



**模式识别与机器学习**

课程设计报告

目录

[第一章 课题概述 1](#_Toc170680737)

[第二章 算法分析 2](#_Toc170680738)

[2.1 数据集及其预处理 2](#_Toc170680739)

[2.2 目标检测算法 4](#_Toc170680740)

[2.3 对抗生成算法 10](#_Toc170680741)

[第三章 对抗 15](#_Toc170680742)

[3.1 RPAttack 15](#_Toc170680743)

[3.2 DPAttack 17](#_Toc170680744)

[3.3 XXX 20](#_Toc170680745)

[3.4 对比分析 20](#_Toc170680746)

[第四章 结束语 20](#_Toc170680747)

# 课题概述

# 算法分析

### 数据集及其预处理

#### 2.1.1 DOTA数据集

DOTA是一种用于航拍图像中目标检测的大规模数据集，它可以用于开发和评估航空图像中的目标探测器。这些图像是从不同的传感器和平台收集的。每张图像的大小范围从800 × 800到20000 × 20000像素，并包含显示各种比例、方向和形状的对象。DOTA图像中的实例由航空图像解译专家用任意四边形（8个自由度）进行注释（如图所示）。目前它有三个版本：

**DOTA-v1.0**包含15个常见类别、2806张图像和188282个实例。在DOTA-v1.0中，训练集、验证集和测试集的比例分别为1/2、1/6和1/3。

**DOTA-v1.5**使用与DOTA-v1.0相同的图像，但也对极小的实例(小于10像素)进行了注释。此外，还增加了一个新的类别“集装箱起重机”。它总共包含403318个实例，图像和数据集分割的数量与DOTA-v1.0相同。

**DOTA-v2.0**收集了更多的谷歌地球、GF-2卫星和航空图像。在DOTA-v2.0中有18个常见类别、11268张图像和1793658个实例。与DOTA-v1.5相比，它进一步增加了“机场”和“直升机停机坪”的新类别。DOTA的11268张图像分为训练集、验证集、测试开发集和测试挑战集。为了避免过拟合问题，训练集和验证集的比例小于测试集。此外，它有两个测试集，即测试开发和测试挑战。训练包含1830张图像和268627个实例；验证包含593个图像和81048个实例。



图 2-1 DOTA数据集示例

考虑到算力与存储容量的限制，我们选取DOTA-v1.5进行训练与测试。DOTA-v1.5中目标类别包括飞机、船舶、储罐、棒球场、网球场、篮球场、田径场、港口、桥梁、大型车辆、小型车辆、直升机、环岛、足球场、游泳池以及集装箱起重机。

图像当中每一个目标都被一个定向边界框（Oriented Bounding Box，OBB）所标注，OBB表示为，其中表示第个顶点，顶点按顺时针顺序排列。除了OBB，每一个实例还标注了类别和检测的困难程度（1表示困难，0表示不困难）。图像的注释保存在具有相同文件名的“.txt”文本文件中，每一行代表一个实例。数据集的文件夹结构如图2-2所示。

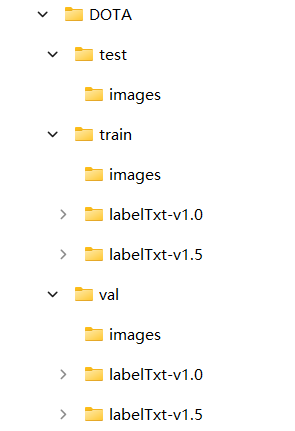


图 2-2 DOTA数据集文件夹结构

#### 2.1.2 数据预处理

为了让数据集满足相应检测网络的大小与格式，我们需要对数据集进行预处理。

由于DOTA数据集中有的图片纵横比太大，无法直接用于后续的训练，因此需要对数据集进行切割成600 × 600固定大小的图片并对切割后的图片生成对应标注信息的“.xml”文件，并按照VOC2007的格式存放数据集，文件夹结构如图2-3所示。

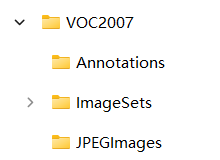


图 2-3 VOC2007数据集结构

同样的方法，也可以将标注信息保存为“.json”，把数据集存放成如图2-4所示的coco格式。

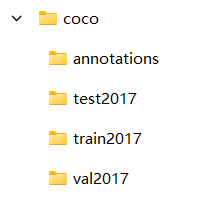


图 2-4 coco数据集结构

经过上述的数据处理，可以使数据集的格式满足一定的标准，有利于模型的训练。

### 目标检测算法

* + 1. Faster R-CNN

目标检测作为计算机视觉领域的一个核心任务，对于理解图像内容和实现高级视觉系统至关重要。近年来，随着深度学习技术的兴起，目标检测技术取得了显著进展。然而，早期的对象检测方法，如R-CNN及其变体Fast R-CNN，虽然在准确性上取得了突破，但仍受限于计算效率和处理速度，难以满足实时或近实时应用的需求。这些方法依赖于耗时的区域提议步骤来生成候选对象框，这成为了整个检测流程的瓶颈。

为了克服这一挑战，Faster R-CNN模型提出了一种创新的区域候选网络（Region Proposal Network, RPN），它与检测网络共享卷积特征，从而显著提高了检测速度，同时保持了高精度的检测结果。Faster R-CNN的提出，不仅推动了对象检测技术的发展，也为实时视频分析、自动驾驶车辆的视觉感知等应用领域提供了强有力的技术支持。

