计算机科学与技术学院

毕业设计

论文题目	基于 SSD 网络模型的房屋瓦片损害检测		
学校导师	刘立	职称	教授
企业导师	刘立	职称	教授
学生姓名	李开运	学号	20144330106
专业班级	物联网	班级	14 级 01 班
系主任	毛宇	院长	刘振宇
起止时间	2017 年 6 月 5 日至 2018 年 5 月 22 日		

2018 年 3 月 8 日

目录

基于 SSD 网络模型的房屋瓦片损害检测

摘　要：　这也是一个摘要

关键词： 人工智能，机器视觉

第一章　前言

目标检测一直是计算机视觉的基础问题，在 2010 年左右就开始停滞不前了。自 2013 年一篇论文的发表，目标检测从原始的传统手工提取特征方法变成了基于卷积神经网络的特征提取，从此一发不可收拾。根着历史的潮流，简要地探讨“目标检测”算法的两种思想和这些思想引申出的算法，主要涉及那些主流算法。

1.1　概述

在深度学习正式介入之前，传统的“目标检测”方法都是区域选择、提取特征、分类回归三部曲，这样就有两个难以解决的问题；其一是区域选择的策略效果差、时间复杂度高；其二是手工提取的特征鲁棒性较差。云计算时代来临后，“目标检测”算法大家族主要划分为两大派系，一个是 R-CNN 系两刀流，另一个则是以 YOLO 为代表的一刀流派。下面分别解释一下“两刀流”和“一刀流”。

两刀流: 顾名思义，两刀解决问题：

1、生成可能区域（Region Proposal）& CNN 提取特征

2、放入分类器分类并修正位置

这一流派的算法都离不开 Region Proposal ，即是优点也是缺点，主要代表人物就是 R-CNN 系。

一刀流: 顾名思义，一刀解决问题，直接对预测的目标物体进行回归。回归解决问题简单快速，但是太粗暴了，主要代表人物是 YOLO 和 SSD 。

无论“两刀流”还是“一刀流”，他们都是在同一个天平下选取一个平衡点、或者选取一个极端——要么准，要么快。两刀流的天平主要倾向准，一刀流的天平主要倾向快。但最后万剑归宗，大家也找到了自己的平衡，平衡点的有略微的不同。接下来我们花开两朵各表一支，一朵两刀流的前世今生，另一朵一刀流的发展历史。

1.2　R-CNN 系列

R-CNN 其实是一个很大的家族，自从 rbg 大神发表那篇论文，子孙无数、桃李满天下。在此，我们只探讨 R-CNN 直系亲属，他们的发展顺序如下：

他们在整个家族进化的过程中，一致暗埋了一条主线：充分榨干 feature maps 的价值。

1.2.1　RCNN

这个模型，是利用卷积神经网络来做「目标检测」的开山之作，其意义深远不言而喻。

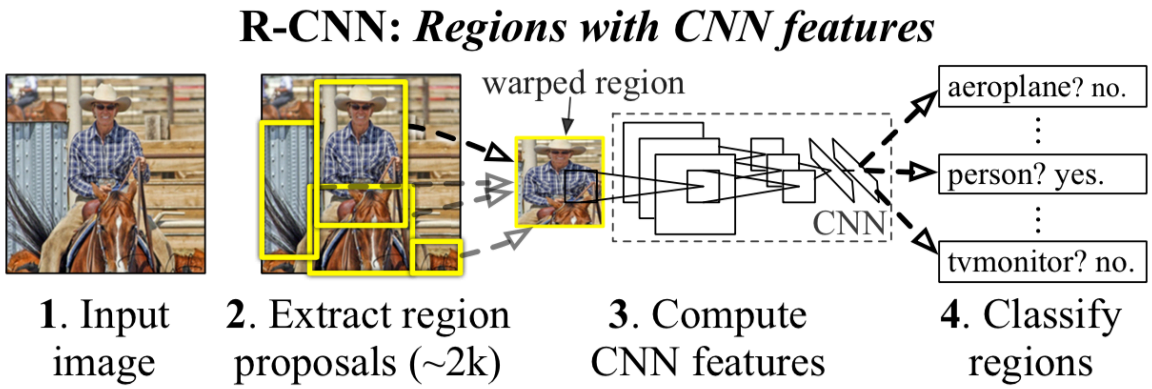

图 1.1: RCNN

解决问题一、速度传统的区域选择使用滑窗，每滑一个窗口检测一次，相邻窗口信息重叠高，检测速度慢。R-CNN 使用一个启发式方法（Selective search），先生成候选区域再检测，降低信息冗余程度，从而提高检测速度。

