参赛密码 _____

（由组委会填写）

“华为杯”第十四届中国研究生数学建模竞赛

学　　校	上海大学
参赛队号	**10280168**
队员姓名	**1. 杨浩**
	2. 刘致金
	3.汪洋鑫

参赛密码 _____

（由组委会填写）

“华为杯”第十四届中国研究生数学建模竞赛

题 目 基于监控视频的前景目标提取

摘 要：

随着监控摄像头的井喷式增长，监控视频数据量也大幅增长，这使得对监控视频的前景目标提取技术变得尤为重要，它是处理和分析监控视频中有用信息的基础。

传统的基于背景模板的背景去除方法提取前景目标不够精确，对镜头晃动、光照突变等干扰不鲁棒。近些年来机器学习方法也被用来解决这个问题，但是存在诸多缺陷，如用于模型训练的大量繁琐的前景标注，模型泛化能力不强等。为了解决上述描述方法中存在的缺陷，我们提出了一种半自动化的基于深度学习方法的移动前景目标提取模型。

考虑到视频监控的应用场景，监控摄像头一般固定或者小范围偏移，监控视频的背景场景变化有限。我们的模型首先采集一小部分监控视频帧进行线下模型训练和初始化，视频中有很多的冗余信息，因此用来训练的视频帧数量极小就可以得到较准确的检测结果。训练好的模型可直接用于整个监控视频后续的前景提取。

具体上，针对前景目标在视频帧中大小不一的特点，我们在在卷积神经网络的基础上提出使用多尺度卷积神经网络（MSCNN）进行模型构建。每一帧进行两个尺度的下采样，训练出三个卷积神经网络模型（CNN）。每个网络包括四个卷

积层和两个全连接层，为了降低复杂度，三个 CNN 共享参数。实验结果表明，我们的模型具有很高的准确度，对光照、动态背景以及噪声具有很强的鲁棒性，而且模型不用线上更新，因此检测复杂度不高。
关键字： 机器学习 MSCNN 前景提取

一、问题重述

监控电子设备的大量应用产生了信息量巨大的视频数据，而监控视频的冗余特性致使其数据的有效管理变得困难，传统的基于人工的视频数据检索耗费人力和时间。因此，监控视频分析，如物体追踪、场景理解、异常检测、交通分析等，具有广泛的应用价值[1]。

如何快速、有效提取出监控视频中的前景目标，是监控视频分析基础但十分重要的问题。视频前景提取能够对的视频处理任务提供有效的辅助，以筛选与跟踪夜晚时罪犯这一应用为例：若能够预先提取视频前景目标，判断出未包含移动前景目标的视频，并事先从公安人员的辨识范围中排除；对于剩下包含了移动目标的视频，只需辨识排除了背景干扰的纯粹前景，对比度显著，肉眼更易辨识。因此，这一技术已被广泛应用于视频目标追踪，城市交通检测，长时场景监测，视频动作捕捉，视频压缩等应用中。但其难度在于，具有移动前景目标的视频往往伴随着复杂多变的、动态的背景，难以提取准确的前景目标。

问题的提出：

(1) 对一个不包含动态背景、摄像头稳定拍摄时间大约 5 秒的监控视频，构造提取前景目标的数学模型，并对该模型设计有效的求解方法，

(2) 对包含动态背景信息的监控视频，设计有效的前景目标提取方案。

(3) 在监控视频中，当监控摄像头发生晃动或偏移时，视频也会发生短暂的抖动现象。对这种类型的视频，如何有效地提取前景目标？

(4) 在附件 3 中提供了 8 组视频。请利用你们所构造的建模方法，从每组视频中选出包含显著前景目标的视频帧标号。

(5) 如何通过从不同角度同时拍摄的近似同一地点的多个监控视频中有效检测和提取视频前景目标？

(6) 利用所获取前景目标信息，能否自动判断监控视频中有无异常事件？

二、问题分析

问题一到问题三分别是针对不同干扰，如背景动态，镜头晃动下监控视频的前景提取。动态背景（如树的晃动，下雨等），镜头平移，晃动这些干扰都是实际情况下可能会出现的情况，这就要求建立的模型要有很强的鲁棒性，能满足不同环境下运动前景目标提取的要求。问题四要求对出现运动前景的视频帧进行定位，要求能及时追踪前景目标的行为动态。问题五的提出是在不同角度下的相似背景的监控视频的前景提取，这是一个比较有挑战性的问题。三个视频是同步的，而且相互之间有相关性，并不是完全独立的三个视频。如果简单的将三个视频拼接起来放在一块检测，尽管背景相似，但是它们之间的切换是有跳跃性的，这要求模型的具有很好的泛化能力。问题六提出对前景目标进行行为分析，对前景目标行为进行异常判断，这要求不但要能准确定位分离出前景目标，还要对前景目标进行追踪，对运动轨迹进行建模，进行异常行为分析。