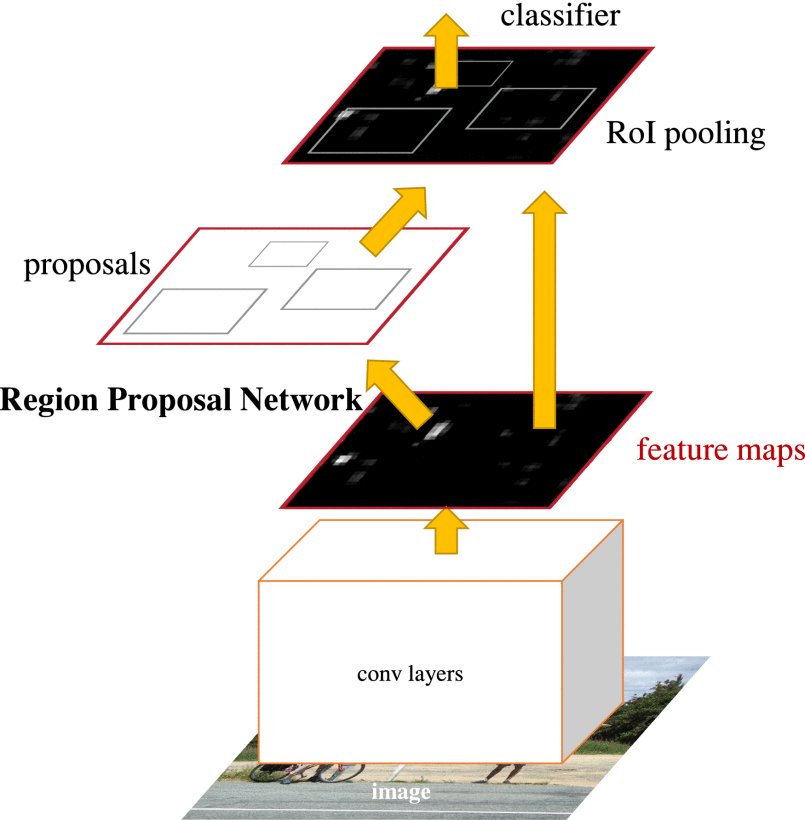


图 2-5 Faster R-CNN架构

Faster R-CNN的总体架构如图2-5所示。其检测部分主要包括四个模块：

1. Conv layers，即特征提取网络，用于提取特征。如图2-6所示，通过一系列的卷积、非线性激活以及池化操作来提取图像的特征图，用于后续的RPN网络和选取候选区域。

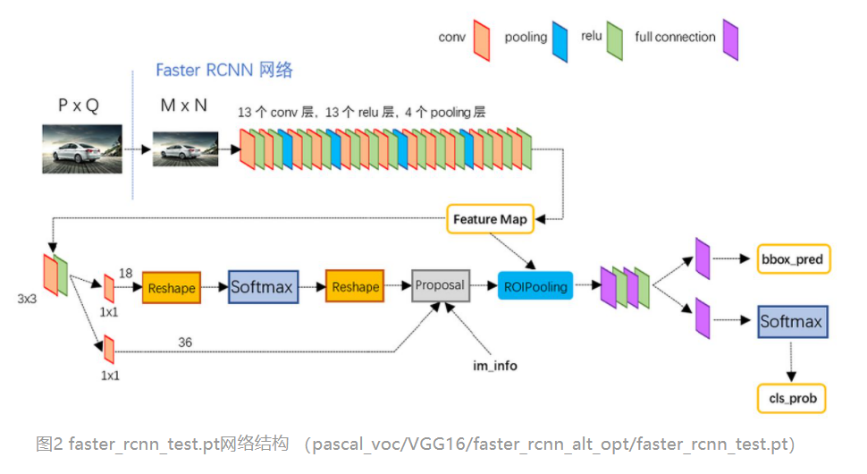


图 2-6 Faster R-CNN工作原理

1. RPN，即区域候选网络，用于生成候选框。该模块分成两个部分，一是分类，在获取卷积网络提取的特征图之后，利用softmax判断所有预设的anchor属于正类还是负类（即anchor内是否有目标，属于二分类问题）；二是边界框回归，计算anchor的边界框偏移量，

式中是anchor中心点坐标以及宽和高，是回归层预测的偏移量，由此计算出修正后的anchor坐标，正得到较为准确的候选区域，计算如下，

最后生成最终的候选区域。

1. RoI pooling，即兴趣域池化，用于收集RPN生成的候选区域。对于大小各不相同的区域，通过预设pooled\_w和pooled\_h将各个区域的特征大小统一，对每一部分进行最大池化，实现固定长度输出。
2. Classification and Regression，即分类与回归。RPN中的分类只是判断对象是目标还是背景，即二分类，而在该部分中，需要对相应的区域进行具体的分类，通常为多分类。从RoI池化处理之后获取到的pooled\_w × pooled\_h的特征图送入网络中，通过全连接层和softmax进行分类，再对候选区域进行边界框回归，获得更高精度的最终边界框。

RPN网络在训练过程中定义了两个损失，其中表示softmax loss，用于分类anchors属于前景还是背景，表示smooth L1 loss，用于修正anchor框，前面乘了一个表示只回归有目标的框，

式中，表示网络预测出来的第个anchor是目标的概率；为对应的ground truth，若，则，若，则，其余情况anchor不参与训练；表示参数化坐标向量，表示预测框和anchor框之间的偏移；为对应的ground truth；为mini-batch的大小；为anchor位置的数量；用于平衡两个损失。

