其他具有装饰性图像进行测试。

2. 抗噪声性干扰的分析

由例程 5.1.1 可知，AlexNet 可对名为 peppers.png 的图像进行正确分类。下面通过例程 5.1.4 对该图像添加噪声，看一下卷积神经网络抗噪声干扰的性能如何。

例程 5.1.4

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.1.4
% 功能：基于 AlexNet 卷积神经网络对添加噪声图像进行分类
% 作者：zhaoxch_mail@sina.com
% 时间：2020 年 3 月 15 日

%% 导入预训练好的 AlexNet,并确定该网络输入图像的大小以及分类种类的名称
net = alexnet;                              % 将 AlexNet 导入工作区
inputSize = net.Layers(1).InputSize;        % 获取 AlexNet 输入层中输入图像的大小
classNames = net.Layers(end).ClassNames;% 获取 AlexNet 输出层中的分类

%% 读入 MATLAB 自带的 RGB 图像,改变图像大小并添加噪声
I = imread('peppers.png');
figure
imshow(I)
I = imresize(I,inputSize(1:2));
I = imnoise(I,'salt & pepper');             % 添加椒盐噪声
```

```
%% 基于 AlexNet 对添加噪声后的图像进行分类
[label1,scores1] = classify(net,I);

%% 在图像上显示分类结果及概率
figure
imshow(I)
title(string(label1) + ", " + num2str(100 * scores1(classNames == label1),3) +
"%");
*****************************************************************
```

由图 5.1.10 可知,在对图像添加噪声后,AlexNet 卷积神经网络将图像中的彩椒误识别成草莓(strawberry),由此可见其对噪声干扰的脆弱性。

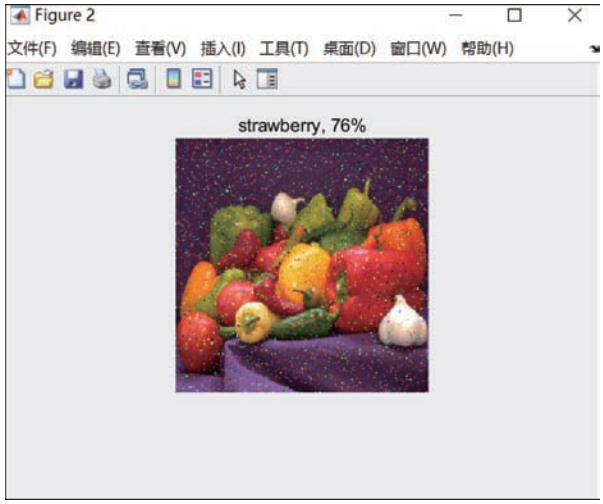

图 5.1.10 例程 5.1.4 的运行效果

| 扩展阅读 |

计算机视觉的发展之路

人类感知外部世界主要是通过视觉、触觉、听觉和嗅觉等感觉器官,其中约 80% 的信息是由视觉获取的,正如“百闻不如一见”“眼睛是心灵的窗户”“眼见为实”。

计算机视觉(Computer Vision, CV)是一门研究如何让计算机达到像人类那样“看”的学科,它是利用视觉传感器和计算机代替人眼和大脑,使得计算机拥有类似于人类的那种对目标进行分割、分类、识别、跟踪、重构、判别决策的能力,它是人工智能领域的一个重要部分。计算机视觉的最终研究目标就是使计算机能像人那样通过视觉观察和理解世界,具有自主适应环境的能力。

计算机视觉的发展史可以追溯到 1966 年,著名的人工智能学家马文·明斯基给他的学生布置了一道非常有趣的暑假作业,就是让学生在计算机前面连一个摄

像机，然后写一个程序，让计算机告诉我们摄像头看到了什么。这是计算机视觉发展的一个起点。

20 世纪 70 年代，研究者认为要让计算机认知到底看到了什么，可能首先要了解人是怎样去理解这个世界的。因为那时有一种普遍的认知，认为人之所以理解这个世界，是因为人有两只眼睛，人看到的世界是立体的，人能够从这个立体的形状里面理解这个世界。在这种认知下，研究者希望先把三维结构从图像中恢复出来，在此基础上再去做理解和判断。

20 世纪 80 年代，人们发现要让计算机理解图像，不一定先要恢复物体的三维结构。例如，让计算机识别一匹马，要让计算机事先知道马的特征；只有建立了这样一个先验知识库，计算机才可能将这样的先验知识和所看到物体的表征进行匹配。如果能够匹配，计算机就算识别或者理解了所看到的物体。因此，特征提取和匹配成为当时计算机视觉研究的主流。

20 世纪 90 年代统计方法的流行，让研究者找到了能够刻画物品特质的一些局部特征（局部不变特征），比如，要识别一辆卡车，可通过形状、颜色、纹理完成，但效果可能并不稳定，如果通过局部特征，即使视角变化了，也会准确对其进行辨识。

2000 年以后，机器学习方法开始盛行。以前需要通过一些规则、知识或者统计模型去识别图像所代表的物品是什么，但是机器学习的方法和以前完全不一样，机器学习能够从海量数据中自动归纳物品的特征，然后去识别它。

2010 年以后，随着 AlexNet 以 15.4%的低失误率，并且领先第二名将近 11%的优势夺得 2012 年 ImageNet 挑战赛的冠军，采用深度学习方法的计算视觉重新得到人们的重视，各大企业（如谷歌、微软、百度、阿里巴巴）等均投入巨大资源进行深度学习研发，并取得了一些突破。

编程体验

体验 GoogLeNet 识别图像的抗噪声能力

编程体验：输入一幅图片（将本书配套资料名为 panda.jpg 的图像复制到 C:\我的文档\ MATLAB 文件夹下，由于版本及安装路径不同，MATLAB 文件夹的路径也不相同，请读者按照自己计算机上安装 MATLAB 的实际情况进行操作），采用 GoogLeNet 对其进行分类，然后添加椒盐噪声，再用 GoogLeNet 对其进行分类。例程 5.1.5 的运行效果如图 5.1.11 所示。

例程 5.1.5

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.1.5
% 功能：基于 GoogLeNet 卷积神经网络对图像及添加噪声的图像分类
```

```
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 3 月 16 日

%% 导入预训练好的 GoogLeNet,并确定该网络输入图像的大小以及分类种类的名称
net = googLeNet;                              % 将 GoogLeNet 导入工作区
inputSize = net.Layers(1).InputSize;          % 获取 GoogLeNet 输入层中输入图像的大小
classNames = net.Layers(end).ClassNames;% 获取 GoogLeNet 输出层中的分类

%% 读入 RGB 图像,改变图像大小,并添加噪声
I = imread('panda.jpg');
figure
imshow(I)
I = imresize(I,inputSize(1:2));
J = imnoise(I,'salt & pepper',0.2);           % 添加椒盐噪声

%% 基于 GoogLeNet 对输入的图像及添加噪声后的图像进行分类
[label1,scores1] = classify(net,I);
[label2,scores2] = classify(net,J);

%% 在图像上显示分类结果及概率
figure
imshow(I)
title(string(label1) + ", " + num2str(100 * scores1(classNames == label1),
3) + "%");

figure
imshow(J)
title(string(label2) + ", " + num2str(100 * scores1(classNames == label2),
3) + "%");
*****************************************************************
```

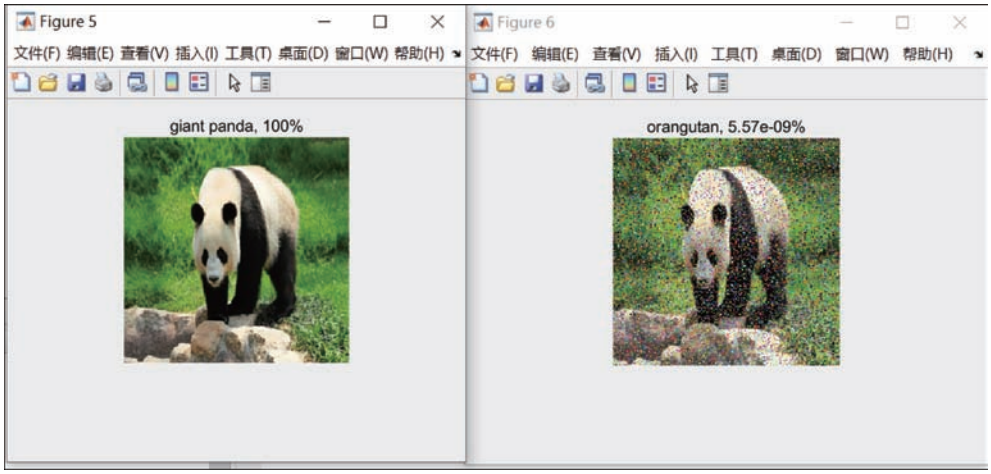

图 5.1.11 例程 5.1.5 的运行效果

5.2　基于 LeNet 卷积神经网络的交通灯识别

交通灯识别在智能驾驶中起着重要的作用。传统的基于特征(如基于颜色、形状等)的方法容易受到光照、视角的影响以及相似物体的干扰,成功率不高。本节通过设计深度卷积网络来进行交通灯的识别。

本节的重点内容如下:

- 如何根据实际需求,对 LeNet 进行改进;
- 巩固第 4 章中所讲解的如何导入训练数据、如何构建网络、如何检查网络、如何训练网络等内容。

5.2.1　实例需求

例 5.2.1　基于 LeNet 的基本构架,设计一种卷积神经网络并对其进行训练,实现对输入的交通灯图像进行分类,计算分类准确率。

5.2.2　卷积神经网络设计

如图 5.2.1 所示,采用 LeNet 为基本构架来设计用于交通灯识别的卷积神经网络,需要将 LeNet 的网络输出分类根据数据集的分类进行改变。关于 LeNet 卷积神经网络的分析与介绍详见 3.4 节。

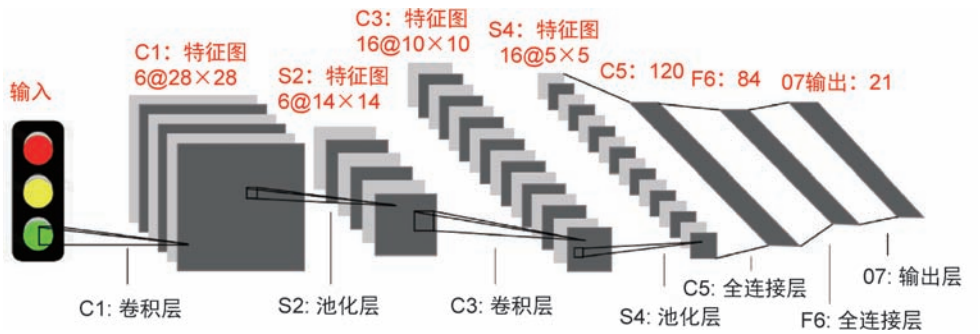

图 5.2.1　基于 LeNet 构架的交通灯分类示意图

改进后的网络结构表 5.2.1 所示。

表 5.2.1　改进后的网络结构

名　称	输　入	卷　积　核	步长	输　出
输入层	像素:60×20 通道数:3	—	—	大小:60×20 通道数:3

续表

名　　称	输　　入	卷　积　核	步长	输　　出
卷积层 1	大小：60×20 通道数：3	大小：5×5 通道数：3 个数：6	1	大小：60×20 通道数：6
池化层 1	大小：60×20 通道数：6	—	2	大小：30×10 通道数：6
卷积层 2	大小：30×10 通道数：6	大小：5×5 通道数：6 个数：16	1	大小：30×10 通道数：16
池化层 2	大小：30×10 通道数：16	—	2	大小：15×5 通道数：16
卷积层 3	大小：15×5 通道数：16	大小：5×5 通道数：16 个数：120	—	大小：15×5 通道数：120
全连接层 1	15×5×120	—	—	84
全连接层 2	84			21(注：数据集的分类数)
Softmax 层	21			21
分类输出层	21	—	—	1