三、模型的建立

MSCNN 模型概述：

如图 1 所示，我们的模型建立分为三步：(i)选取一部分视频帧作为训练数据；(ii)训练模型；(iii)检测监控视频。

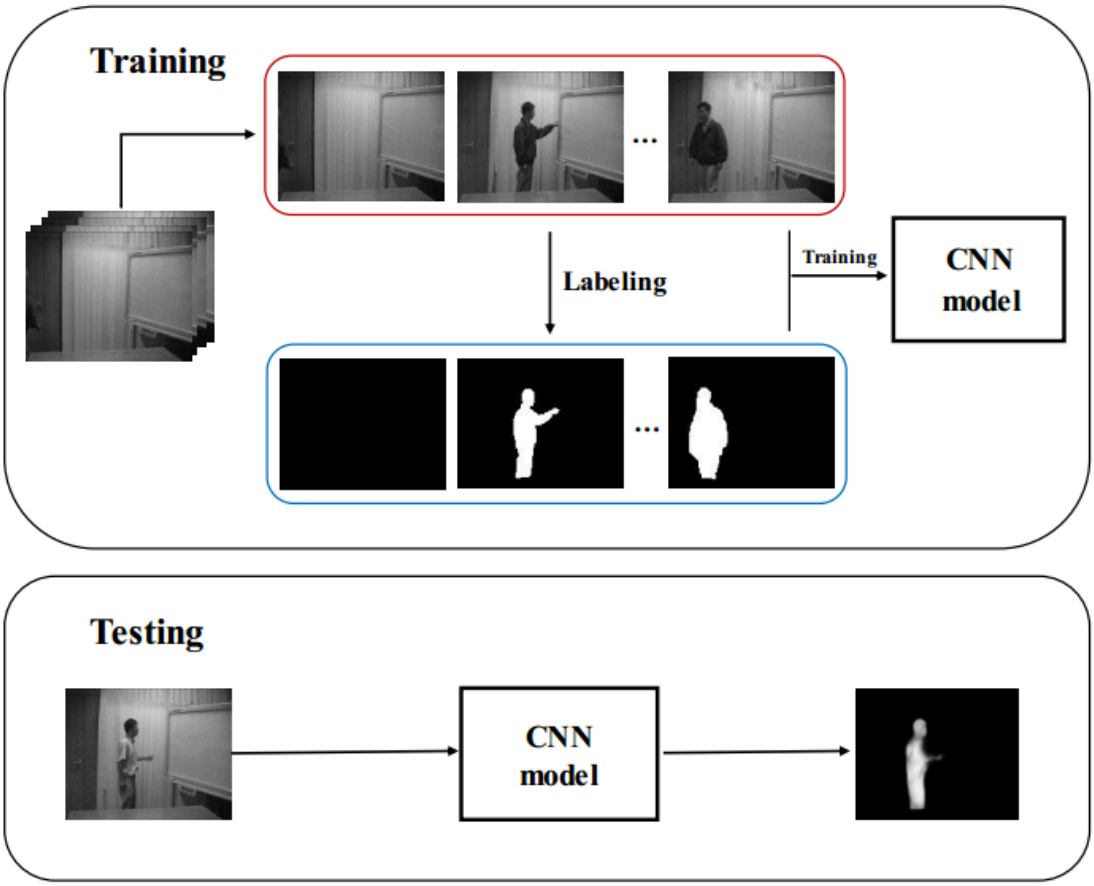


图 1 模型框架

我们用来学习前景和背景的模型是深度 CNN 模型，之所以选择 CNN 有两个原因：1，CNN 模型能很好的学习到拟合数据的特征。这种方法与存在的人工选择的特征，如 HOG，SIFT，LBP，相比有更大的优势。并且多层的结构得到的特征能够提取视频不同层次的特征；2，CNN 能利用并行卷积操作，因此在测试阶段的速度会非常快。图 2 是我们设计的 basic CNN 模型，包含四层卷积层以及两层全连接层[2-3]。

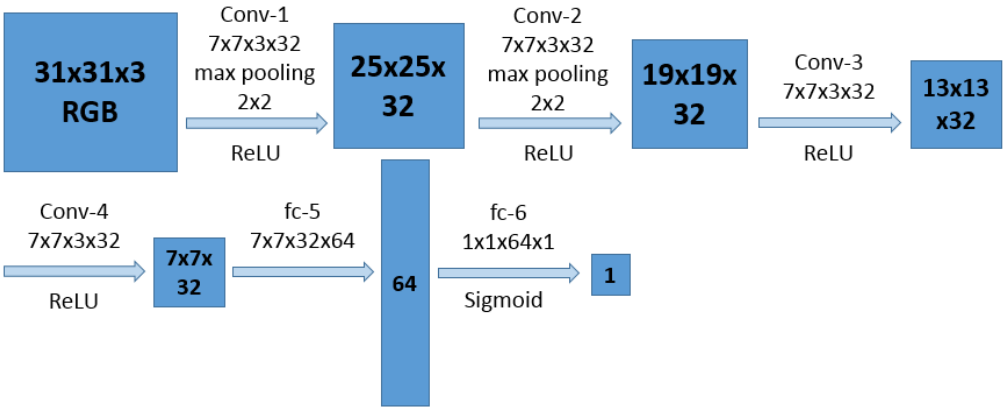


图 2 basic CNN

相比于对全图进行分类，我们的目的是要预测每个像素所属前景或后景。为

了做到预测像素分类的目的，我们在每个像素的周围建立一个 31X31 的块，并把这个块作为要分类的图像。

Basic CNN 的详细描述见表 1。如表中所描述，我们的 basic CNN 模型包含了 4 层卷积层以及 2 层全连接层。激活函数采用的是 ReLU 函数，如公式(1)，该函数不会损失反向回传的梯度值，在深度网络前后向计算的传递性方面效果好，适合 CNN 这样层数比较深的网络。Fc-6 层后接 sigmoid 函数（参见公式(2)）是为了将预测结果值转变到 0 和 1 之间的概率值，值越靠近 1 表示该像素属于前景的概率越大[4-5]。

$$f(x) = max(x) \tag{1}$$

$$S(x) = \frac{1}{1+e^{-x}} \tag{2}$$

由于 CNN 的输出是一似然概率，所以我们考虑使用交叉上损失函数作为训练用的损失函数[6]：

$$Loss = -\frac{1}{K}\sum_{K=1}^{K}[C_K log\hat{p}_k + (1 - C_K)\log(1 - \hat{p}_k)] \tag{3}$$

K是训练的像素总数，C_K是标记的标签，$\hat{p}_k$是预测的前景概率。

Table 1 basic CNN 模型结构

Layer	1	2	3	4	5	6
Stage	conv	conv	conv	conv	FC	FC
Input size	31x31	25x25	19x19	13x13	7x7	1x1
Filter size	7x7	7x7	7x7	7x7	-	-
Conv stride	1x1	1x1	1x1	1x1	-	-
Pooling method	max	max	-	-	-	-
Pooling size	2x2	2x2	-	-	-	-
Pooling stride	1x1	1x1	-	-	-	-
Padding stride	[0,1,0,1]	[0,1,0,1]	-	-	-	-
#Channels	32	32	32	32	64	1