解决问题二、特征提取传统的手工提取特征鲁棒性差，限于如颜色、纹理等低层次（Low level）的特征。使用 CNN （卷积神经网络）提取特征，可以提取更高层面的抽象特征，从而提高特征的鲁棒性。

该方法将 PASCAL VOC 上的检测率从 35.1% 提升到 53.7 % ，提高了好几个量级。虽然比传统方法好很多，但是从现在的眼光看，只能是初窥门径。

1.2.2 SPP Net

R-CNN 提出后的一年，以何恺明、任少卿为首的团队提出了 SPP Net ，这才是真正摸到了卷积神经网络的脉络。也不奇怪，毕竟这些人鼓捣出了 ResNet 残差网络，对神经网络的理解是其他人没法比的。尽管 R-CNN 效果不错，但是他还有两个硬伤：

硬伤一、算力冗余先生成候选区域，再对区域进行卷积，这里有两个问题：其一是候选区域会有一定程度的重叠，对相同区域进行重复卷积；其二是每个区域进行新的卷积需要新的存储空间。何恺明等人意识到这个可以优化，于是把先生成候选区域再卷积，变成了先卷积后生成区域。“简单地”改变顺序，不仅减少存储量而且加快了训练速度。

硬伤二、图片缩放无论是剪裁（Crop）还是缩放（Warp），在很大程度上会丢失图片原有的信息导致训练效果不好，如上图所示。直观的理解，把车剪裁成一个门，人看到这个门也不好判断整体是一辆车；把一座高塔缩放成一个胖胖的塔，人看到也没很大把握直接下结论。人都做不到，机器的难度就可想而知了。

何恺明等人发现了这个问题，于是思索有什么办法能不对图片进行变形，将图片原汁原味地输入进去学习。最后，他们发现问题的根源是 FC Layer（全连接层）需要确定输入维度，于是他们在输入全连接层前定义一个特殊的池化层，将输入的任

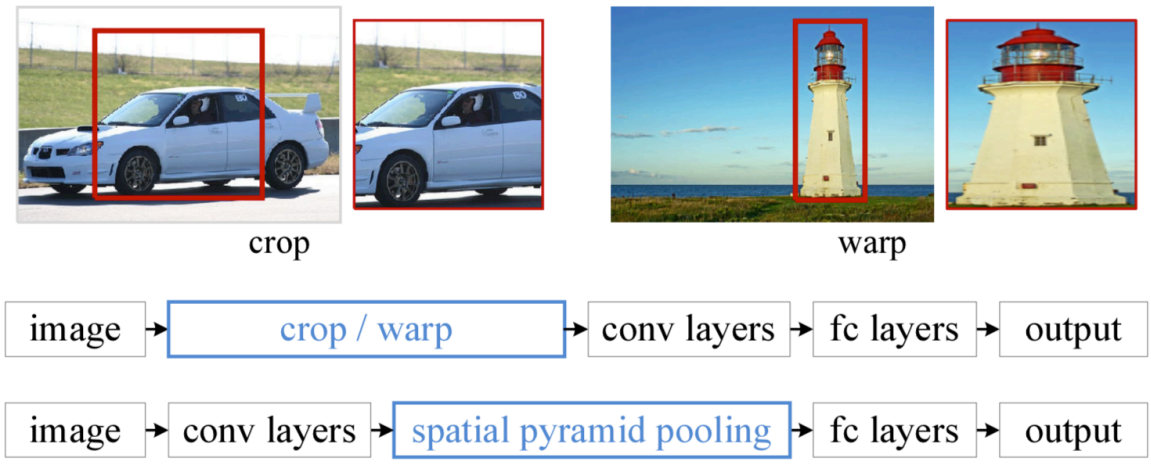

Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

图 1.2: RCNN

意尺度 feature maps 组合成特定维度的输出，这个组合可以是不同大小的拼凑，如同拼凑七巧板般。举个例子，我们要输入的维度 64*256 ，那么我们可以这样组合 32*256+16*256+8*256+8*256 。

SPP Net 的出现是如同一道惊雷，不仅减少了计算冗余，更重要的是打破了固定尺寸输入这一束缚，让后来者享受到这一缕阳光。

1.2.3 Fast R-CNN

在这篇论文中，引用了 SPP Net 的工作，并且致谢其第一作者何恺明的慷慨解答。纵观全文，最大的建树就是将原来的串行结构改成并行结构

原来的 R-CNN 是先对候选框区域进行分类，判断有没有物体，如果有则对 Bounding Box 进行精修回归。这是一个串联式的任务，那么势必没有并联的快，所以 rbg 就将原有结构改成并行——在分类的同时，对 Bbox 进行回归。这一改变将 Bbox 和 Clf 的 loss 结合起来变成一个 Loss 一起训练，并吸纳了 SPP Net 的优点，最终不仅加快了预测的速度，而且提高了精度。