Faster R-CNN的提出显著推动了目标检测领域的发展，它不仅提高了检测速度，还通过引入区域提议网络简化了整个检测流程，减少了对外部区域提议算法的依赖。

* + 1. **YOLOv4**

YOLOv4网络是YOLO系列目标检测网络的第四代。其继承了YOLO系列网络的传统：在保持相对较高的mAP的同时，大大降低计算量。虽然其没有像前几代一样提出很多创新性的东西，但是其综合了诸多那时最新的技术成果并且进行了很多尝试和对比实验，提出了一种新的网络结构。

YOLOv4 网络的结构可分为四部分：输入端、主干网络(Backbone)-主干特征提取网络、颈部网络(Neck)-加强特征提取网络和头部网络（Head）--用来预测（Prediction）。图2-7为 YOLOv4 算法的网络框架示意图。

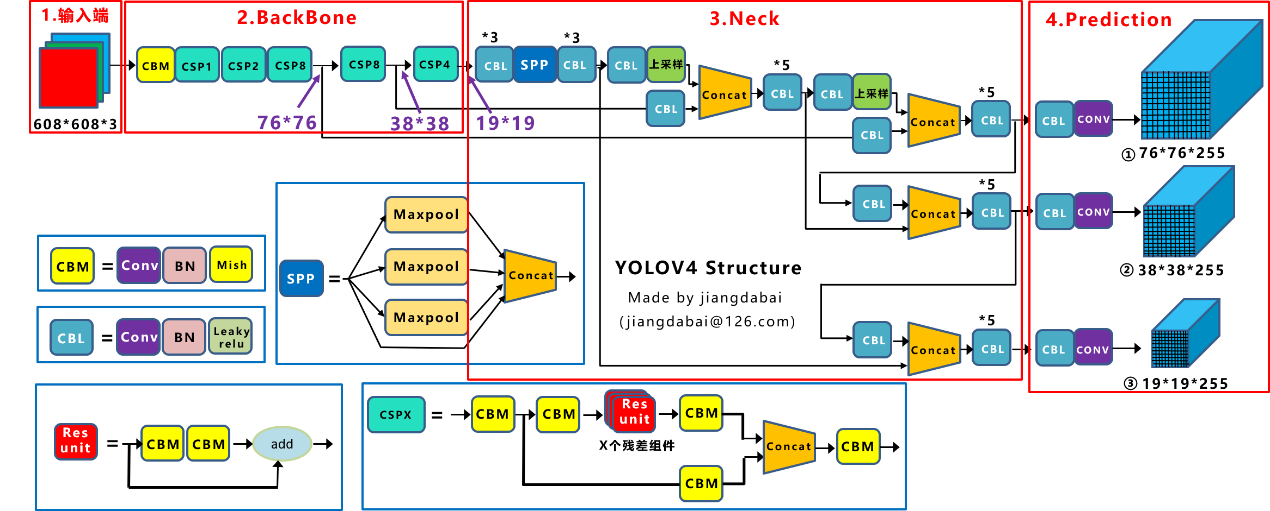


图 2-7 YOLOv4网络框架

正如图中所示，除去输入端之外，YOLOv4网络主要由BackBone、Neck和Prediction Head三个组件组成。

首先是特征提取网络BackBone，这是一个在不同图像细粒度上聚合并形成图像特征的卷积神经网络。YOLOv3中，特征提取网络使用的是Darknet53，而在YOLOv4中，对Darknet53做了一点改进，借鉴CSPNet解决了其他大型卷积神经网络框架Backbone中网络优化的梯度信息重复问题，将梯度的变化从头到尾地集成到特征图中，因此减少了模型的参数量和FLOPS数值，既保证了推理速度和准确率，又减小了模型尺寸，如图2-8。

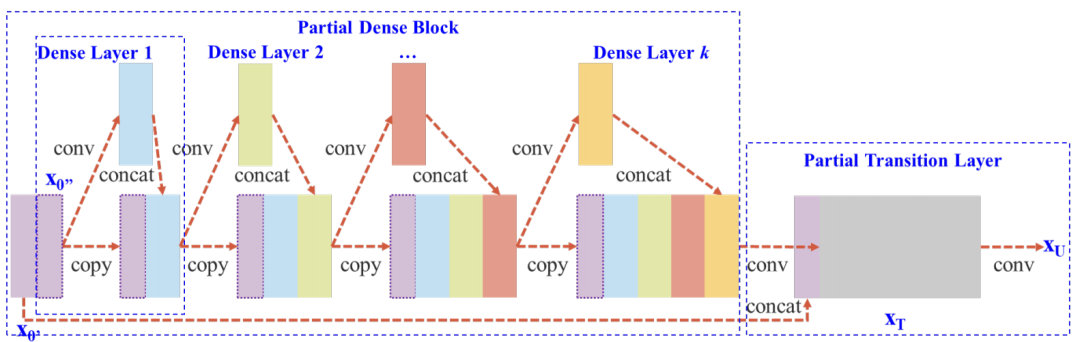


图 2-8 特征提取网络

CSPNet实际上是基于Densnet的思想，复制基础层的特征映射图，通过dense block发送副本到下一个阶段，从而将基础层的特征映射图分离出来。这样可以有效缓解梯度消失问题(通过非常深的网络很难去反推丢失信号) ，支持特征传播，鼓励网络重用特征，从而减少网络参数数量。CSPNet思想可以和ResNet、ResNeXt和DenseNet结合，目前主要有CSPResNext50 和CSPDarknet53两种改造Backbone网络。