从 basic CNN 的模型以及描述中可以发现该模型的一个缺点是它固定了输入的块的大小是 31x31，因此它对于处理前景或者背景小于该块的大小的视频效果好。但是输入视频的前景，如车子，往往是大于这个尺寸的，特别是当前景离摄像头较近时，如图 3。

图 3 包含大的移动物体的视频帧

为了解决这个问题，我们采用 multi-scale CNN（MSCNN）模型, 如图 4。给定一个待分割的图片I，我们将这张图片用两个尺度进行缩放得到I_{scale1}和I_{scale2}，我们取的缩放尺度分别是 0.75 和 0.5。然后三张图片分别送入 Basic CNN 中进行训练得到不同尺度的输出O，O_{scale1}和O_{scale2}。再将O_{scale1}和O_{scale2}放大到原输入图像I的大小。最终得到的O_{final}是三种尺度结果的平均值。另外，图中的三个 CNN 模型共享参数，且参数的大小完全是从数据中学习到的。

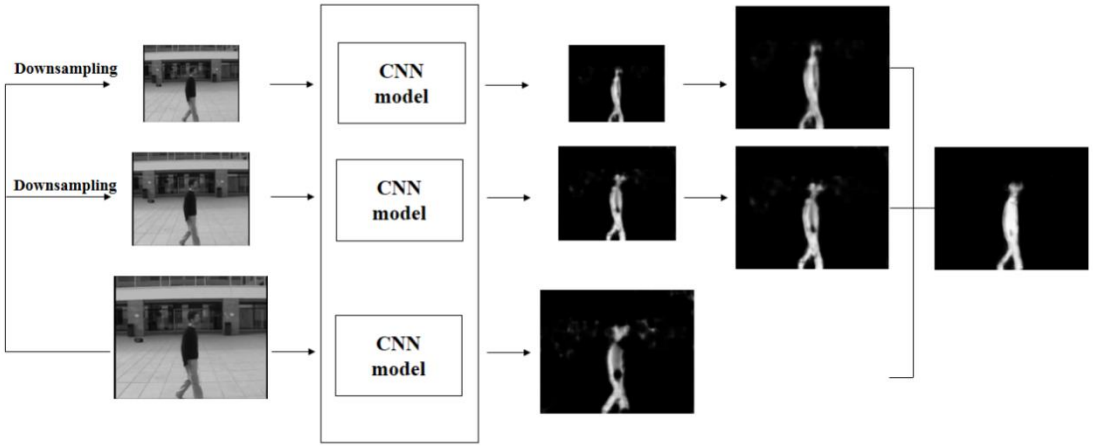


图 4 MSCNN 模型

三、问题分析与求解

问题 1

对于静态背景我们直接用构造的 MSCNN 模型来提取前景目标，我们测试了附件 3 中 lobby 的整个视频，图片 5 展示的是部分帧检测结果：

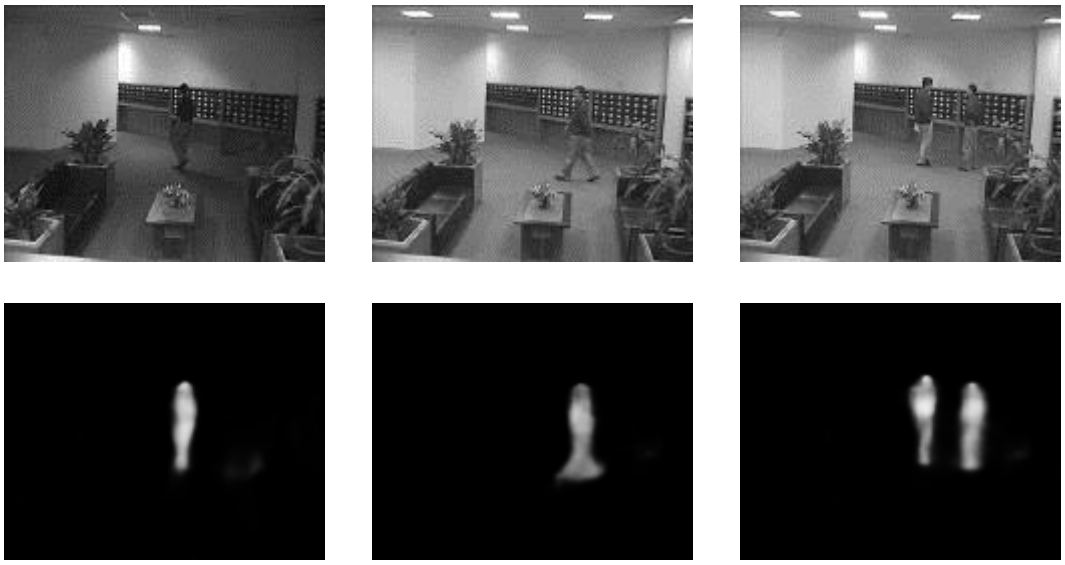


图 5 视频中 360，984，1538 帧原图以及前景提取图

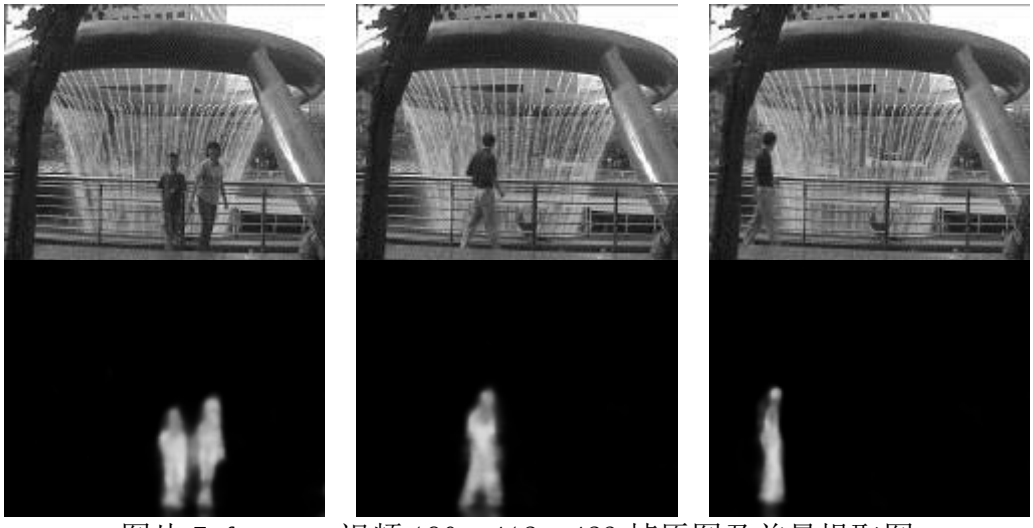