对于如表 5.2.1 所示的网络结构，其实现程序如下：

```
LeNet = [imageInputLayer([60 20 3],'Name','input')
    convolution2dLayer([5 5],6,'Padding','same','Name','Conv1')
    maxPooling2dLayer(2,'Stride',2,'Name','Pool1')
    convolution2dLayer([5 5],16,'Padding','same','Name','Conv2')
    maxPooling2dLayer(2,'Stride',2,'Name','Pool2')
    convolution2dLayer([5 5],120,'Padding','same','Name','Conv3')
    fullyConnectedLayer(84,'Name','fc1')
    fullyConnectedLayer(numClasses,'Name','fc2')
    softmaxLayer('Name','softmax')
    classificationLayer('Name','output') ];
```

5.2.3 加载交通灯数据集

本节中所采用的数据集见本书配套资料中的文件夹 Traffic Light Samples。该数据集中的部分图像如图 5.2.2 所示。

该样本中有 21 个分类，每一个分类都有自己的标签。分类标签由字母数字及下画线组成。具体含义如下：G 代表绿灯，R 代表红灯等。AF 代表向前箭头。AL 代表向左箭头，AR 代表向右箭头，C 代表圆形，数字代表灯的个数，N 代表负样本(不是交通灯)。如图 5.2.3 所示的图像的分类标签为 RC_3，如图 5.2.4 所示的图像的分类标签为 GAL_3

图 5.2.2　数据集中的部分图像

图 5.2.3　分类标签为 RC_3 的图像

图 5.2.4　分类标签为 GAL_3 的图像

将名为 Traffic Light Samples 的文件夹复制到图像数据集在 C 盘的\Documents\MATLAB 下(注：MATLAB 安装在不同的硬盘分区里，路径可能不同。此处的路径为 C:\Users\zhao\Documents\MATLAB)，通过如下程序段进行加载。

```
imds = imageDatastore('Traffic Light Samples', ...
'IncludeSubfolders',true, 'LabelSource','foldernames');
```

5.2.4　程序实现

实现 5.2.1 节的实例需求主要包括以下几个步骤：

步骤 1，加载交通灯数据样本；

步骤 2，将样本划分为训练集与测试集；

步骤 3，构建改进的 LeNet 卷积神经网络并进行分析；

步骤 4，将训练集与验证集中图像的大小调整为与所设计的网络输入层的大小相同；

步骤 5,配置训练选项并对网络进行训练;

步骤 6,将训练好的网络用于对新的输入图像进行分类,并计算准确率;

步骤 7,显示验证效果;

步骤 8,创建并显示混淆矩阵。

上述步骤可以通过例程 5.2.1 来实现。读者可以结合程序的注释以及第 4 章的相关内容进行理解。例程 5.2.1 的运行效果如图 5.2.5～图 5.2.8 所示。

例程 5.2.1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 实例 5.2.1
% 功能: 对输入的交通灯图像进行分类
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 4 月 2 日
% 注: 请将本书配套资料中的 Traffic Light Samples 文件夹复制到 MATLAB 文件下

%% 清除内存、清除屏幕
clear
clc

%% 步骤 1: 加载交通灯数据样本
imds = imageDatastore('Traffic Light Samples', ...
    'IncludeSubfolders',true, ...
'LabelSource','foldernames');

%% 步骤 2: 将样本划分为训练集与测试集,并随机显示训练集中的图像
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7);

% 统计训练集中分类标签的数量
numClasses = numel(categories(imdsTrain.Labels));

% 随机显示训练集中的部分图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end

%% 步骤 3: 构建改进的 LeNet 卷积神经网络并进行分析
% 构建改进 LeNet 卷积神经网络
LeNet = [imageInputLayer([60 20 3],'Name','input')
   convolution2dLayer([5 5],6,'Padding','same','Name','Conv1')
   maxPooling2dLayer(2,'Stride',2,'Name','Pool1')
   convolution2dLayer([5 5],16,'Padding','same','Name','Conv2')
   maxPooling2dLayer(2,'Stride',2,'Name','Pool2')
```

```
    convolution2dLayer([5 5],120,'Padding','same','Name','Conv3')
    fullyConnectedLayer(84,'Name','fc1')
    fullyConnectedLayer(numClasses,'Name','fc2')
    softmaxLayer( 'Name','softmax')
    classificationLayer('Name','output') ];

% 对构建的网络进行可视化分析
lgraph = layerGraph(LeNet);
analyzeNetwork(lgraph)

%% 步骤 4: 将训练集与验证集中图像的大小调整为与所设计的网络输入层的大小相同
inputSize = [60 20 3];
% 将训练图像的大小调整为与输入层的大小相同
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
% 将验证图像的大小调整为与输入层的大小相同
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);

%% 步骤 5: 配置训练选项并对网络进行训练
% 配置训练选项
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.001, ...   % 体会初始学习率为 0.01 以及 0.0001
    'MaxEpochs',3, ...              % 可以对最大轮数进行设置,体会对准确率的影响
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',true, ...
    'Plots','training-progress');

% 对网络进行训练
net = trainNetwork(augimdsTrain,LeNet,options);

%% 步骤 6: 将训练好的网络用于对新的输入图像进行分类,并计算准确率
YPred = classify(net,augimdsValidation);
   YValidation = imdsValidation.Labels;
   accuracy = sum(YPred == YValidation)/numel(YValidation)