基于CSPNet的网络框架，YOLOv4还加入了SPP结构和PAN结构。

其次是Neck组件，这个组件是一系列混合和组合图像特征的网络层，并将图像特征传递到预测层，其中存在的重要结构是SPP结构和PAN结构。

SPP结构又称SPP-Net结构，全称Spatial Pyramid Pooling Networks，主要作用是解决不同尺寸的特征图如何进入全连接层的问题。SPP结构会对任意尺寸的特征图直接进行固定尺寸的池化，来得到固定数量的特征，如图2-9。

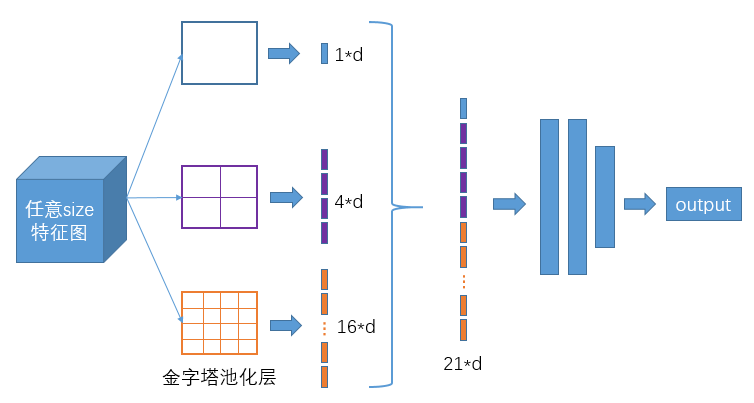


图 2-9 SPP结构

另一重要结构是PAN结构。YOLOv4使用PANet(Path Aggregation Network)代替FPN进行参数聚合以适用于不同level的目标检测, PANet论文中融合的时候使用的方法是Addition，YOLOv4算法将融合的方法由加法改为Concatenation。如图2-10。

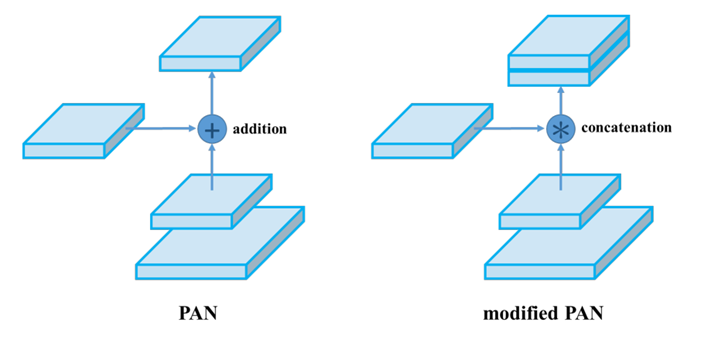


图 2-10 PAN结构

最后是检测头Prediction Head。YOLOv4的检测头部分和YOLOv3相同，这里不作赘述。

此外，YOLOv4的训练和推理策略中也相对YOLOv3做出了一些改进，这里仅对其中的Mish激活函数和CIoU-loss进行介绍。

YOLOv4使用Mish函数取代leacky\_relu函数作为激活函数。Mish函数的定义如下：

采用mish函数的理由主要有两个方面：首先是该激活函数允许比较小的负梯度流入从而保证了信息的流动：其次是利用其无边界的性质避免了饱和，更有利于梯度下降。

在检测头训练过程中，YOLOv4使用CIoU-loss作为损失函数，一次来弥补IoU-loss在模型做出错误预测的情况下无法指明正确方向的缺陷。CIoU计算方法如下：

### 对抗生成算法

#### 2.3.1 RPAttack

RPAttack（Refined Patch Attack on general object detectors）是一种基于像素的对抗性攻击方法。主要思路是在尽量少地修改图像像素的情况下，生成对抗性扰动，使得目标检测器无法正确识别图像中的对象。以往基于补丁的攻击方法对目标检测器产生的扰动区域非常大，容易被人眼察觉，且效率不高，而RPAttack利用一种补丁选择和细化方案，以找到对攻击最重要的像素点，并逐渐去除无关紧要的扰动，实现了在极少修改像素的情况下成功攻击目标检测器。

RPAttack攻击了两个不同架构的检测器，Faster R-CNN和YOLO v4，并试图隐藏这两个检测器的所有对象。像YOLO v4这样的一级检测器直接用置信度分数回归边界框，而像Faster R-CNN这样的两级检测器则使用RPN得到带有置信度分数的提案的边界框。本网络用、表示边界框和置信度分数。目标是用最小的扰动将所有的物体从这两个探测器中隐藏起来，可以表示为:

上式中，为第张图像，表示第*i*个检测器检测到的图像*x*的边界框数，为我们对第*j*张图像加的扰动，为第*j*张图像的扰动面积。

RPAttack的工作流程如图2-11。

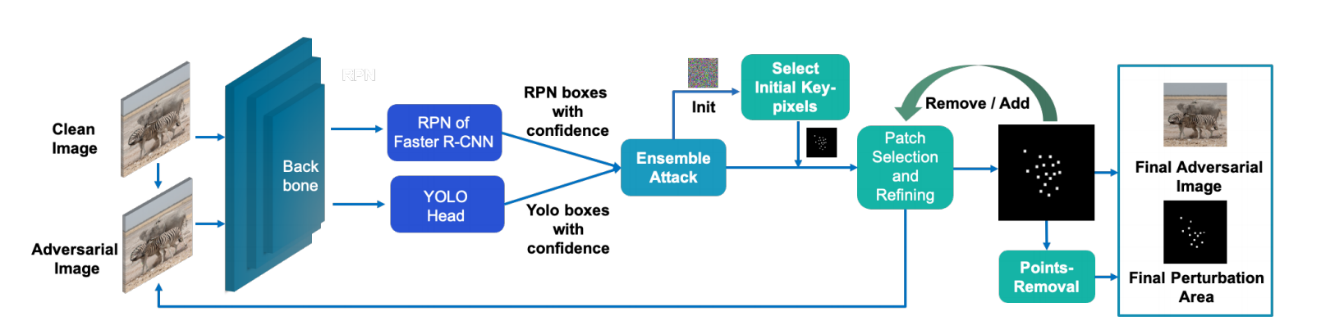


图 2-11 RPAttack工作流程

RPAttack的详细工作过程如下：

（1）对通用对象检测器的攻击

为了对不同的检测器隐藏目标需要降低每个边界框的置信度分数。

基于此，定义损失函数为：

上式中，L(·;·)为均方误差(MSE)， 为第*i*个边界框的置信分数。

在本网络中，同时攻击Faster R-CNN和YOLO v4，整个攻击流水线如上图所示。首先，将图像输入到两个检测器中，得到具有置信度分数的边界框。然后，执行实例级攻击，这意味着可以修改边界框中的所有像素。一次迭代的梯度不能准确地揭示键像素的位置，因此重复实例级攻击，直到所有检测器都无法检测到任何物体，以获得稳定可靠的梯度热图来寻找初始键像素。

接下来，使用从累积梯度热图中获得的初始键像素攻击原始图像。然后，进行了一种集成攻击，在每次*Ak*迭代中自适应地找到最合适的修改位置，并在攻击稳定时去除无关紧要的扰动。最后，为了生成扰动较小的像素，执行点去除，进一步去除对最终结果没有影响的扰动像素，在此过程中，可以得到扰动更小、性能更好的对抗图像。

初始键像素由累积梯度确定，如图2-12所示。具体来说，将从原始图像预测的每个*H* × *W*实例框划分为网格单元(每个单元的大小为*m*×*m*)。然后，对所有实例框中每个单元格的绝对梯度值求和，并选择最上面的*K*个单元格(在实验中设置*K*=5)。由于在实验中，网格形状的对抗块可以成功地攻击具有最小扰动像素区域的图像，因此将顶部的*K*个单元格转移到网格形状的单元格上，并将网格线上的像素作为初始键像素。