1.2.4 Faster R-CNN

在 Faster R-CNN 前，我们生产候选区域都是用的一系列启发式算法，基于 Low Level 特征生成区域。这样就有两个问题：

第一个问题是生成区域的靠谱程度随缘，而两刀流算法正是依靠生成区域的靠谱程度——生成大量无效区域则会造成算力的浪费、少生成区域则会漏检；

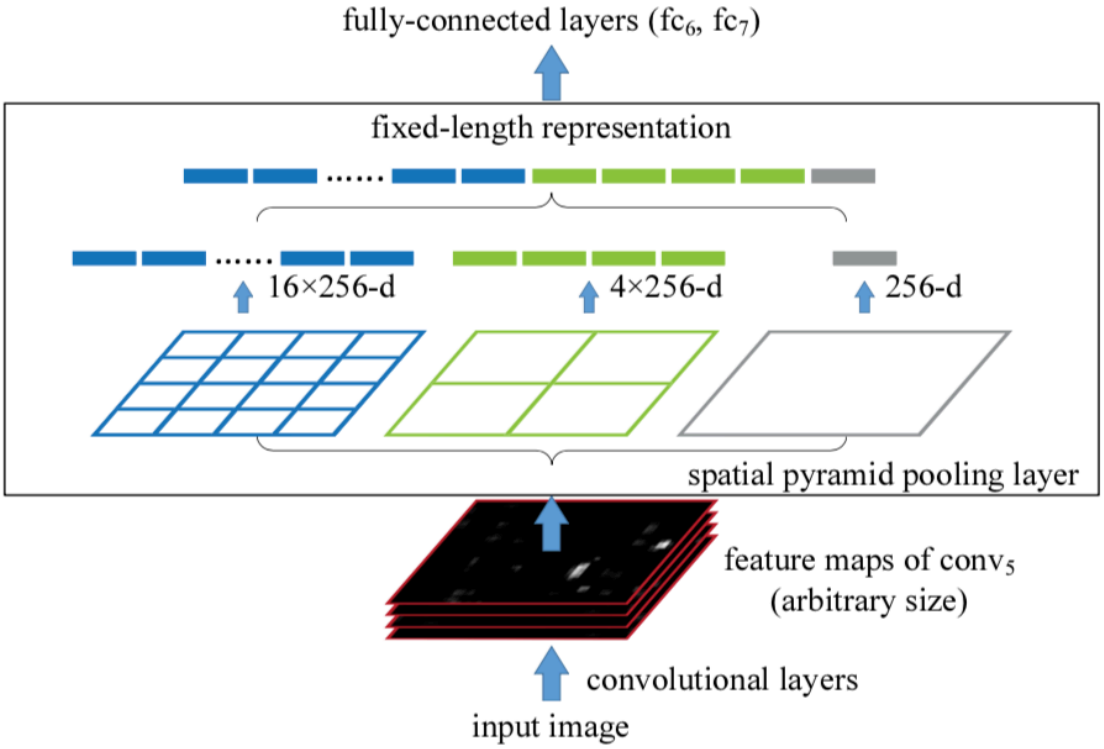

Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

图 1.3: RCNN

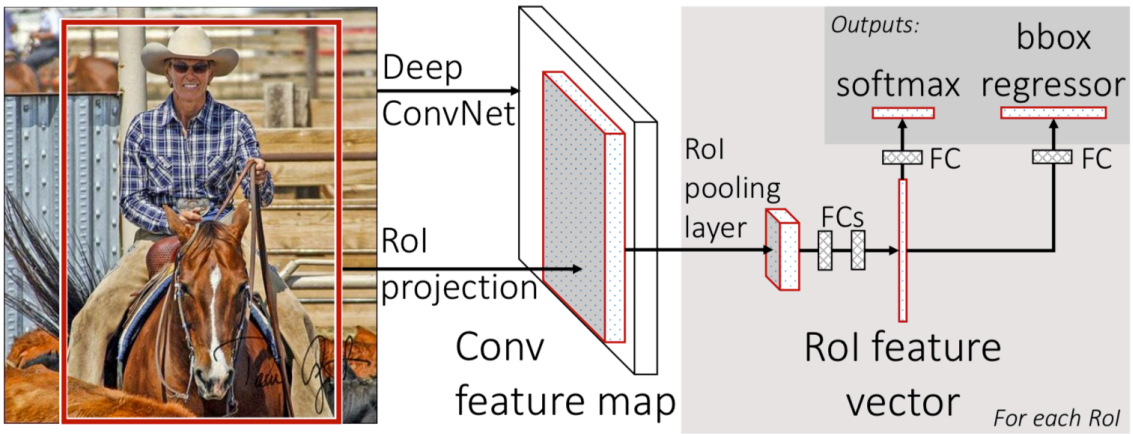

图 1.4: RCNN