%% 步骤 7: 显示验证效果
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end

%% 步骤 8: 创建并显示混淆矩阵
figure
confusionchart(YValidation,YPred)
**********************************************************************
```

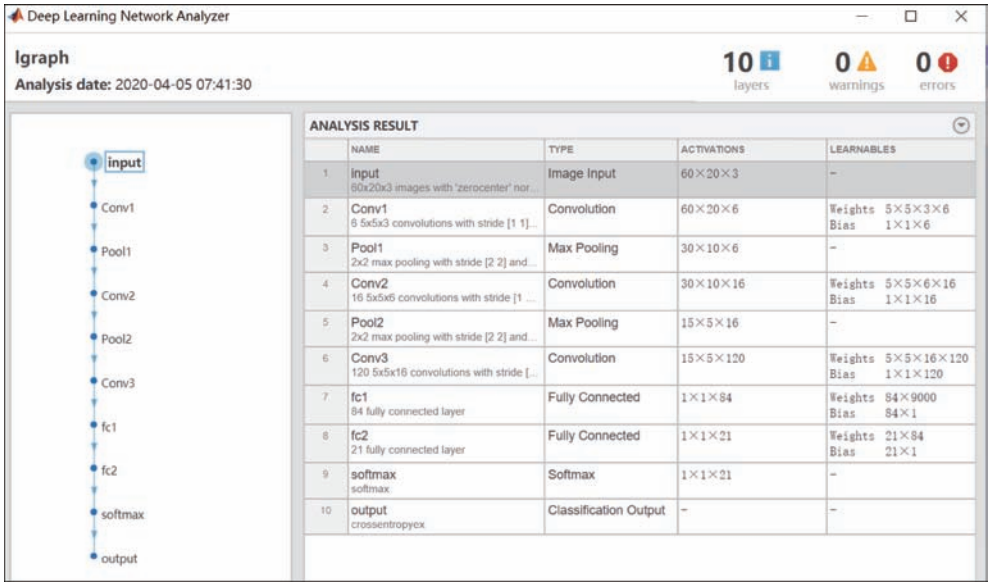

图 5.2.5 采用 Deep Learning Network Analyzer 对网络进行分析

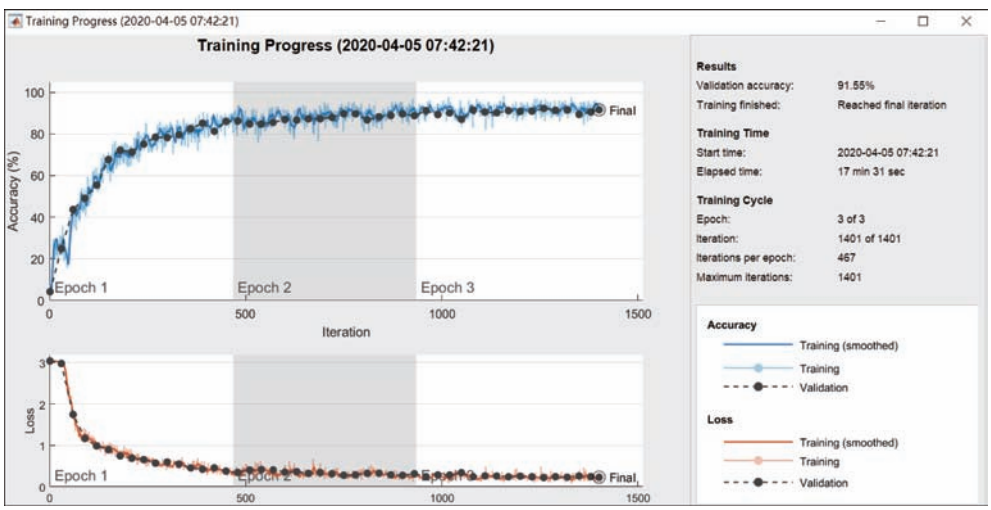

图 5.2.6 网络训练的过程

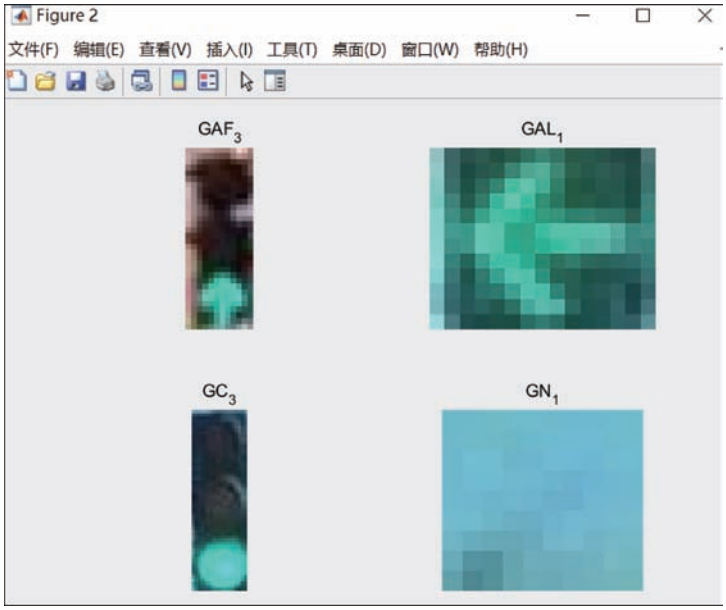

图 5.2.7　随机显示测试的效果

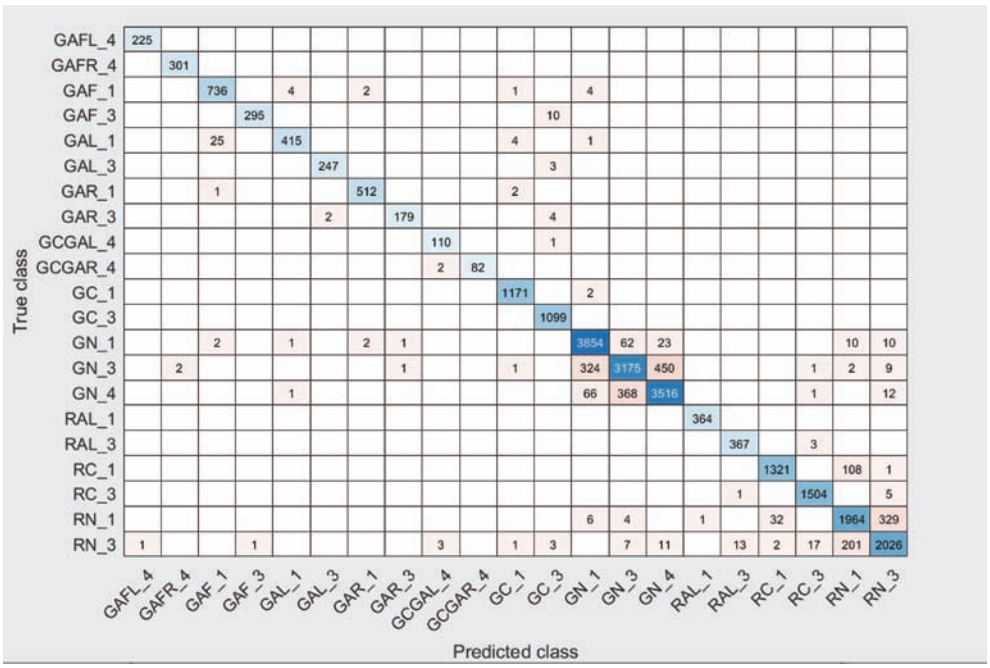

图 5.2.8　混淆矩阵

由于在配置选项中,将'Verbose'设置为 true,所以该网络的训练和验证过程也会在命令窗口中显示,如表 5.2.1 所示。

表 5.2.1 例程 5.2.1 的运行效果在命令窗口显示

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Validation Accuracy	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:21	7.81%	4.22%	3.0434	3.0434	0.0010
1	30	00:00:50	27.34%	24.84%	2.9917	2.9821	0.0010
1	50	00:00:54	31.25%		2.4262		0.0010
1	60	00:01:16	42.19%	43.69%	1.6757	1.7477	0.0010
1	90	00:01:39	50.00%	49.13%	1.1969	1.1690	0.0010
1	100	00:01:41	45.31%		1.3262		0.0010
1	120	00:02:00	56.25%	55.40%	1.0474	0.9887	0.0010
1	150	00:02:29	71.09%	67.71%	0.9436	0.8979	0.0010
1	180	00:02:49	71.88%	72.21%	0.7619	0.7490	0.0010
1	200	00:02:53	72.66%		0.6120		0.0010
1	210	00:03:10	68.75%	71.34%	0.8437	0.6918	0.0010
1	240	00:03:31	78.13%	75.11%	0.6321	0.6572	0.0010
1	250	00:03:33	77.34%		0.6102		0.0010
1	270	00:03:51	74.22%	78.45%	0.5568	0.5738	0.0010
1	300	00:04:10	82.03%	78.21%	0.5187	0.5970	0.0010
1	330	00:04:30	79.69%	79.63%	0.6066	0.5501	0.0010
1	350	00:04:34	81.25%		0.4331		0.0010
1	360	00:04:58	78.91%	82.51%	0.5205	0.4559	0.0010
1	390	00:05:22	82.03%	85.20%	0.4436	0.4258	0.0010
1	400	00:05:24	85.16%		0.4659		0.0010
1	420	00:05:44	84.38%	81.29%	0.4423	0.4566	0.0010
1	450	00:06:06	85.94%	86.02%	0.4426	0.3777	0.0010
2	480	00:06:27	92.19%	86.21%	0.2740	0.3452	0.0010
2	500	00:06:31	83.59%		0.5579		0.0010
2	510	00:06:48	87.50%	84.79%	0.3393	0.4049	0.0010
2	540	00:07:08	84.38%	84.78%	0.3246	0.4173	0.0010
2	550	00:07:10	82.81%		0.4454		0.0010
2	570	00:07:30	87.50%	85.56%	0.3448	0.4088	0.0010
2	600	00:07:50	82.81%	86.91%	0.4438	0.3536	0.0010
2	630	00:08:12	91.41%	86.66%	0.2421	0.3640	0.0010
2	650	00:08:16	85.94%		0.3146		0.0010
2	660	00:08:35	87.50%	87.25%	0.3003	0.3235	0.0010
2	690	00:08:56	85.16%	87.28%	0.5747	0.3444	0.0010
2	700	00:08:57	89.06%		0.4063		0.0010
2	720	00:09:16	87.50%	87.96%	0.3510	0.3147	0.0010
2	750	00:09:37	92.19%	89.68%	0.2726	0.2794	0.0010
2	780	00:09:59	89.06%	89.63%	0.3025	0.2875	0.0010
2	800	00:10:02	86.72%		0.3849		0.0010
2	810	00:10:22	89.84%	86.72%	0.2550	0.3263	0.0010
2	840	00:10:45	89.06%	88.26%	0.2977	0.3274	0.0010
2	850	00:10:47	86.72%		0.3011		0.0010
2	870	00:11:06	88.28%	88.88%	0.2289	0.2891	0.0010
2	900	00:11:33	85.16%	89.61%	0.3398	0.2763	0.0010
2	930	00:11:55	91.41%	88.78%	0.2194	0.3144	0.0010

续表

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Validation Accuracy	Mini-batch Loss	Validation Loss	Base Learning Rate
3	950	00:11:59	92.97%		0.2410		0.0010
3	960	00:12:19	91.41%	91.15%	0.2790	0.2444	0.0010
3	990	00:12:40	92.19%	89.31%	0.2748	0.2857	0.0010
3	1000	00:12:42	91.41%		0.2427		0.0010
3	1020	00:13:01	92.19%	90.12%	0.1944	0.2831	0.0010
3	1050	00:13:22	91.41%	87.02%	0.2794	0.3399	0.0010
3	1080	00:13:44	92.19%	91.44%	0.1798	0.2448	0.0010
3	1100	00:13:48	90.63%		0.2801		0.0010
3	1110	00:14:06	93.75%	90.49%	0.1786	0.2634	0.0010
3	1140	00:14:26	94.53%	90.20%	0.1955	0.2602	0.0010
3	1150	00:14:28	94.53%		0.1498		0.0010
3	1170	00:14:46	93.75%	91.14%	0.1785	0.2416	0.0010
3	1200	00:15:07	92.97%	90.81%	0.2454	0.2567	0.0010
3	1230	00:15:28	90.63%	90.95%	0.2325	0.2408	0.0010
3	1250	00:15:32	91.41%		0.2481		0.0010
3	1260	00:15:49	92.19%	92.22%	0.1924	0.2237	0.0010
3	1290	00:16:08	89.84%	91.37%	0.2334	0.2445	0.0010
3	1300	00:16:10	92.97%		0.2195		0.0010
3	1320	00:16:28	92.19%	91.48%	0.2337	0.2311	0.0010
3	1350	00:16:48	90.63%	89.37%	0.2031	0.2621	0.0010
3	1380	00:17:11	88.28%	90.51%	0.2234	0.2449	0.0010
3	1400	00:17:15	93.75%		0.1551		0.0010
3	1401	00:17:31	92.97%	91.55%	0.1967	0.2318	0.0010

请读者在理解例程 5.2.8 的基础上，尝试对所设计的网络的结构、学习参数进行调整，实现识别成功率的提升。

5.3　融合卷积神经网络与支持向量机的图像分类

在 5.1 节中讲到，在传统的图像分类方法中，特征需要人工提取，如何选择合理的特征至关重要；在 3.2 节的学习过程中，也讲到卷积神经网络的卷积层可以自动提取输入样本的特征进行学习。那么，是否可以采用卷积神经网络进行特征提取，然后将所提取的特征采用传统的分类方法进行分类呢？答案是肯定的。本节将重点讲解融合卷积神经网络与支持向量机的图像分类方法。

本节重点讲解如下内容：

- 如何联合使用 CNN 和 SVM 来对图像进行分类；
- 利用从 CNN 中不同的层抽取特征对分类结果的影响。

5.3.1 整体思路

支持向量机是一种分类器,其基本原理已在1.3节介绍过,但基于支持向量机的分类需要输入特征,分类器基于特征进行分类;而卷积神经网络可以自动地提取特征并进行层层抽象。因此对于输入图像,可以通过卷积神经网络进行特征提取,并将提取的特征输入支持向量机分类器中,进行分类,整体思路如图5.3.1所示。

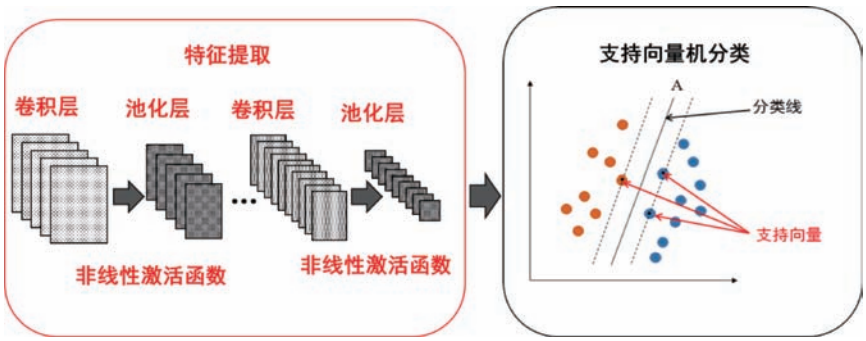

图5.3.1 分类的整体思路示意图

5.3.2 本节所用到的函数

特征提取与激活函数为activations。

功能:提取卷积神经网络某层输出的特征信息。

用法:features = activations(net,data,layer,Name,Value)

输入:net——卷积神经网络。

data——输入卷积神经网络的数据。

layer——待提取特征的层的名称。

Name和Value用来设置其他参数的值,如特征输出的形式等。

例如:

```
featuresTrain = activations(net,augimdsTrain,'pool5','OutputAs','rows');
```

上述语句实现的功能是提取名为net卷积神经网络在augimdsTrain数据的驱动下pool5层的特征,输出的形式为列向量。

5.3.3 实现步骤与程序

例程5.3.1实现的是将VGG16卷积的'pool5'层的输出特征作为输入,拟合SVM分类器,对输入的图像进行分类。例程5.3.1的运行效果如图5.3.2所示。

例程 5.3.1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.3.1
% 功能：基于卷积神经网络与支持向量机对输入进行分类(提取 vgg16 的 pool5 层)
% 作者：zhaoxch_mail@sina.com
% 时间：2020 年 4 月 6 日

%% 清除内存、清除屏幕
clear
clc

%% 导入数据集,划分训练集与测试集,并随机显示训练集中的 16 幅图像
% 导入数据集
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

%将数据集划分为训练集与测试集
[imdsTrain,imdsTest] = splitEachLabel(imds,0.7,'randomized');

%随机显示其中的 16 幅图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end

%% 加载训练好的网络并显示网络结构
net = vgg16;
analyzeNetwork(net)

%% 将数据集图像大小调整到与网络的大小相同
inputSize = net.Layers(1).InputSize
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);

%% 提取卷积神经网络中 'pool5'层的特征
layer = 'pool5';
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');

%% 用提取的卷积神经网络(pool5)层的特征拟合 SVM 分类器
YTrain = imdsTrain.Labels;
```

```
classifier = fitcecoc(featuresTrain,YTrain);

%% 用测试集测试分类器的精度,并随机显示测试结果
% 计算准确度
YPred = predict(classifier,featuresTest);
YTest = imdsTest.Labels
accuracy = mean(YPred == YTest)

% 显示分类结果
idx = [1 5 10 15];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTest,idx(i));
    label = YPred(idx(i));
    imshow(I)
    title(char(label))
end
*****************************************************************
```