我们另外测试了收集的视频，如图 6 所示：

图 6 测试视频 1387，2399，4591 帧原图以及前景提取图

从测试结果中可以看出，我们设计使用的 MSCNN 能有效检测静态背景监控视频。

问题 2

对于动态背景我们直接用构造的 MSCNN 模型来提取前景目标，我们测试了附件 3 的 fountain 和 curtain 的整个视频，图片 7,8 分别展示 fountain 和 curtain 的部分帧检测结果。



图片 7 fountain 视频 190，412，423 帧原图及前景提取图

图片 8 curtain 视频 2207，2797， 2903 帧原图及前景提取图

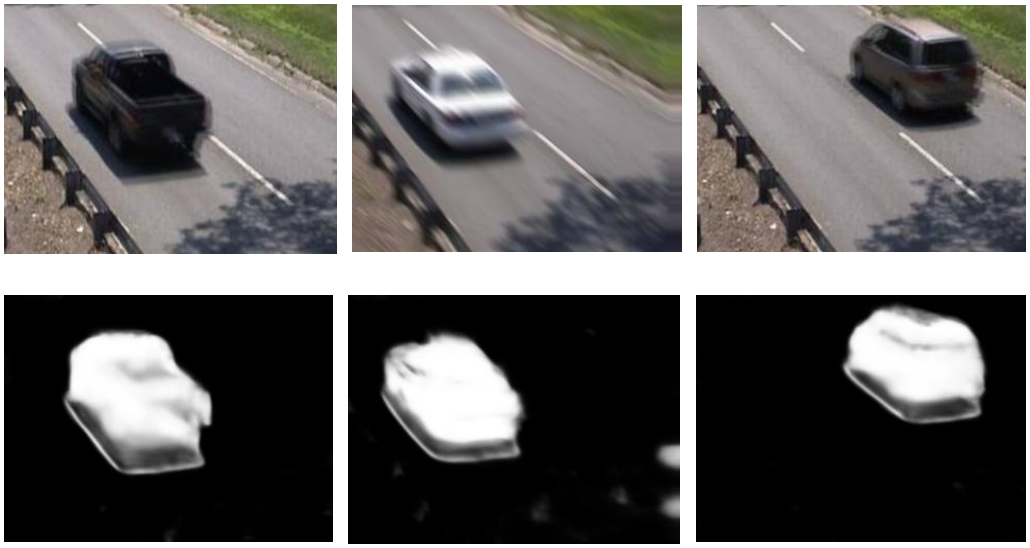
从测试结果中可以看出，我们设计使用的 MSCNN 能有效检测动态背景监控视频，动态背景在我们的 MSCNN 模型下与静态背景的表现几乎没有区别。

问题 3

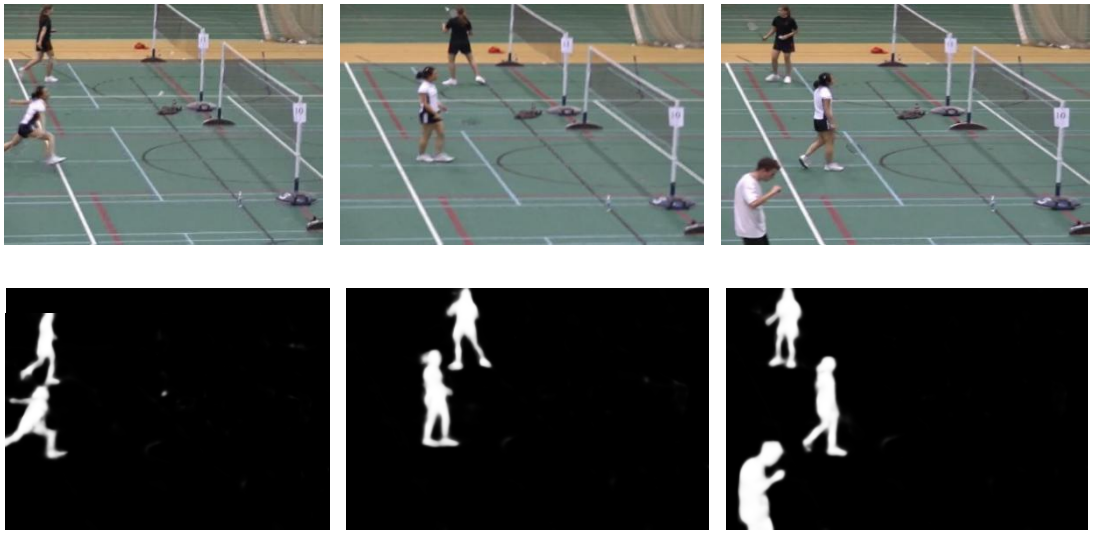
对于监控摄像头发生抖动的视频，我们用构造的 MSCNN 模型测试了 4 组视频，分别是附件 2 的 car6 和 car7 视频，以及从提供网站中下载的数据集 dataset2014 中 cameraJitter 中的 traffic 和 badminton 视频。其中 traffic 和 badminton 视频的监控摄像头发生剧烈抖动。下面分别展示 4 组视频的检测结果。