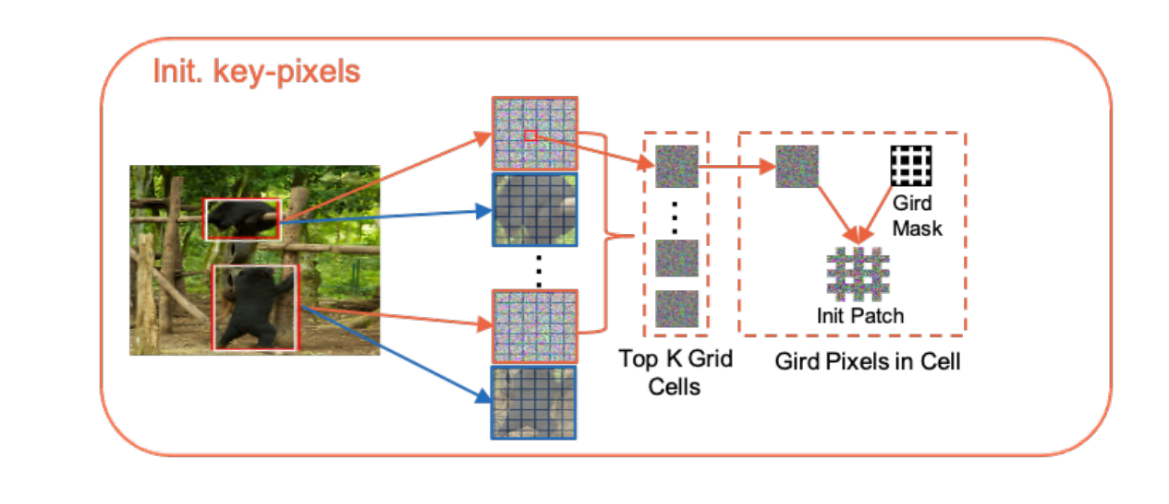


图 2-12 初始键像素的确定

（2）补丁选择和精炼

为了找到攻击的关键像素并去除无关紧要的扰动，本网络提出了一种新的补丁选择和精炼方案，具体流程如图2-13。

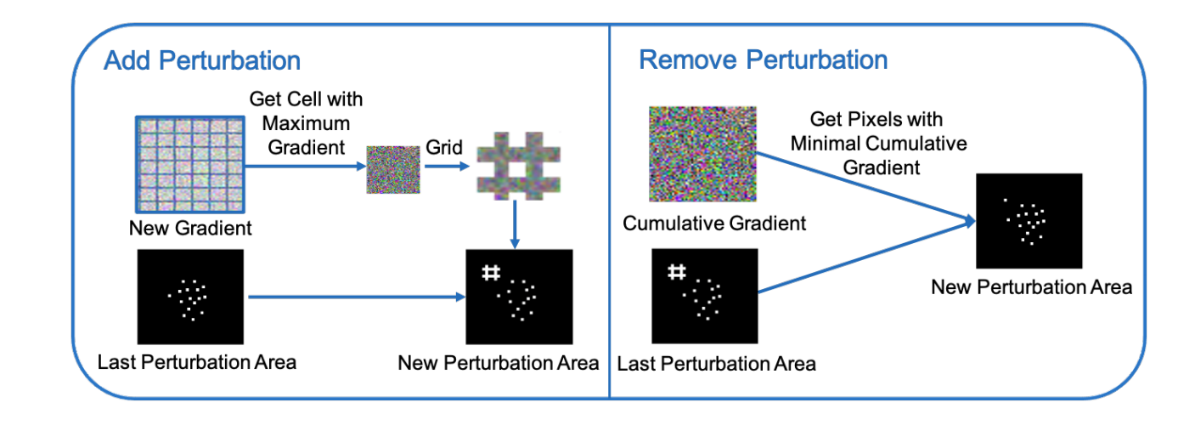
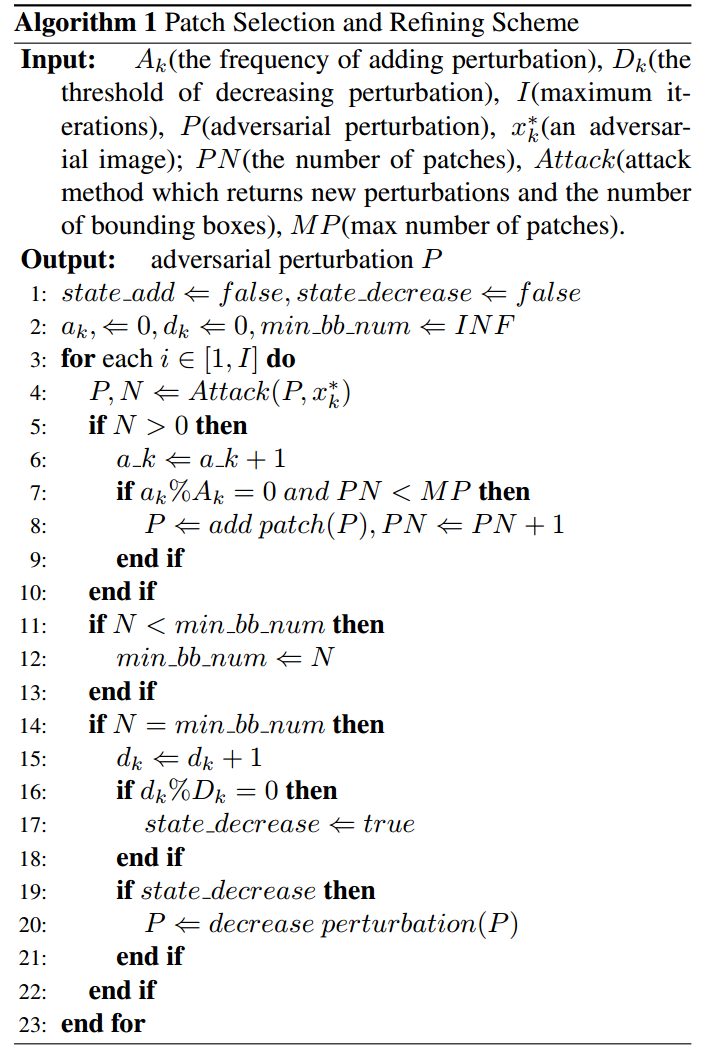


图 2-13 补丁选择与精炼

虽然可以根据累积梯度热图得到键像素的初始位置，但当更新扰动时，键像素会发生变化。此外，初始扰动可能不包括所有键像素。本方法提出了一种自适应的方法来解决这些问题，即在每*Ak*次迭代中根据当前梯度对图像添加一个对抗补丁。这样，就可以在当前迭代中找到最合适的新键像素。

修改的像素并不都在攻击中发挥关键作用。但是，在攻击过程之后删除其中的一些可能会导致攻击性能下降，因为其他键像素没有被充分修改。去除攻击过程中的无关扰动并保持攻击过程，可以对剩余的键像素进行充分修改。在z这项的工作中，当攻击变得稳定时(边界框的数量至少达到最小*Dk*次)，去除像素变化值小于平均变化值除以3的扰动。重复这个过程，直到达到迭代的最大次数*I*。利用该方案可以自适应地找到关键像素，并消除无关的扰动。它不仅大大地提高了RPAttack的攻击性能，并且产生的扰动也更少。

这个流程的伪代码如下：



（3）稳定集合共计

RPAttack旨在同时攻击YOLO v4和Faster R-CNN。然而，每个探测器产生的梯度会影响另一个探测器。为了稳定集成攻击并避免过度优化特定检测器，使用以下方法修改图像:

其中*xk*是第*k*次迭代的对抗样本，*Ji*是用来攻击with检测器的损失函数，*α*是更新的步长，权重*wi*是平衡梯度的参数，*Di*(*x*)是在*x*中被with检测器检测到的实例数。