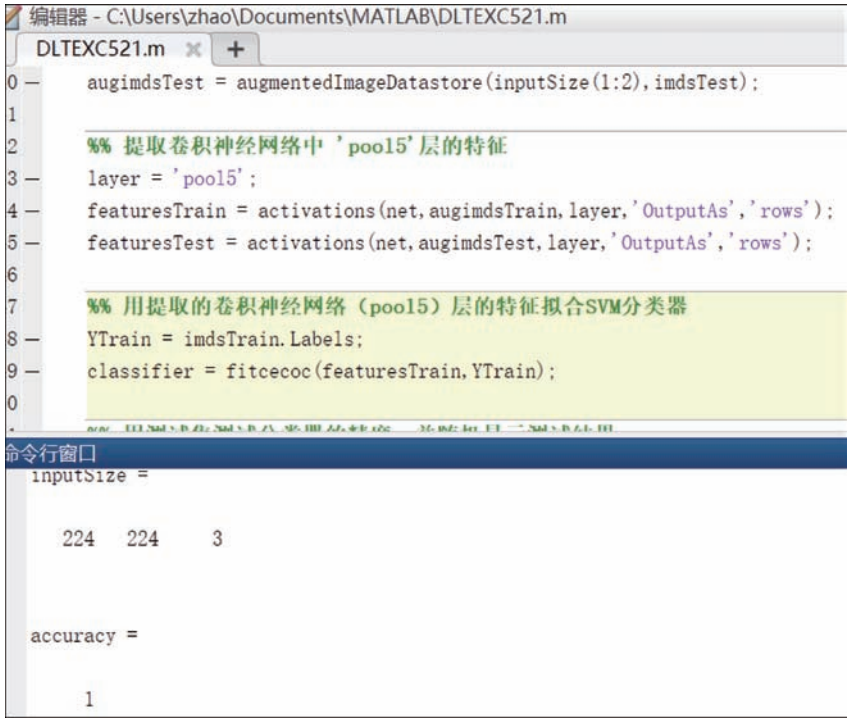

图 5.3.2 例程 5.3.1 的运行效果

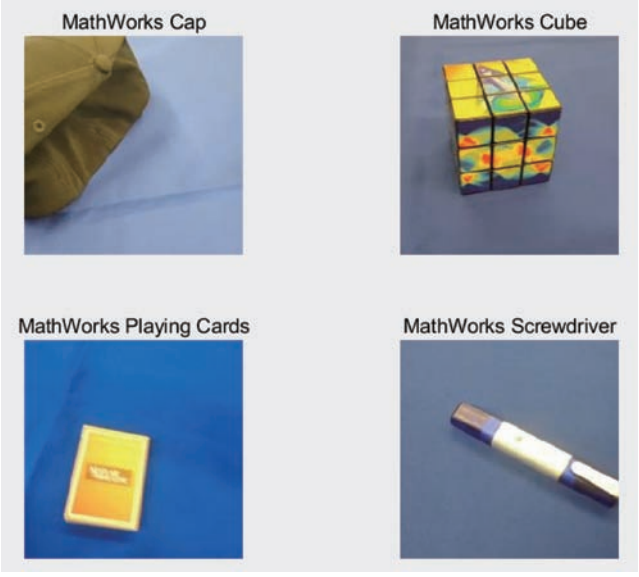

图 5.3.2　（续）

如果想要看一下将其他层的特征输入 SVM 分类器的分类效果，可以将例程 5.3.1 中采用 activations 函数提取网络中的层的名称改一下。例程 5.3.2 实现的便是提取 vgg16 网络 pool3 层特征并用于训练 SVM 分类器的程序，其运行效果如图 5.3.3 所示。

例程 5.3.2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.3.2
% 功能：基于卷积神经网络与支持向量机对输入进行分类(提取 vgg16 的 pool3 层)
% 作者：zhaoxch_mail@sina.com
% 时间：2020 年 4 月 6 日

%% 清除内存、清除屏幕
clear
clc

%% 导入数据集，划分训练集与测试集，并随机显示训练集中的 16 幅图像
% 导入数据集
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

%将数据集划分为训练集与测试集
[imdsTrain,imdsTest] = splitEachLabel(imds,0.7,'randomized');
```

```
% 随机显示其中的 16 幅图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end

%% 加载训练好的网络并显示网络结构
net = vgg16;
analyzeNetwork(net)

%% 将数据集图像大小调整到与网络的大小相同
inputSize = net.Layers(1).InputSize
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);

%% 提取卷积神经网络中 pool3 层的特征
layer = 'pool3';
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');

%% 用提取的卷积神经网络 pool3 层的特征拟合 SVM 分类器
YTrain = imdsTrain.Labels;
classifier = fitcecoc(featuresTrain,YTrain);

%% 用测试集测试分类器的精度,并随机显示测试结果
% 计算准确度
YPred = predict(classifier,featuresTest);
YTest = imdsTest.Labels;
accuracy = mean(YPred == YTest)

% 显示分类结果
idx = [1 5 10 15];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTest,idx(i));
    label = YPred(idx(i));
    imshow(I)
    title(char(label))
end
*****************************************************************
```

为了比较上述卷积神经网络不同层提取的特征对分类效果的不同影响,分别抽取了 pool1、pool2、pool3、pool4、pool5 层的特征,用于支持向量机算法的分类,效果对比如图 5.3.4 所示。由图 5.3.4 可知,层数越高,抽取的特征越抽象,用于分类的效果越好。

```
编辑器 - C:\Users\zhao\Documents\MATLAB\DLTEXC521.m*
DLTEXC521.m*
40 —    augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);
41
42      %% 提取卷积神经网络中 'pool3'层的特征
43 —    layer = 'pool3';
44 —    featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
45 —    featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');
46
47      %% 用提取的卷积神经网络（pool3）层的特征拟合SVM分类器
48 —    YTrain = imdsTrain.Labels;
49 —    classifier = fitcecoc(featuresTrain,YTrain);
50
命令行窗口
inputSize =

   224   224     3


accuracy =

    0.7500
```

图 5.3.3　例程 5.3.2 的运行效果

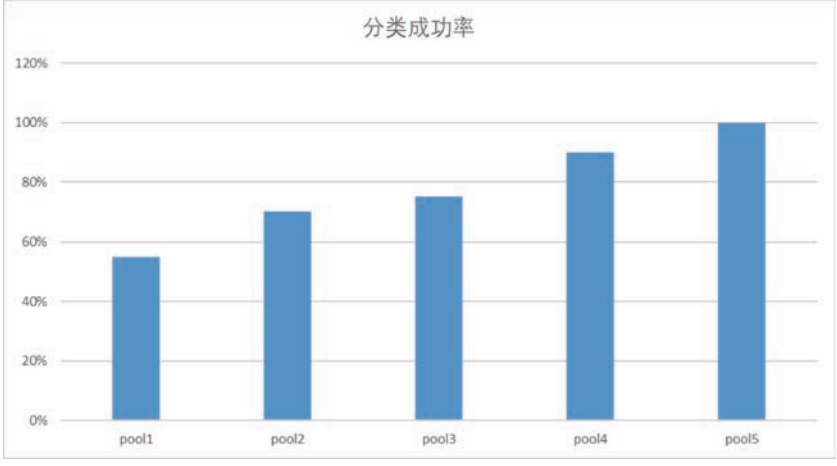

图 5.3.4　采用不同层特征的分类成功率对比

编程体验

基于 AlexNet 和 SVM 的图像分类

编程体验：提取 AlexNet 的 fc7 层特征，输入 SVM 图像分类器，实现对输入图像的分类。

例程 5.3.3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.3.3
% 功能: 基于卷积神经网络与支持向量机对输入进行分类(提取 AlexNet 的 fc7 层)
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 4 月 6 日

%% 清除内存、清除屏幕
clear
clc

%% 导入数据集,划分训练集与测试集,并随机显示训练集中的 16 幅图像
% 可加载数据
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

% 将数据分成训练集和测试集
[imdsTrain,imdsTest] = splitEachLabel(imds,0.7,'randomized');

% 随机显示其中的 16 幅图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end

%% 加载训练好的网络并显示网络结构
% 加载预训练网络
net = alexnet;
% 查看网络结构
net.Layers

%% 将数据集图像大小调整到与网络的大小相同
inputSize = net.Layers(1).InputSize
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);

%% 提取卷积神经网络中 'fc7'层的特征
layer = 'fc7';
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');

% 提取类别标签
```

```
YTrain = imdsTrain.Labels;
YTest = imdsTest.Labels;

%% 用提取的卷积神经网络(fc7)层的特征拟合 SVM 分类器
classifier = fitcecoc(featuresTrain,YTrain);

%% 用测试集测试分类器的精度,并随机显示测试结果
% 对测试集进行分类
YPred = predict(classifier,featuresTest);

% 预览结果
idx = [1 5 10 15];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTest,idx(i));
    label = YPred(idx(i));
    imshow(I)
    title(char(label))
end

% 计算准确率
accuracy = mean(YPred == YTest)
**********************************************************************
```

例程 5.3.3 的运行效果如图 5.3.5 所示,读者可以结合注释以及第 4 章讲解的内容对程序进行深入的了解。

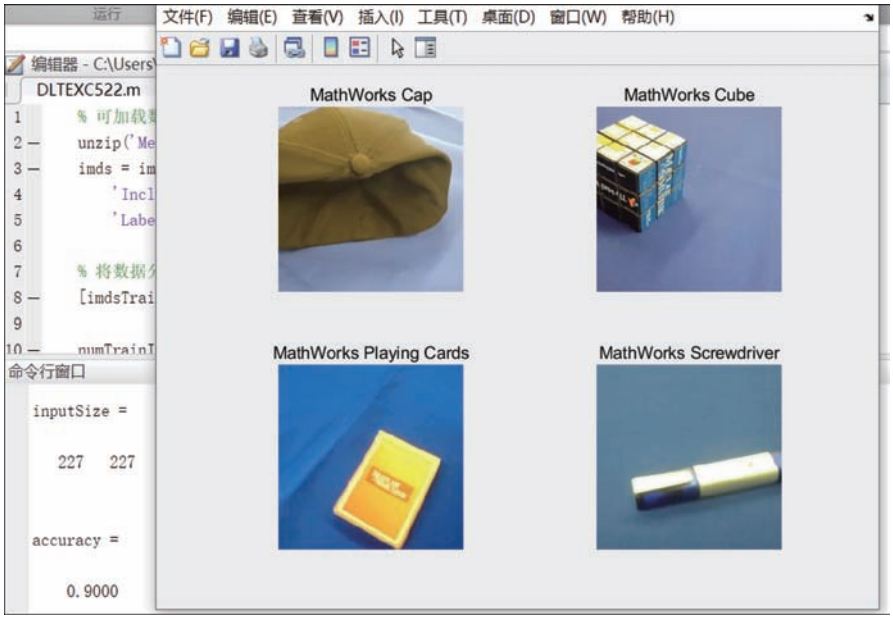

图 5.3.5　例程 5.3.3 的运行效果

5.4 基于 R-CNN 的交通标志检测

目标检测是计算机视觉的一个重要领域,如今,基于深度学习的目标检测技术如雨后春笋,破土而出、蓬勃发展。本节主要介绍基于经典目标检测算法 R-CNN(Regions with CNN features)的交通标志检测。

本节的主要内容包括:

- 目标检测的概念及评价指标;
- R-CNN 的原理及实现步骤;
- 如何使用 Image Labeler 来构建用于训练目标分类器的数据。

5.4.1 目标分类、检测与分割

在计算机视觉领域,目标分类、检测与分割是常用的技术,三者有哪些联系和区别呢?