图片 9 car6 视频 4，12，28 帧原图及前景提取图

图片 10 car7 视频 8，15，24 帧原图及前景提取图



图片 10 traffic 视频 966，1107，1247 帧原图及前景提取图



图片 11 badminton 视频 850，922，963 帧原图及前景提取图

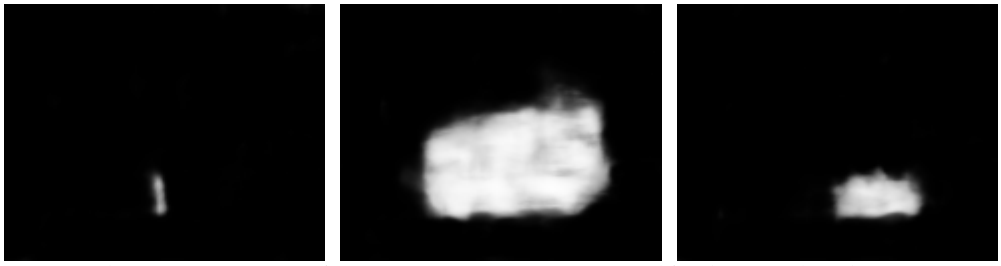
从测试结果中可以看出，无论监控摄像头发生平移或是剧烈抖动，我们设计使用的 MSCNN 能有效提取目标前景，具有良好的抗干扰性和鲁棒性。

问题 4

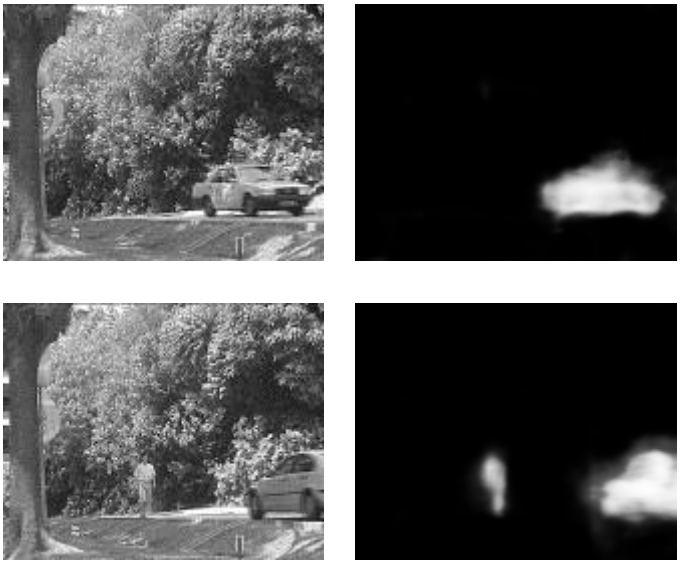
对于附录 3 提供的 8 组视频，我们用构造的 MSCNN 模型测试的结果如下：

campus 前景帧:

85, 200-224, 306-522, 600, 644-683, 690-712, 737-905, 1005-1036, 1193, 1264, 1329-1374, 1377-1405。