#### 2.3.2 DPAttack

DPAttack（Diﬀused Patch Attacks）是一种创新的对抗性攻击算法，专门设计来欺骗深度学习模型中的目标检测系统。与传统的全像素攻击或小区域局部像素攻击不同，DPAttack采用了一种扩散补丁的方法，通过在图像中引入少量像素的改变来实现攻击。这种方法使用星形或网格形状的补丁，这些补丁被放置在目标对象的边界框内，利用基于梯度的方法迭代更新补丁，以最大化对目标检测器特征图的影响。DPAttack的设计巧妙地平衡了攻击效果和隐蔽性，使得攻击难以被人眼察觉，同时对目标检测器的干扰却非常显著。

选择补丁位置是实现对抗攻击的第一个问题，在边界框范围内附加补丁是最简介有效的办法，DPAttack使用边界框中心作为星形或网格形的补丁中心。算法的核心在于损失函数的设计，其目的是使类的置信度分数低于阈值，同时引入yizhi6假阳性的提议，损失函数表示为

式中分别表示输入图像、对抗补丁以及补丁分值，*t*是对目标识别分类的阈值，为第*i*个提议分到*c*类得分的函数。只有分为正类的才会有损失，假阳性会得到有效的抑制。

扩散补丁攻击的细节如下所示。

|  |
| --- |
| 算法1： DPAttack |
| 输入：模型函数 ；输入图像；边界框 ；最大迭代次数；分数阈值；步骤。  输出：对抗图像样例  1： 根据检测框位置生成 mask *M*  2：  3:  4:  5:  6:  7:  8:  9:  10:  11: |

# 对抗

### RPAttack

实验的过程如下：

1、配置环境

Python3.8

Cuda10.2

pytorch==1.6.0

Python packages：分别配置mmdetection和yolo v4所需的库

2、将训练好的Faster-rcnn和Yolov4检测网络的权重以及相关参数替换为对应的文件和参数

3、运行hhattack.py文件（包含生成攻击主函数），调整网络中相关参数（如迭代次数等）

文件中主要功能函数如下：

count\_connected\_domin\_score: 计算两个图像之间的差异，并返回差异分数。

connected\_domin\_detect\_and\_score: 检测图像中的连通区域，并计算得分。

adjust\_bbox\_size: 根据给定的比率调整边界框的大小。

makeGrid: 生成网格图像，用于可视化或其他目的。

py\_cpu\_nms: 实现非极大值抑制（NMS）算法，用于去除重叠的检测框。

toTensor: 将图像转换为适合深度学习模型输入的张量。

adjust\_attack\_map: 根据原始图像和攻击图像调整攻击图，减少不必要的噪声。

attack\_imgs: 主要函数，用于初始化模型，加载图像，并执行集成攻击检测。

4、运行结果：

使用在DOTA数据集上训练的Faster R-CNN和YOLO v4检测网络作为检测器，使用RPAttack网络生成的对抗图像示例如图3-1所示。

|  |  |
| --- | --- |
|  |  |
| 1. 原始图检测效果 | |
|  |  |
| 1. 生成的对抗图 | |
|  |  |
| 1. 对抗图像检测效果 | |
| 图 3-1 RPAttack生成对抗样例 | |

由实验结果图的对比可以看到，检测目标为飞机或者直升机，原始未攻击图像的检测效果良好，均能以较大的准确率正确被识别。对抗图像在目标周围生成了一些像素补丁，部分目标已经完全不能被检测到，所有目标的检测识别的准确率均明显下降。可以看出，RPAttack网络的攻击性能较强。

我们使用了46张图片生成对抗图像并对其检测结果进行统计分析，实验数据如下：XXX（图3-2）

### DPAttack

#### 3.2.1 算法复现

根据DPAttack的算法原理，我们首先在DOTA数据集上进行复现，用DPAttack同时攻击Faster R-CNN和YOLOv4网络，得到对抗补丁。对原图和对抗样例的检测结果如图3-3。

|  |  |
| --- | --- |
| 8_test | 14_test |
| 1. 原图检测 | |
| 8 | 14 |
| 1. 对抗样例检测 | |
| 图 3-3 DPAttack生成对抗样例 | |

可以发现加上对抗补丁之后检测器无法检测到目标，对抗攻击取得了不错的效果。但是在测试中发现，若对对抗样例进行简单的图片变换（如旋转、明暗变化等），就会导致攻击的失败，如图3-4，由此可见，在已经固定对抗补丁的情况下，针对不同的成像条件，攻击的效果会有不同，即补丁的鲁棒性不足。

|  |  |
| --- | --- |
| 1616_test | 1616_4_test |
| 1. 初始对抗样例 | 1. 旋转后的对抗样例 |
| 图 3-4 DPAttack鲁棒性测试 | |

同时，原算法中的补丁会对目标造成部分遮挡，这导致现实中应用的可行性不高，现实中的补丁难以平整、准确地施加到希望掩饰的目标表面。可见，DPAttack并不具备较高的现实应用潜力。

#### 3.2.2 算法改进

针对以上两点问题，我们做出了几点改进，旨在提升算法的鲁棒性，同时让其物理实现更近一步。

为了提高对抗补丁的现实可行性，我们将其修改为如图3-5的形状。考虑到我们的数据集是高空遥感图像，只需在底片印刷或者铺放相同纹理的图案，即可将对抗补丁应用于现实世界。