目标分类,解决的是图像中的物体"是什么"的问题。

目标检测,解决的是图像中的物体(可能有多个物体)"是什么""在哪里"这两个问题。

目标分割,目的是将目标和背景分离,找出目标的轮廓线。

目标分类、检测与分割三者的联系与区别如图 5.4.1 所示。

图 5.4.1 目标分类、检测与分割三者的联系与区别

因此,衡量目标检测性能优劣的指标一方面要体现其分类特性(如准确率、精确率、召回率);另一方面要体现其定位特征。对于定位特性,常用交并比(IoU)来评判。交并比是计算两个边界框交集和并集之比;它衡量了两个边界框的重叠程度。一般约定,在计算机检测任务中,如果 IoU≥0.5,则表示检测正确。IoU 越高,边界框越精确。如果预测器和实际边界框完美重叠,IoU 就是 1,因为交集就等于并集。

5.4.2 目标检测及其难点问题

目标检测就是找出图像中所有感兴趣的目标(物体),确定它们的种类和位置。

由于各类物体有不同的外观、形状、姿态，加上成像时光照、遮挡等因素的干扰，目标检测一直是机器视觉领域最具挑战性的问题之一；此外，目标检测的难点还包括：目标可能出现在图像的任何位置，目标有各种不同的大小，目标可能有各种不同的形状。

传统的目标检测方法是以特征提取与匹配为核心，其流程如下：

（1）区域选择（穷举策略：采用不同大小、不同长宽比的窗口滑动遍历整幅图像，时间复杂度高）。

（2）特征提取（经典的特征提取方法有 SIFT、HOG 等；但角度变化、形态变化、光照变化、背景复杂使得特征鲁棒性差）。

（3）通过分类器对所选择区域进行分类（常用的分类器有 SVM、AdaBoost 等）。

随着深度学习技术的发展，此类方法逐渐被基于卷积神经网络的深度学习目标检测算法所取代。当前，主流的基于卷积神经网络的深度学习目标检测算法主要包括两大类。第一类算法将目标检测划分成两个较为独立的阶段：第一阶段从可能的目标区域里筛选出候选框，从而提取相应的图像特征；第二阶段采用分类器对候选框数据进行处理，从而确定最终的目标边框和类别。第一类算法包括原始 R-CNN、Fast R-CNN、Faster R-CNN，此类算法的优点是目标检测结果准确率高，缺点是由于拆分成独立的两个步骤，影响了算法的运行效率。第二类算法的核心思路是通过一轮图像数据运算，同时确定目标边框和目标类别，此类算法的典型代表是 SDD 和 YOLO，此类算法的优点是运行效率高，但准确率不如第一类算法高。

5.4.3　R-CNN 目标检测算法的原理及实现过程

2012 年，AlexNet 在 ImageNet 上一鸣惊人，受此启发，科研工作者尝试将 AlexNet 在图像分类上的能力迁移到目标检测上。众所周知，将图像分类网络迁移到目标检测上，应主要解决两个难点问题：

（1）如何利用卷积网络去目标在图像中的定位；

（2）如何通过小规模数据集上训练出较好的效果。

2014 年，Ross Girshick（罗斯·吉瑞克）等人发表了题为 *Rich feature hierarchies for accurate object detection and semantic segmentation* 的论文，提出了使用“候选区域＋卷积神经网络”的方法代替传统目标检测使用的“滑动窗口＋手工设计特征”的方法，提出了 R-CNN 框架，使得目标检测取得巨大突破，并掀起了基于深度学习目标检测的热潮。

R-CNN 利用候选区域（region proposal）的方法，解决了图像中的定位问题，这也是该网络被称为 R-CNN 的原因；对于小规模数据集的问题，R-CNN 利用 AlexNet 在 ImageNet 上预训练好的模型，基于迁移学习的原理，对参数进行微调。

基于 R-CNN 的进行目标检测的核心思路如图 5.4.2 所示。

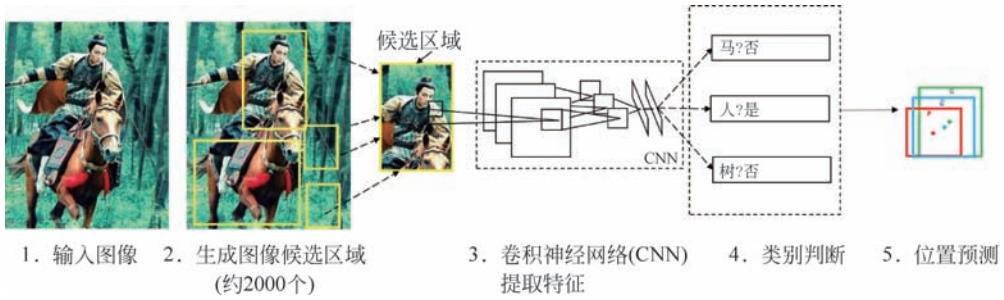

图 5.4.2 基于 R-CNN 的进行目标检测的核心思路

步骤 1,对于输入图像,通过选择性搜索(selective search)算法找到大约 2000 个候选区域。

选择性搜索算法的具体实现过程如下:

(1) 生成初始区域集合;

(2) 计算区域集合中所有相邻区域的相似度(相似度综合考虑了颜色、纹理、尺寸和空间交叠);

(3) 合并相似度最高的两个区域,并移除所有与这两个区域有关的区域;

(4) 重新计算合并的区域和其他所有区域的相似度并执行合并过程直到结束(达到阈值)。

步骤 2,将候选区域的尺寸进行调整,这是因为 CNN 模型中的全连接层需要固定的尺寸输入。

步骤 3,用卷积神经网络提取各候选区域特征。

步骤 4,用支持向量机对候选区域的目标进行种类判别。

步骤 5,对候选区域的边框进行预测调整,使预测边框与真实边框更接近。

目标检测是在多个候选区域上分别执行的,最终必然会产生大量的预测边框,而最终希望得到一个最好的边框来确定目标的位置,抑制冗余的矩形框,保留最优边框。具体来说,对于某一个目标,R-CNN 模型框除了有很多预测边框,每一矩形框都会有一个对应的分类概率,将它们从大到小排序,然后舍弃与最大概率的矩形框重合度高的矩形框,保留剩下的矩形框;然后再对次大概率的矩形框进行同样操作,直到各预测边框具有较高的分类概率,且重合度较低。

5.4.4 实例需求

例 5.4.1 设计一个 R-CNN 目标检测深度网络,用于检测图像中的 stop sign 的交通路标。

图 5.4.3　需求实例示意图

5.4.5　实现步骤

实现 5.4.4 节的实例需求主要包括以下几个步骤：

步骤 1，构建一个卷积神经网络；

步骤 2，采用 CIFAR-10 数据集，训练所构建的卷积神经网络；

步骤 3，验证卷积神经网络的分类效果；

步骤 4，基于迁移学习的原理，在上述步骤构建的卷积神经网络的基础上，通过 41 张包括 stop sign 的图像训练 R-CNN 检测器；

步骤 5，检验检测器对 stop sign 图像的检测效果。

5.4.6　本节所用到的函数

训练 R-CNN 检测器：trainRCNNObjectDetector

功能：实现对 R-CNN 检测器的训练。

用法：rcnn = trainRCNNObjectDetector(traindata, net, options, ...
'NegativeOverlapRange', [0 a], 'PositiveOverlapRange',[b 1])

输入：traindata——训练数据；
net——已有的或完成预训练的卷积神经网络；
options——训练策略参数。

在训练 R-CNN 的过程中，依据根据预测边框与目标实际边框的重合面积将其分为正向重合区域 PositiveOverlapRange（范围为[b 1]）和负向重合区域 NegativeOverlapRange（范围为[0 a]）。

输出：rcnn 为训练好的目标检测网络。

例如，rcnn = trainRCNNObjectDetector(stopSigns, net, options, ...
'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

这个语句实现的功能为：基于 stopSigns 数据集，对预训练好的网络 net 采用

options所定义的训练策略进行rcnn的训练,将重合区域比例在[0 0.3]的区域定义为负向重合区域,将重合区域比例在[0.5 1]的区域定义为正向重合区域。

5.4.7 程序实现

5.4.5节所述的步骤可以通过例程5.4.1来实现。读者可以结合5.4.6节的函数解析、程序的注释以及第4章的相关内容进行理解。例程5.4.7的运行效果如图5.4.4所示。

注意:本例需要用到CIFAR-10数据集,请读者按照4.7.3节介绍的步骤进行操作。

例程5.4.1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.4.1
% 功能: 训练 R-CNN 用于识别交通标志
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 4 月 19 日

%% 步骤 1: 构建一个卷积神经网络
% 输入层(与训练集图像的大小相同)
inputLayer = imageInputLayer([32 32 3]);
% 卷积层
filterSize = [5 5];            % 卷积核大小
numFilters = 32;               % 卷积核个数
middleLayers = [
convolution2dLayer(filterSize, numFilters, 'Padding', 2)
reluLayer()
maxPooling2dLayer(3, 'Stride', 2)
convolution2dLayer(filterSize, numFilters, 'Padding', 2)
reluLayer()
maxPooling2dLayer(3, 'Stride',2)
convolution2dLayer(filterSize, 2 * numFilters, 'Padding', 2)
reluLayer()
maxPooling2dLayer(3, 'Stride',2)
];
% 全连接层
finalLayers = [
fullyConnectedLayer(64)
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer
];
% 构建整个卷积神经网络
layers = [
    inputLayer
    middleLayers
```

```
    finalLayers
    ];

%% 步骤 2: 采用 CIFAR-10 数据集,训练所构建的卷积神经网络;
% 导入 CIFAR-10 数据集,要求与步骤详见 4.7 节
[trainingImages,trainingLabels,testImages,testLabels] = helperCIFAR10Data.
load('cifar10Data');

% 显示其中的 100 幅
figure
thumbnails = trainingImages(:,:,:,1:100);
montage(thumbnails)

% 设置训练策略参数
opts = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
  'LearnRateDropPeriod', 8, ...
    'MaxEpochs', 25, ...
    'MiniBatchSize', 128, ...
    'Verbose', true);

% 训练网络,trainNetwork 函数的参数依次分别为: 训练数据集,训练集标签,网络结构,
% 训练策略
    cifar10Net = trainNetwork(trainingImages, trainingLabels, layers, opts);

%% 步骤 3: 验证卷积神经网络的分类效果
YTest = classify(cifar10Net, testImages);
% 计算正确率.
accuracy = sum(YTest == testLabels)/numel(testLabels)

%% 步骤 4: 训练 RCNN 检测器
% 导入 41 张包括 stop sign 的图像
data = load('stopSignsAndCars.mat', 'stopSignsAndCars');
stopSignsAndCars = data.stopSignsAndCars;
% 设置图像路径参数
visiondata = fullfile(toolboxdir('vision'),'visiondata');
stopSignsAndCars.imageFilename = fullfile(visiondata, stopSignsAndCars.imageFilename);
% 显示数据
summary(stopSignsAndCars)
% 只保留文件名及其所包含的 stop sign 区域
stopSigns = stopSignsAndCars(:, {'imageFilename','stopSign'});
% 显示一张照片及其所包含的真实 stop sign 区域
I = imread(stopSigns.imageFilename{1});
I = insertObjectAnnotation(I,'Rectangle',stopSigns.stopSign{1},'stop sign',
'LineWidth',8);
figure
```

```
imshow(I)

% 设置训练策略
    options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 35, ...
        'Verbose', true);

  % 训练 R-CNN 网络
    rcnn = trainRCNNObjectDetector(stopSigns, cifar10Net, options, ...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