图片 12 campus 视频 431，666，1364 示例帧原图及前景提取图

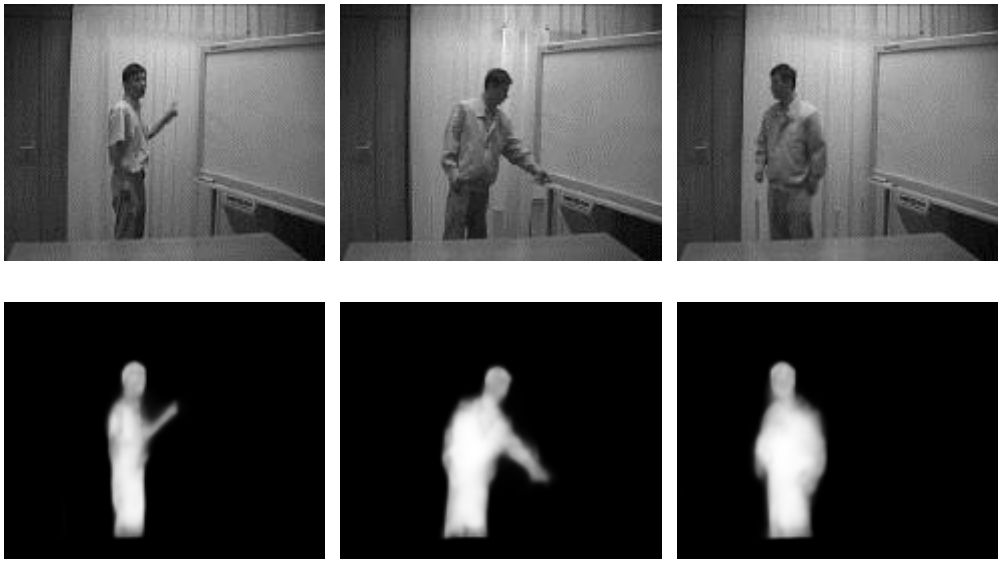


图片 13 campus 视频 85，600 干扰帧原图及前景提取图

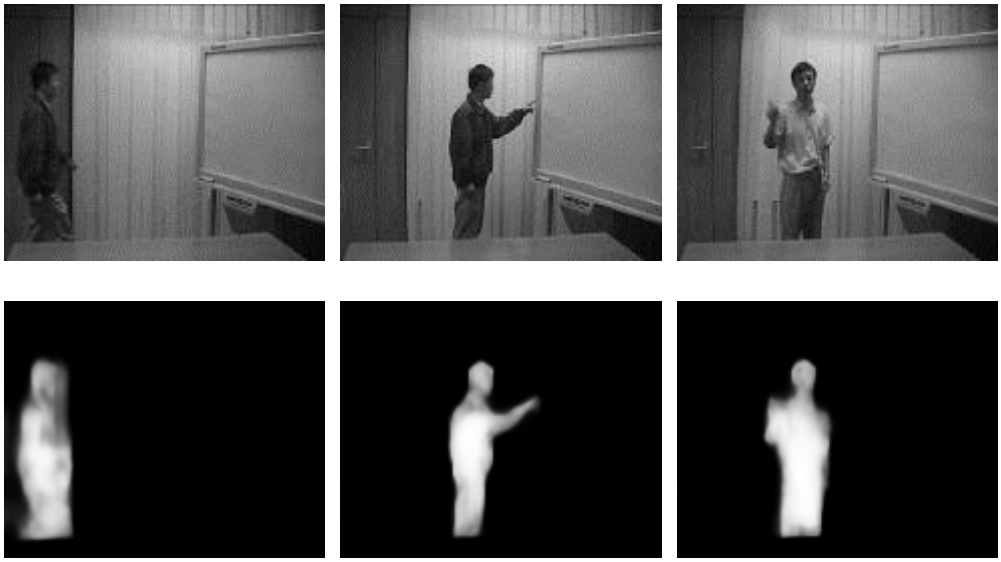
对于动态背景视频 campus，我们准确的追踪并提取到了前景目标，同时能敏锐的检测出干扰帧。campus 视频中 85，600 帧为干扰帧，从监控视频中一闪而过，不过我们提出的算法能准确的把它捕捉并提取。

curtain 前景帧：

411, 967, 1561, 1757-1907, 2126, 2168-2316, 2642, 2768-2931。



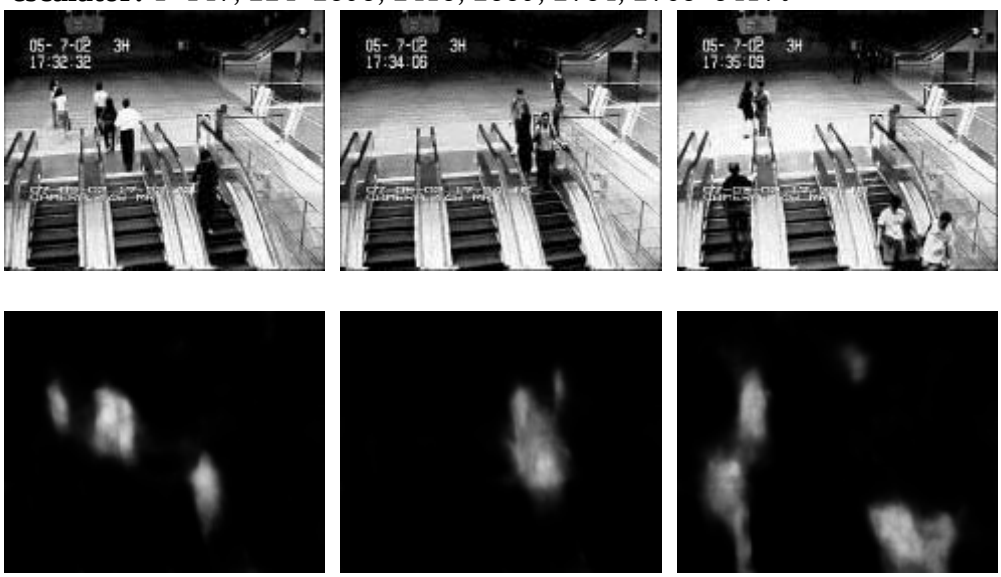
图片 14 curtain 视频 2207，2797， 2903 示例帧原图及前景提取图



图片 15 curtain 视频 411，967, 1564 干扰帧原图及前景提取图

对于动态背景视频 curtain，我们准确的追踪并提取到了前景目标，同时能敏锐的检测出干扰帧。curtain 视频中 411，967, 1561 帧为干扰帧，从监控视频中一闪而过，不过我们模型能准确的把它捕捉并提取。

escalator: 1-147, 224-2393, 2415, 2539, 2754, 2768-3417。



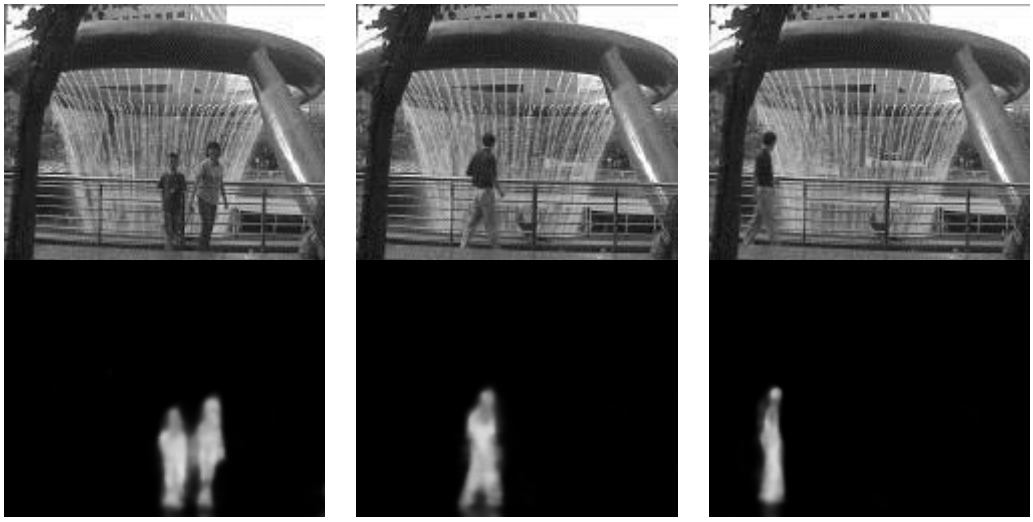
图片 16 escalator 视频 352, 1252, 1868 示例帧原图及前景提取图