第二个问题是生成候选区域的算法是在 CPU 上运行的，而我们的训练在 GPU 上面，跨结构交互必定会有损效率。

那么怎么解决这两个问题呢？于是乎，任少卿等人提出了一个 Region Proposal Networks 的概念，利用神经网络自己学习去生成候选区域。

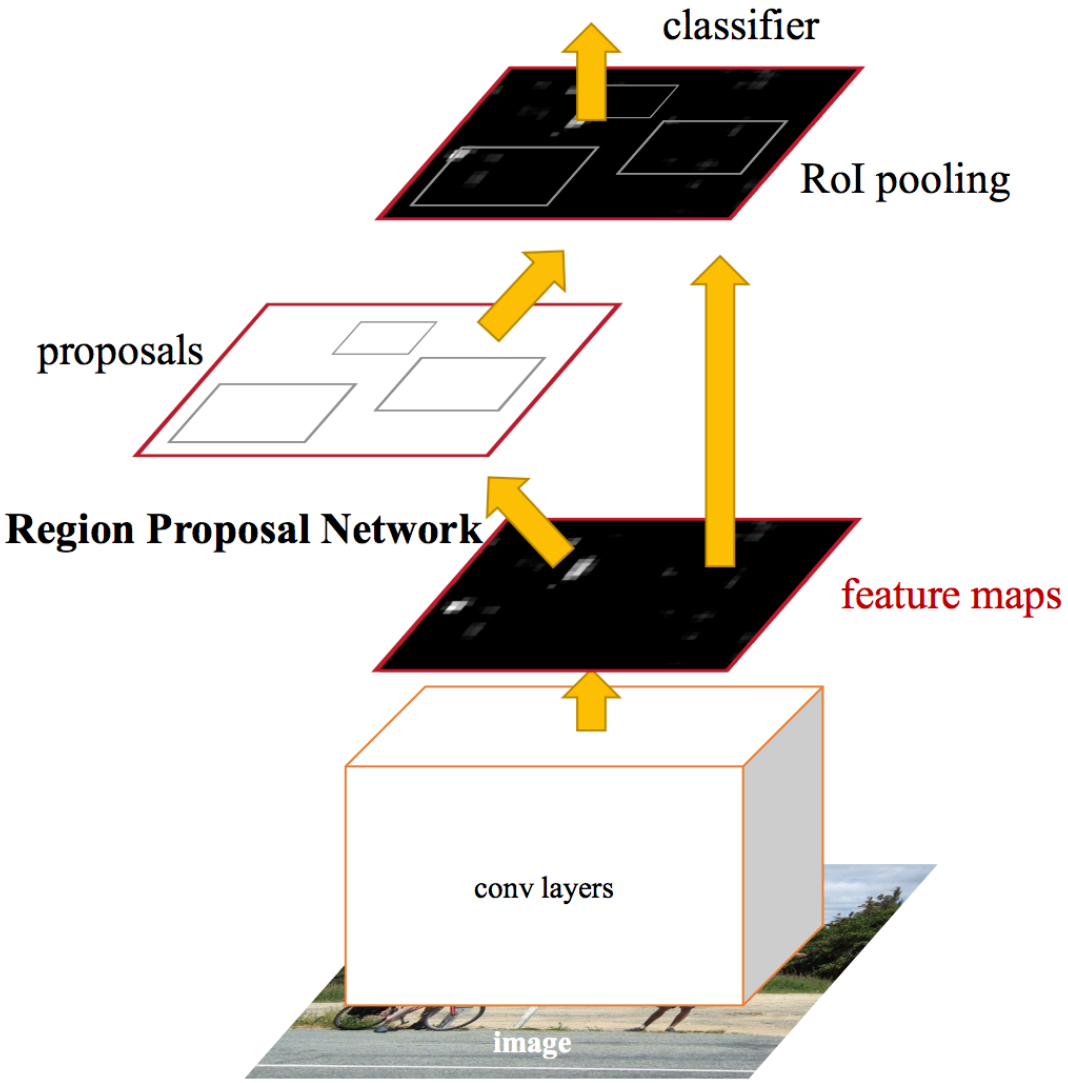

Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

图 1.5: RCNN

这种生成方法同时解决了上述的两个问题，神经网络可以学到更加高层、语义、抽象的特征，生成的候选区域的可靠程度大大提高；可以从上图看出 RPNs 和 RoI Pooling 共用前面的卷积神经网络——将 RPNs 嵌入原有网络，原有网络和 RPNs 一