% 载入测试图片
testImage = imread('stopSignTest.jpg');
%% 步骤 5: 检验检测器对 stop sign 图像的检测效果
% 检测 stop sign
[bboxes,score,label] = detect(rcnn,testImage,'MiniBatchSize',128)
% 标注置信度
[score, idx] = max(score);
bbox = bboxes(idx, :);
annotation = sprintf('%s: (Confidence = %f)', label(idx), score);
outputImage = insertObjectAnnotation(testImage, 'rectangle', bbox, annotation);
figure
imshow(outputImage)
**********************************************************************
```

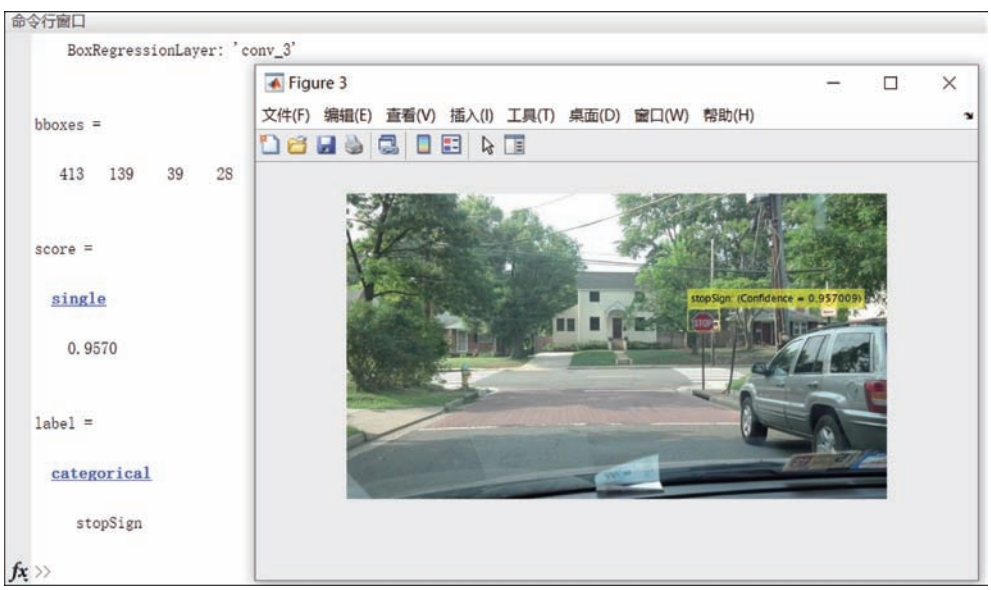

图 5.4.4 例程 5.4.1 的运行效果

5.4.8　基于 AlexNet 迁移学习的 R-CNN 实现

例程 5.4.1 构建了一个新的卷积神经网络，采用 CIFAR-10 数据集对网络预训练，然后再基于 41 幅包括 stop sign 的图像训练 R-CNN 目标检测网络。那么，已有的成熟网络是否可以通过训练，形成 R-CNN 目标检测网络呢？答案是肯定的。

在 AlexNet 的基础上，基于迁移学习，来训练 R-CNN 目标检测网络的步骤如下：

步骤 1，导入 AlexNet。

步骤 2，基于迁移学习的原理，在 AlexNet 卷积神经网络的基础上，通过 41 幅包括 stop sign 的图像训练 R-CNN 检测器。

步骤 3，检验检测器对 stop sign 图像的检测效果。

上述步骤可以通过例程 5.4.2 来实现，其运行效果如图 5.4.5 所示。

注意：关于 AlexNet 的下载及安装方式，请参见 4.3.4 节，此处不再赘述。

例程 5.4.2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.4.2
% 功能：基于 AlexNet 训练 R-CNN 用于识别交通标志
% 作者：zhaoxch_mail@sina.com
% 时间：2020 年 4 月 19 日

%% 步骤 1：载入 AlexNet，载入的方法详见 4.3.4 节
net = alexnet;

%% 步骤 2：训练 R-CNN 检测器
% 载入训练集
data = load('stopSignsAndCars.mat', 'stopSignsAndCars');
stopSignsAndCars = data.stopSignsAndCars;
% 设置图像路径参数
visiondata = fullfile(toolboxdir('vision'),'visiondata');
stopSignsAndCars.imageFilename = fullfile(visiondata, stopSignsAndCars.imageFilename);
% 显示数据
summary(stopSignsAndCars)

% 只保留文件名及其所包含的 stop sign 区域
stopSigns = stopSignsAndCars(:, {'imageFilename','stopSign'});
% 显示一张照片及其所包含的真实 stop sign 区域
I = imread(stopSigns.imageFilename{1});
I = insertObjectAnnotation(I,'Rectangle',stopSigns.stopSign{1},'stop sign','
LineWidth',8);
figure
```

```
imshow(I)

% 设置训练策略
    options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 2, ...
        'Verbose', true);

% 训练网络
    rcnn = trainRCNNObjectDetector(stopSigns, net, options, ...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

%% 步骤 5: 检验检测器对 stop sign 图像的检测效果
% 载入测试图片
testImage = imread('stopSignTest.jpg');
% 检测 stop sign 标志
[bboxes,score,label] = detect(rcnn,testImage,'MiniBatchSize',128)

% 计算置信度并显示
[score, idx] = max(score);
bbox = bboxes(idx, :);
annotation = sprintf('%s: (Confidence = %f)', label(idx), score);
outputImage = insertObjectAnnotation(testImage, 'rectangle', bbox, annotation);
figure
imshow(outputImage)
**********************************************************************
```

图 5.4.5 例程 5.4.2 的运行效果

对比例程 5.4.1 与例程 5.4.2 的实现过程以及两个程序的运行效果可知，基于 AlexNet 迁移学习的 R-CNN 检测器实现更加简单，检测成功率更高。

5.4.9　基于 Image Labeler 的 R-CNN 目标检测器构建

在例程 5.4.1 和例程 5.4.2 训练 R-CNN 目标检测器的过程中，所采用的数据集都是 MATLAB 自带的，那么是否可以自己构建用于训练 R-CNN 目标检测器的数据集呢？MATLAB 中自带的 Image Labeler(见图 5.4.6)便是一个很好的工具。

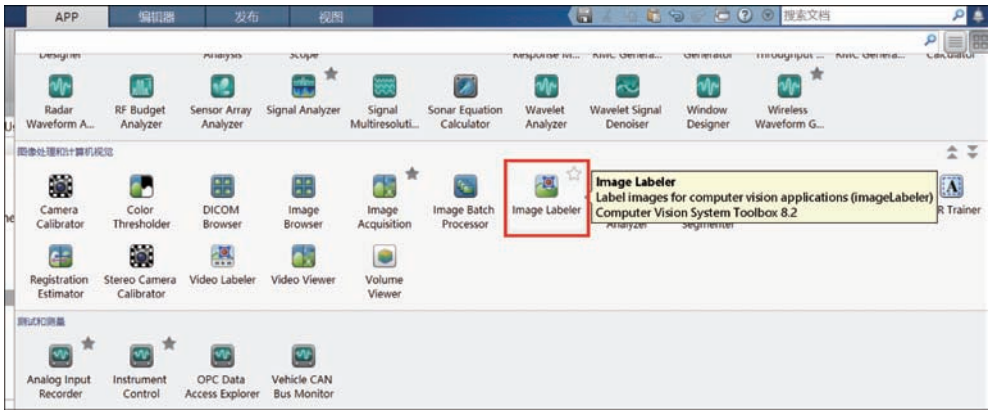

图 5.4.6　Image Labeler

将本书配套资料中的名为 stopsign 的文件夹复制到 C 盘的\Documents\MATLAB 的文件夹中(注：MATLAB 安装在不同的硬盘分区里，路径可能不同。此处的路径为 C:\Users\zhao\Documents\MATLAB\)。

单击 Image Labeler 图标，出现如图 5.4.7 所示的界面。

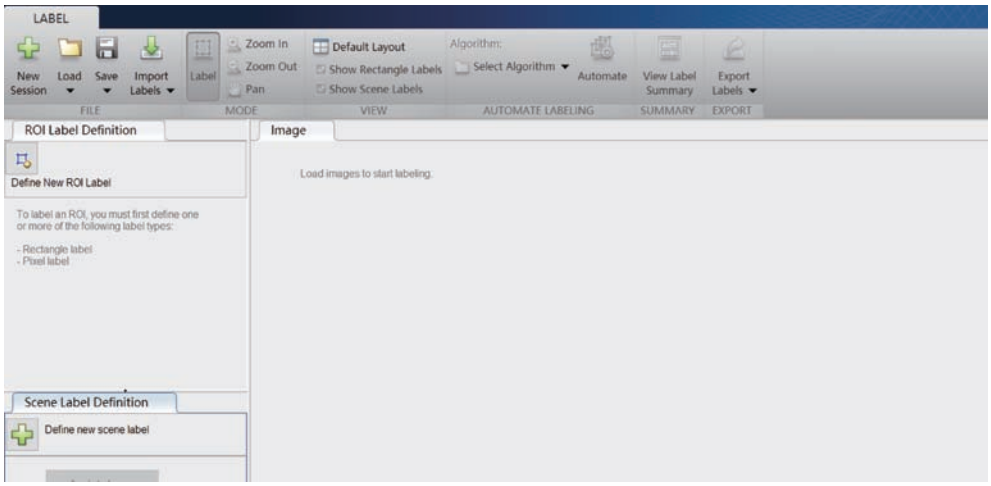

图 5.4.7　Image Labeler 界面

如图 5.4.8 所示,单击 Load 按钮,将 stopsign 文件夹下的所有图片导入。

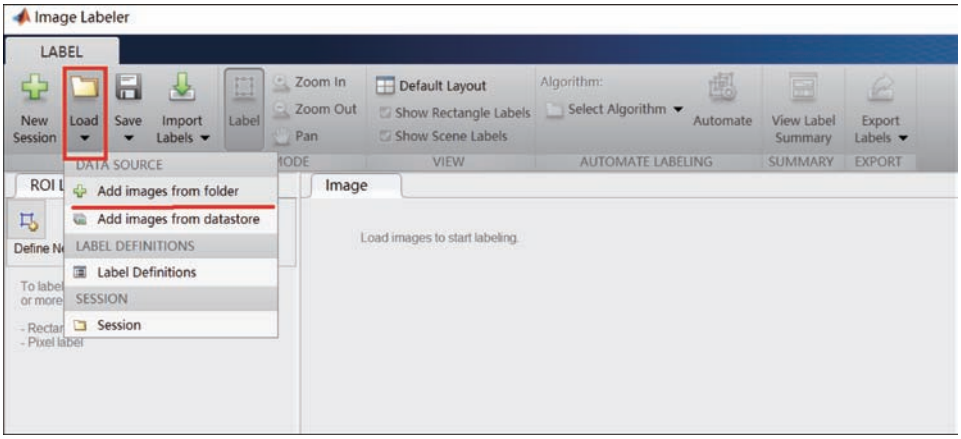

图 5.4.8 导入图片

在导入图片之后,采用 Define New ROILabel 工具(见图 5.4.9)对目标的类别(见图 5.4.10)和位置(见图 5.4.11)进行标注。

图 5.4.9 导入图像

图 5.4.10 设置分类标签

图 5.4.11　标注目标位置

依次完成对所有图像的位置和种类的标注，如图 5.4.12 所示。单击 ExportLabels，选择 ToWorkspace，生成 grouthTruth 格式的数据，如图 5.4.13 所示。

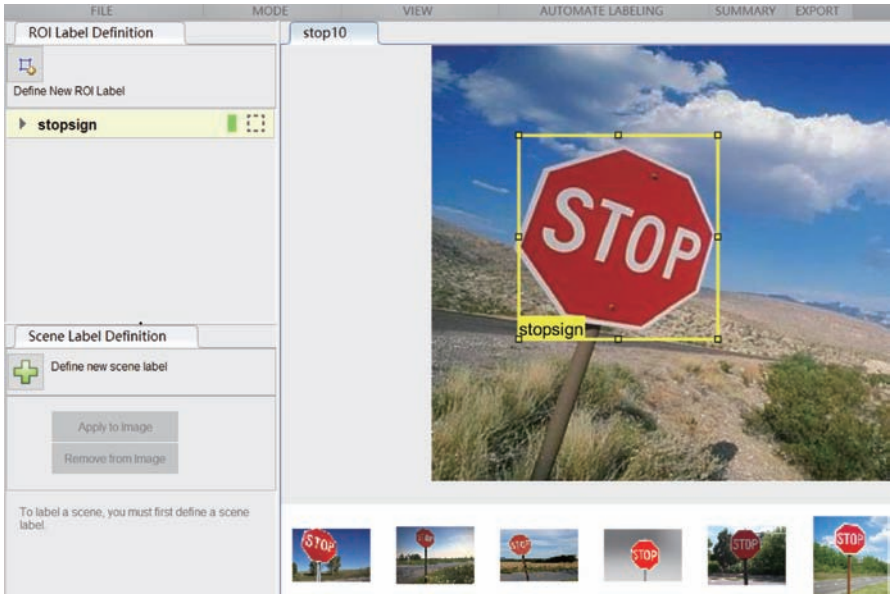

图 5.4.12　依次完成对所有图像的位置和种类的标注

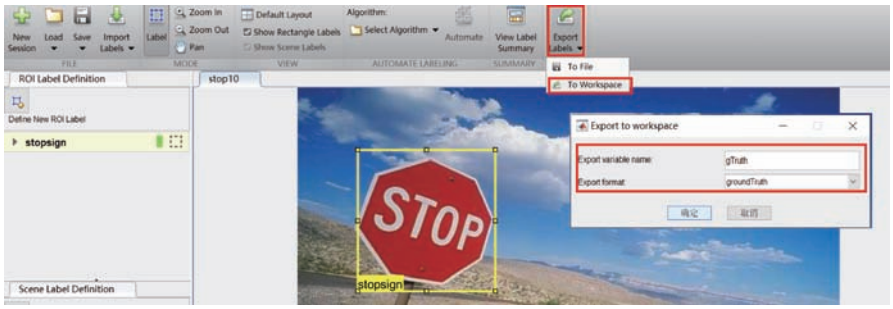

图 5.4.13　将标注的样本导出到工作空间

在完成上述操作后,通过例程 5.4.3 可实现训练 R-CNN 网络。在运行程序之前,请将本书配套资料中的名为 stoptest.jpg 的文件夹复制到 C 盘的\Documents\MATLAB 的文件夹之中(注:MATLAB 安装在不同的硬盘分区里,路径可能不同。此处的路径为 C:\Users\zhao\Documents\MATLAB\)。