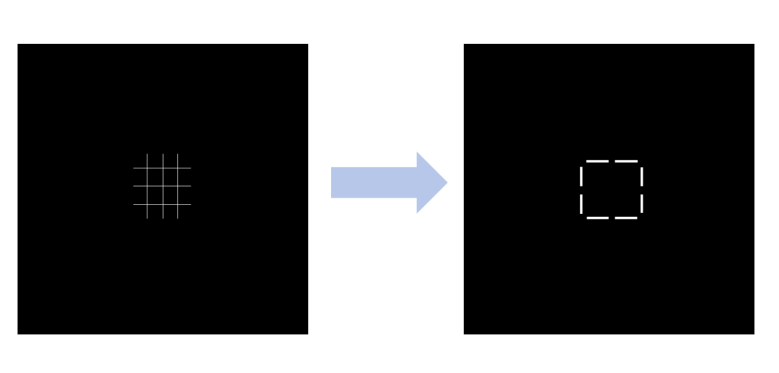


图 3-5 补丁形状优化

为了增强算法的鲁棒性，我们在生成对抗补丁的过程中就对图像施加随机的扰动，包括旋转任意角度（针对现实成像角度的多样性）以及亮度、饱和度、对比度的随机浮动（针对现实的成像光照条件的不确定性）。

改进之后的算法流程如图3-6所示。

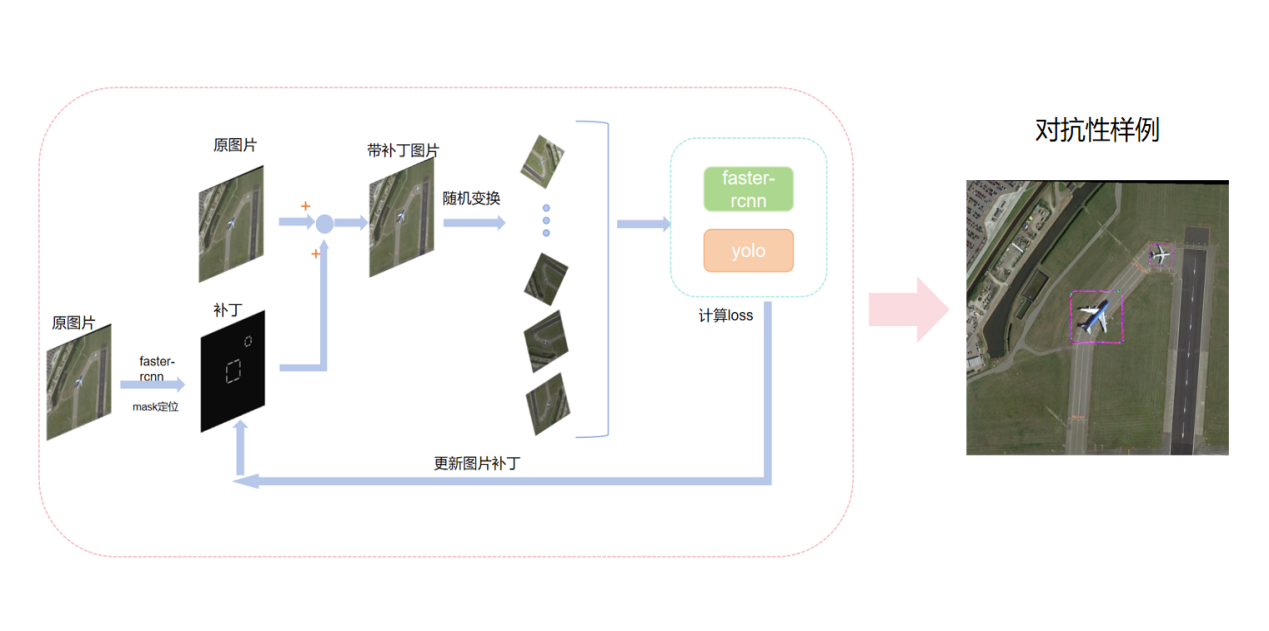


图 3-6 改进后的算法流程

### XXX

### 对比分析

# 结束语

在模式识别课程设计中，一开始拿到这个题目，大家都感到很陌生，不知如何上手。但小组内的成员都积极讨论，大家一起分析问题，并在老师的讲解下拨云见日，慢慢地确立了研究计划。小组内的每个人都有明确的分工，我负责利用RPAttack方法生成对抗图像，以攻击深度检测网络。这个过程不仅是对技术的考验，更是一次深刻的自我学习和成长之旅。

在项目开始时，我首先遇到了环境配置的问题。软件的安装和依赖库的配置看似简单，却隐藏着许多细节上的陷阱。由于生成对抗的代码使用的相关检测网络的版本都很老旧，我花费了很多时间在调试和修正上，并且和小组内的同学们一起讨论解决这个问题。这个过程虽然充满挫折，但也极大地锻炼了我的耐心和解决问题的能力。我学会了如何细致地分析问题，一步步地排查和解决每一个可能出现的错误。

随后，我面临了另一个更为棘手的问题：检测网络和生成对抗图像的网络版本不一致。这导致我在生成对抗图像时遇到了重重障碍。我不得不重新审视我的工作流程，调整策略，通过不懈的努力，我终于解决了这个问题——在应用不同版本的检测网络到RPAttack对抗网络时，只需修改对应的参数值而不能改检测网络结构文件。这个过程不仅加深了我对深度学习模型的理解，也提高了我的技术适应性和创新能力。

通过这次课程设计，我获得了宝贵的个人成长和深刻的认识。我学会了如何在面对困难时保持冷静，如何独立思考和解决问题，并培养了我的团队协作的能力。更重要的是，这次经历让我认识到了在人工智能领域，持续的学习和创新是至关重要的。在未来，要继续保持学习探索的热情与持之以恒的努力，为人工智能领域的发展做出自己的贡献。