起预测，大大地减少了参数量和预测时间。

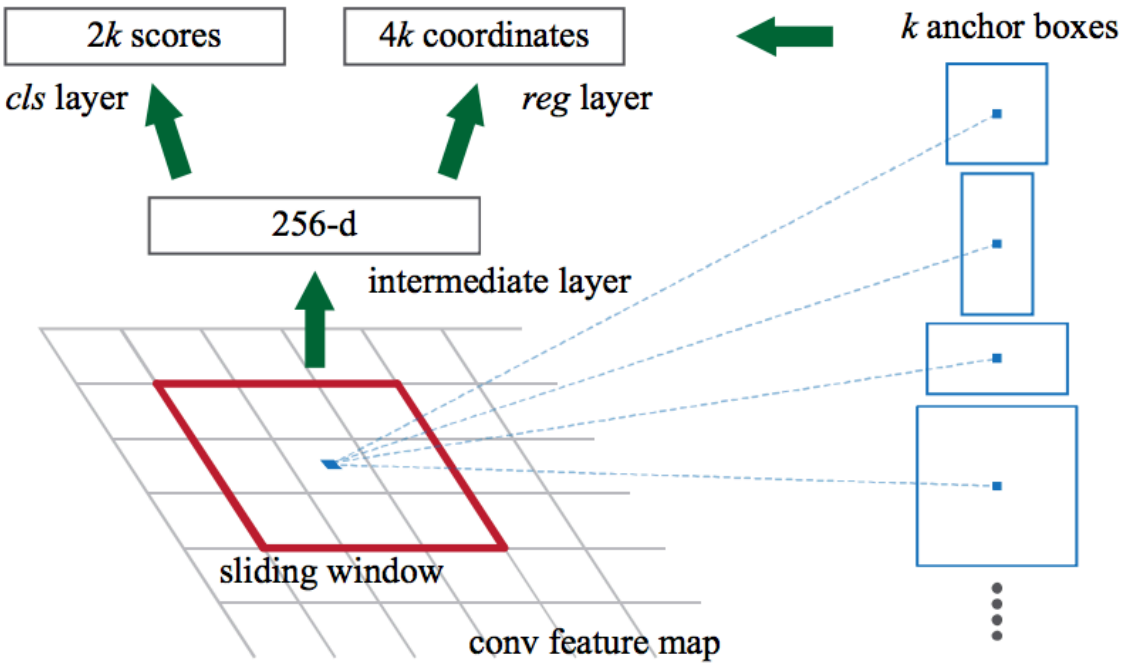

图 1.6: RCNN

在 RPNs 中引入了 anchor 的概念，feature map 中每个滑窗位置都会生成 k 个 anchors，然后判断 anchor 覆盖的图像是前景还是背景，同时回归 Bbox 的精细位置，预测的 Bbox 更加精确。

1.2.5 Mask R-CNN

时隔一年，何恺明团队再次更新了 R-CNN 家族，改进 Faster R-CNN 并使用新的 backbone 和 FPN 创造出了 Mask R-CNN 。

加一条通道

我们纵观发展历史，发现 SPP Net 升级为 Fast R-CNN 时结合了两个 loss ，也就是说网络输入了两种信息去训练，结果精度大大提高了。何恺明他们就思考着再加一个信息输入，即图像的 Mask ，信息变多之后会不会有提升呢？于是乎 Mask R-CNN 就这样出来了，不仅可以做「目标检测」还可以同时做「语义分割」，将两个计算机视觉基本任务融入一个框架。没有使用什么 trick ，性能却有了较为明显的提升，这个升级的版本让人们不无啧啧惊叹。作者称其为 meta algorithm ，即一个基础的算法，只要需要「目标检测」或者「语义分割」都可以使用这个作为 Backbone 。

1.3 YOLO 系列

1.3.1 YOLO

1.3.2 SSD

1.3.3 YOLO9000

1.4 小结

第二章　为什么要使用 SSD

第三章　如何使用 SSD

第四章　实验过程与结果

第五章　总结

第六章　致谢

参考文献