注意:关于 AlexNet 的下载及安装方式,请参考 4.3.4 节,此处不再赘述。

例程 5.4.3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 例程 5.4.3
% 功能:基于 Image Labeler 输出数据的 R-CNN 目标检测器构建
% 作者:zhaoxch_mail@sina.com
% 时间:2020 年 4 月 19 日

%% 进行数据类型的转化
trainingdate = objectDetectorTrainingData(gTruth);
%% 导入网络
net = alexnet;
%% 设置训练策略参数并进行训练
% 设置训练策略参数
options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs',10, ...
        'Verbose', true);

% 训练网络.
    rcnn = trainRCNNObjectDetector(trainingdate, net, options, ...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

%% 显示测试结果
% 读取数据
I = imread('stoptest.jpg');
% 用检测器测试
[bboxes,scores] = detect(rcnn,I);
% 标注测试结果并显示
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

在例程 5.4.3 中，通过 objectDetectorTrainingData 函数将 grouthTruth 格式的数据转换成为可以用于训练的数据。例程 5.4.3 的运行效果如图 5.4.14 所示。

图 5.4.14　例程 5.4.3 的运行效果

5.5　基于 Video Labeler 与 R-CNN 的车辆检测

在 5.4 节讲解了目标检测的概念及 R-CNN 目标检测算法的原理和实现过程，在此基础上，本节重点介绍如何基于 Video Labeler 与 R-CNN 的车辆检测。本节采用步骤指引式的讲解方式，读者可以按照文中的步骤进行操作，在操作中学习、体会。

5.5.1　实例需求

例 5.5.1　基于 Video Labeler 构建设计一个 R-CNN 目标检测深度网络，用于检测图像中的汽车目标，如图 5.5.1 所示。

图 5.5.1　目标图像

5.5.2　实现步骤

步骤 1，将本书配套资料中的名为 drivingdata.mp4 的视频和名为 cars.png 的图像复制到 C 盘的\Documents\MATLAB 文件夹中(注：MATLAB 安装在不同的硬盘分区里，路径可能不同。此处的路径为 C:\Users\zhao\Documents\MATLAB\)。

步骤 2，如图 5.5.2 所示，单击 Video Labeler 图标，出现如图 5.5.3 所示的界面。

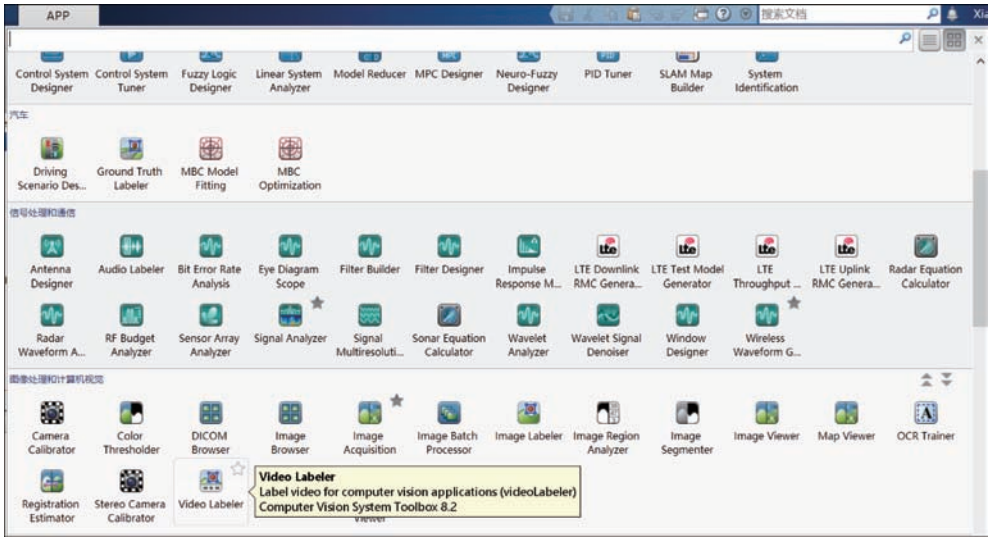

图 5.5.2 单击 Video Labeler 图标

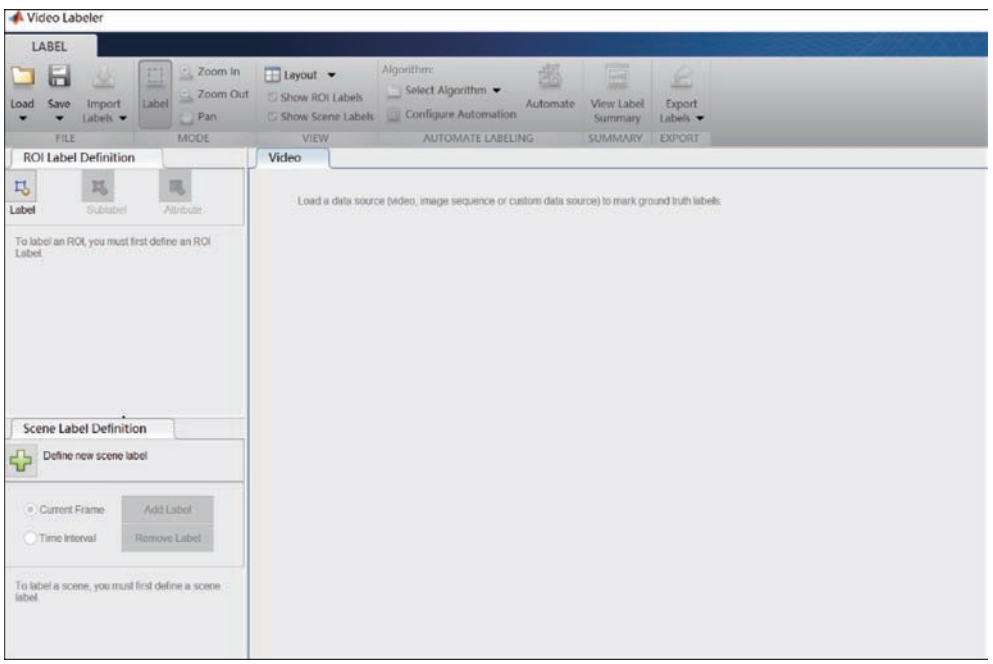

图 5.5.3 Video Labeler 界面

步骤 3,如图 5.5.4 所示,单击 Load 按钮,导入 drivingdata.mp4 视频。导入待标注视频后,会出现如图 5.5.5 所示的界面。

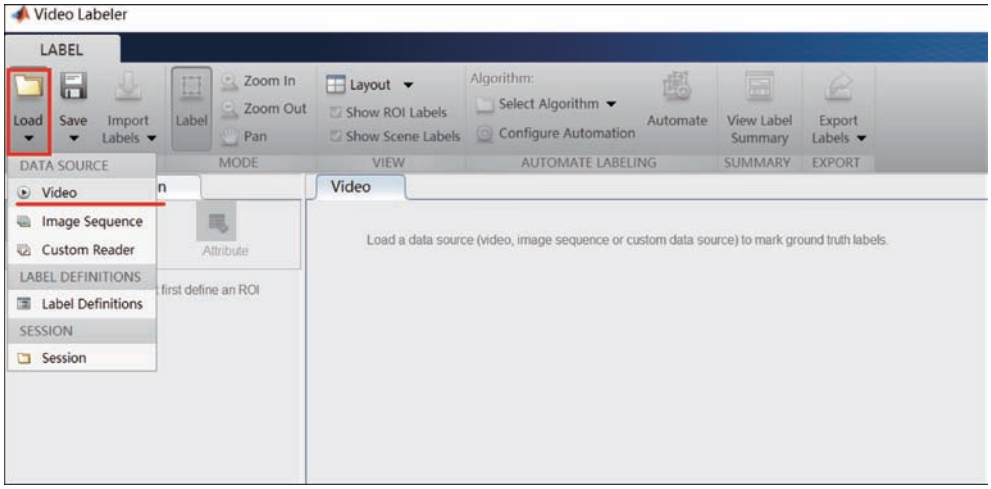

图 5.5.4　导入视频

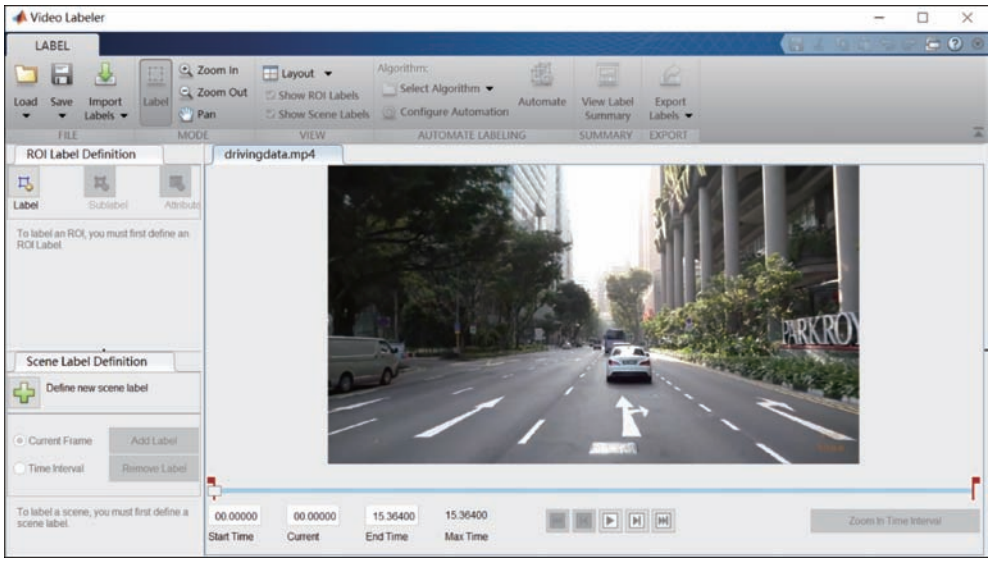

图 5.5.5　导入视频后的界面

步骤 4，单击 Label 按钮设定标注的类别（见图 5.5.6），在 Label Name 栏内输入 car，可以选定标注的形状。

步骤 5，自动标注需要选择跟踪算法，可供选择的有 ACF 车辆检测算法、ACF 行人检测算法以及 KLT 光流跟踪算法，在本实例中选择 ACF 车辆检测算法，如图 5.5.7 所示。

步骤 6，算法选择好后，设定好时间轴上的标定时间范围，单击工具栏中的 Automate 按钮进入自动标注选定界面（见图 5.5.8），标注的过程如图 5.5.9 所示。

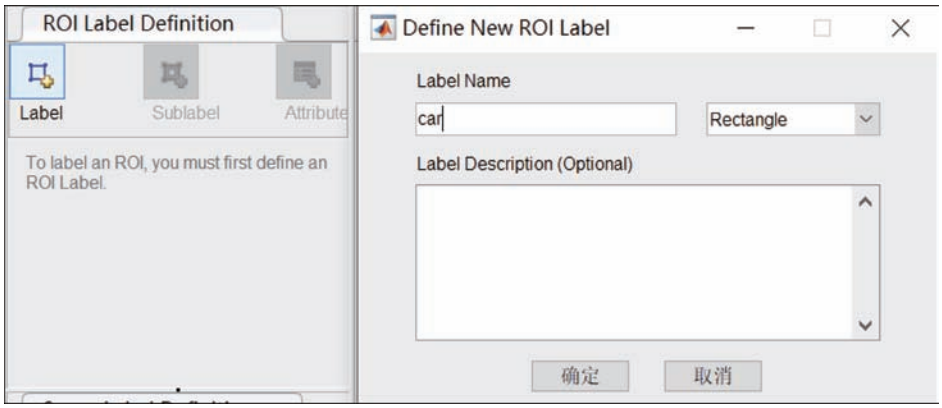

图 5.5.6 设定标注的类别

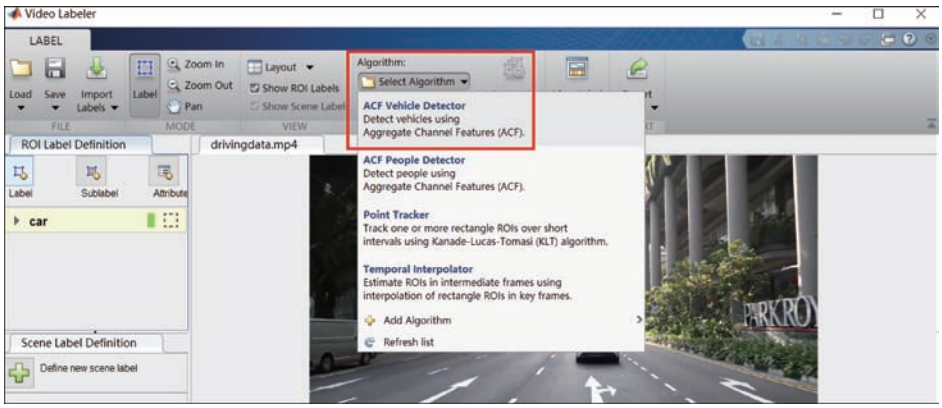

图 5.5.7 选择目标跟踪算法

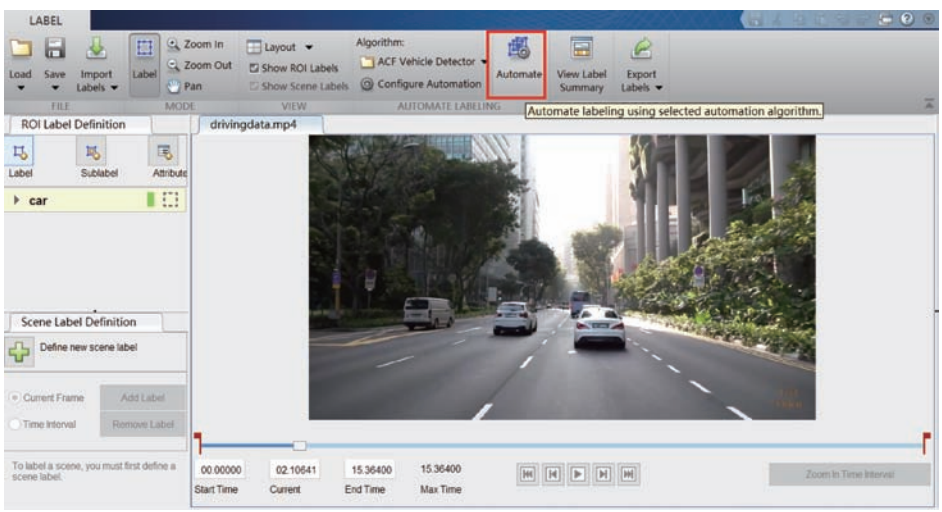

图 5.5.8 进入自动标注选定界面

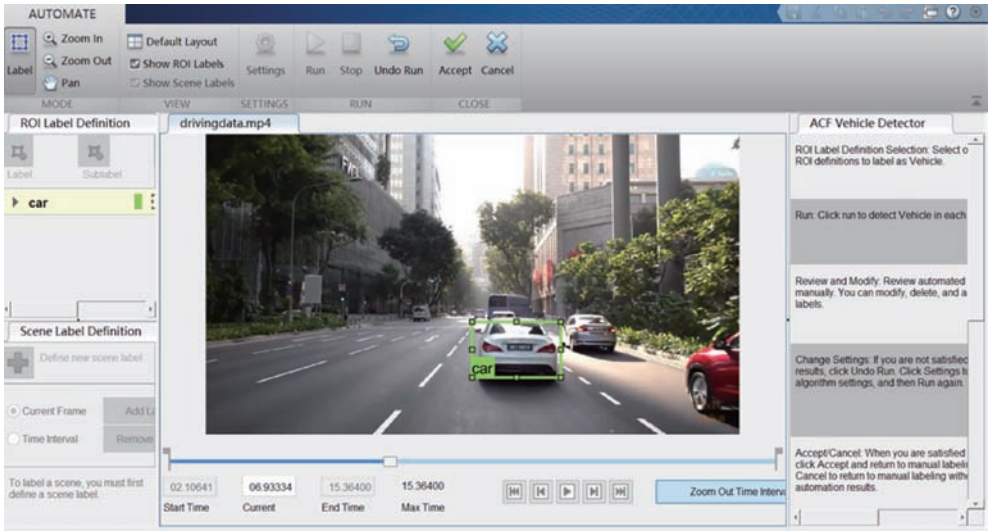

图 5.5.9　自动标注的过程

自动标注完成后可以单击时间轴中的播放按钮查看标注效果。如果效果良好，则单击工具栏中的 Accept 按钮完成自动标注。

步骤 7，当自动标注完成后，单击工具栏中的 Export Labels 按钮（见图 5.5.10），生成 grouthTruth 格式的数据。

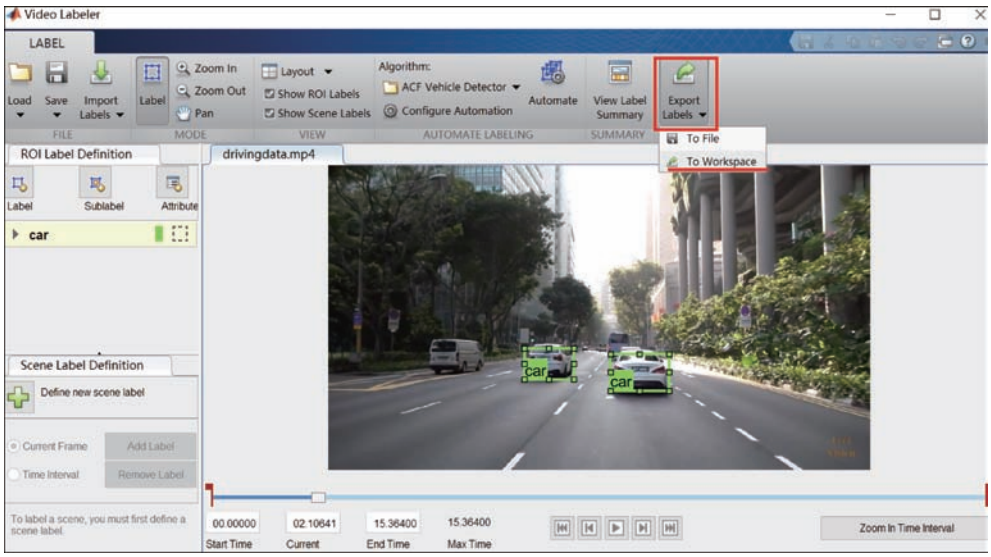

图 5.5.10　导出标注数据

groundTruth 格式包括视频文件路径、标注定义以及一个包括每帧标注点坐标，如图 5.5.11 所示。

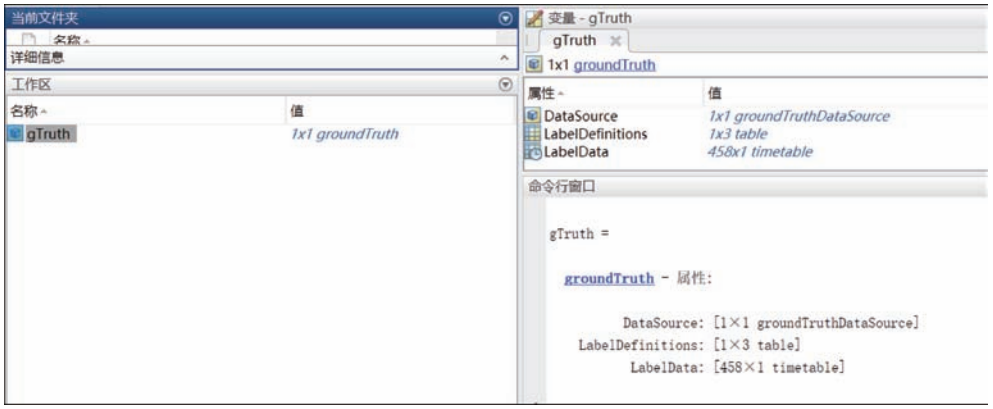

图 5.5.11 groundTruth 格式的数据

步骤 8,运行例程 5.5.1,其运行效果如图 5.5.12 所示。

例程 5.5.1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 程序说明
% 实例 5.5.1
% 功能: 基于 Video Labeler 输出数据的 R-CNN 目标检测器构建
% 作者: zhaoxch_mail@sina.com
% 时间: 2020 年 4 月 19 日

%% 进行数据类型的转化
trainingdate = objectDetectorTrainingData(gTruth);
%% 导入网络
net = alexnet;

%% 设置训练策略参数并进行训练
% 设置训练策略参数
options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 2, ...
        'Verbose', true);

% 训练网络
    rcnn = trainRCNNObjectDetector(trainingdate, net, options, ...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

%% 显示测试结果
% 读取数据
```

```
I = imread('cars.png');
% 用检测器测试
[bboxes,scores] = detect(rcnn,I);
% 标注测试结果并显示
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)
**********************************************************************
```

图 5.5.12　例程 5.5.1 的运行效果

在例程 5.5.1 中，通过 objectDetectorTrainingData 函数将 grouthTruth 格式的数据转换成为可以用于训练的数据。

5.6　思考与练习

1. 什么是图像分类？图像分类面临的难点有哪些？图像分类有哪些评价指标？

2. 传统的基于特征的图像分类与基于深度学习的图像分类的主要区别是什么？

3. 通过 MATLAB 编程实现采用 AlexNet 对自己制作的图像数据集进行分类。

4. 通过 MATLAB 编程实现采用 LeNet 对手写数字图片的分类。

5. 目标分类、目标检测、目标分割的区别是什么？

6. 评价目标检测性能的指标有哪些？

7. 请简述 R-CNN 的实现步骤。

8. 用 Image Labeler 构建目标检测所用的数据集并将其到如 MATLAB 工作空间中，转化成为可以用于训练的数据。

9. 请在 MATLAB 中编写人行横道 R-CNN 的检测程序。

10. 下列程序的功能是什么？请为下列程序添加注释。

```
% 功能: _____

% _____
net = alexnet;

% 载入训练集
data = load('stopSignsAndCars.mat', 'stopSignsAndCars');
stopSignsAndCars = data.stopSignsAndCars;
% 设置图像路径参数
visiondata = fullfile(toolboxdir('vision'),'visiondata');
stopSignsAndCars.imageFilename = fullfile(visiondata, stopSignsAndCars.imageFilename);
% 显示数据
summary(stopSignsAndCars)

% 只保留文件名及其所包含的 stop sign 区域
stopSigns = stopSignsAndCars(:, {'imageFilename','stopSign'});
% _____
    options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 0.0001, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 10, ...
        'Verbose', true);

% _____
    rcnn = trainRCNNObjectDetector(stopSigns, net, options, ...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

% 载入测试图片
testImage = imread('stopSignTest.jpg');
% _____
[bboxes,score,label] = detect(rcnn,testImage,'MiniBatchSize',128)
```