352, 1252, 1868 分别是处理整个视频得到的某些帧的结果，可以看到，虽然利用 MSCNN 可也提取前景目标，并且准确的追踪到了目标，但是提取的目标轮廓发生了变形，检测结果并不理想。这与模型训练的结果有关，整个视频由 3417 帧，我们用来训练的帧数是 68 帧，训练数据少量是导致模型轮廓变形的原因，但是也不可用太多的视频帧训练，因为会耗费人力，这与我们的初衷不符。

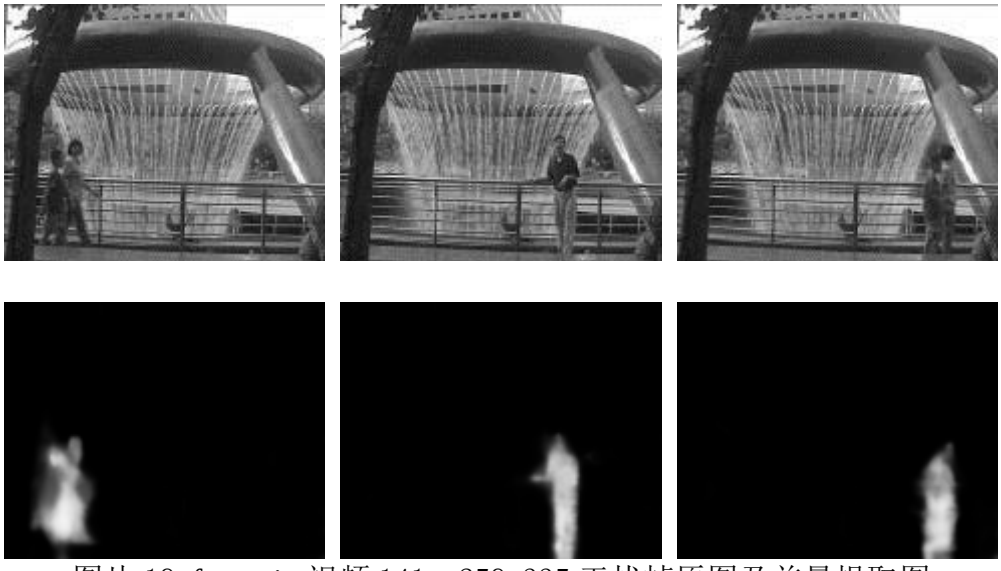
图片 17 escalator 视频 2415，2539, 2754 干扰帧原图及前景提取图

2415，2539, 2754 是 escalator 视频中出现的干扰帧，虽然轮廓变形，但是检测的前景目标的位置以及目标追踪都达到了精确的程度。

fountain：141, 156-210, 259, 335, 408-523。



图片 18 fountain 视频 190，412，423 示例帧原图及前景提取图



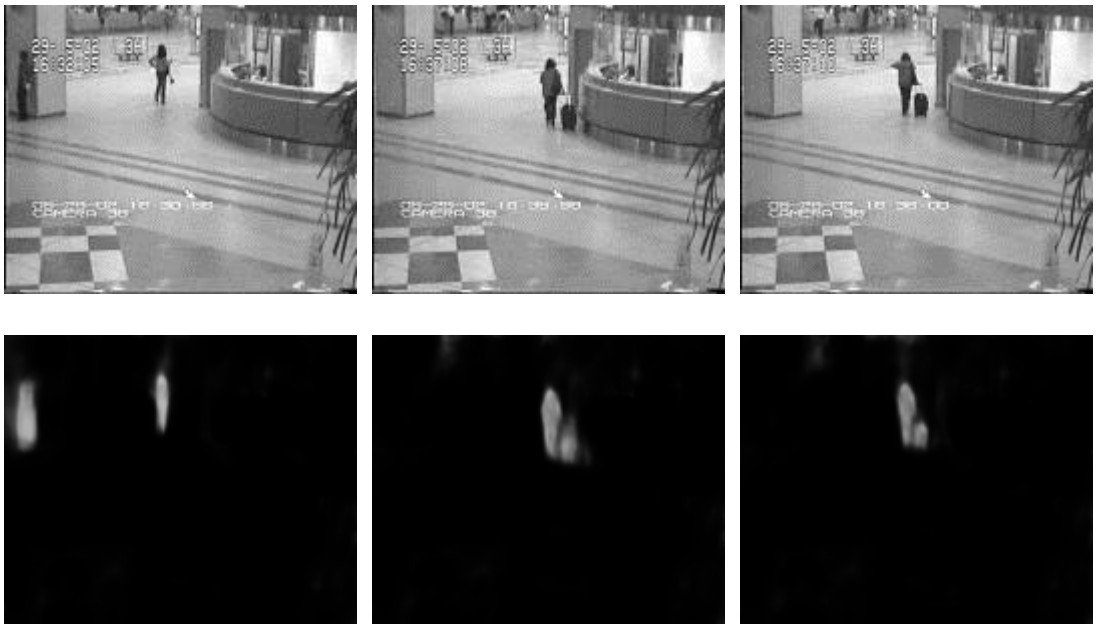
图片 19 fountain 视频 141，259, 335 干扰帧原图及前景提取图

对于动态背景视频 fountain，我们准确的追踪并提取到了前景目标，同时能敏锐的检测出干扰帧。

hall 前景帧:

1-510, 578, 599-739, 795, 817-1055, 1138, 1153-1213, 1246, 1278-3534。

图片 20 hall 视频第 5， 1587, 2823 示例帧原图及前景提取图

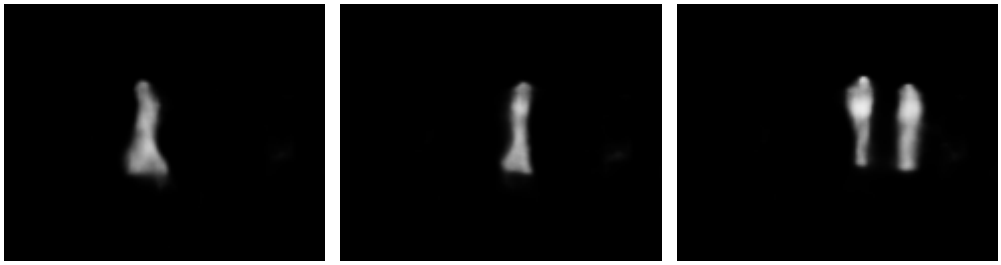


图片 21 hall 视频 578，795, 1138 干扰帧原图及前景提取图

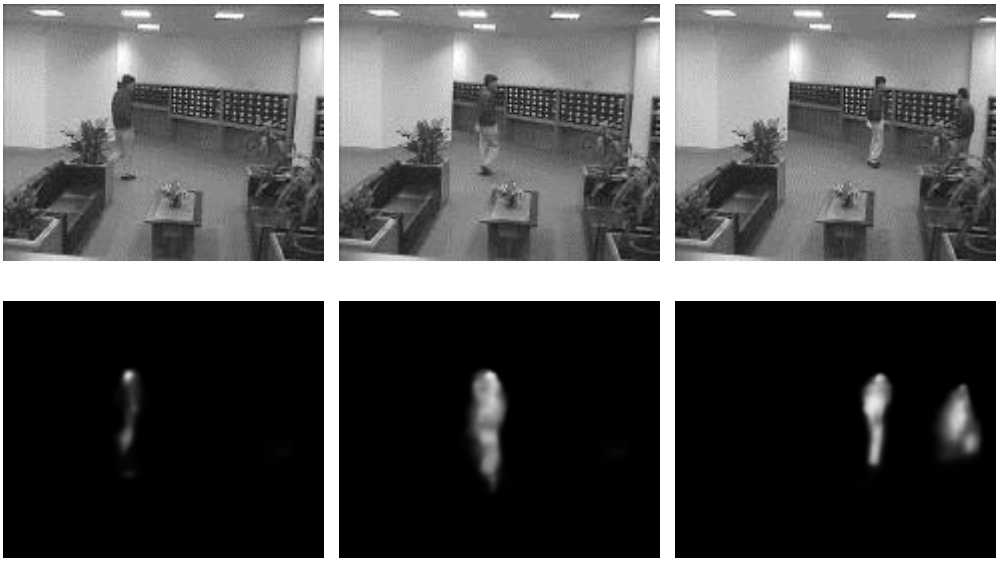
对于密集人群背景视频 hall，我们准确的追踪并提取到了前景目标，同时能敏锐的检测出干扰帧。

lobby 前景帧:

79, 154-196, 259, 345-394, 521, 622-669, 870, 962-1038 , 1161, 1238-1282, 1333-1538。



图片 22 lobby 视频 369，1266，1534 示例帧原图及前景提取图



图片 23 lobby 视频 79，259，870 干扰帧原图及前景提取图

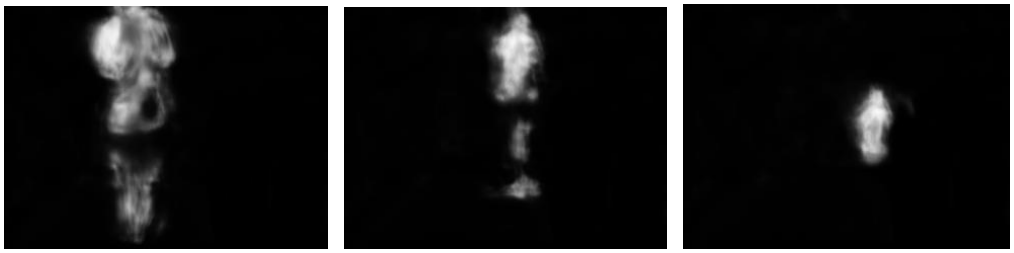
对于光照突变干扰背景视频 lobby，我们首次实验时在 1062-1078 帧受到了光照突变的影响，产生了错误的前景区域，不过这对后续帧的前景结果的质量并没有影响。后来我们发现，是因为这段光照突变的帧序列中没有一帧参与到训练模型中，导致效果不好，当我们选取这段序列中的一帧加入训练阶段再次训练后，结果得到改善，错误消失。当参与训练的帧图像选取合适时，本模型可以做到对干扰光照的鲁棒性。

office 前景帧： office 的检测结果混乱。

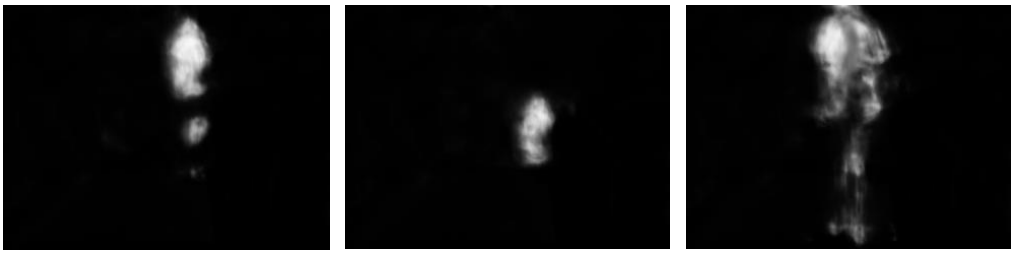
模型的初始化以及各种参数对模型的影响会造成检测失败的情况出现，我们在检测各类视频时是使用统一的参数。统一参数会影响模型对某一类的视频造成失败的现象，我们并不想为特定的视频调整参数来达到提取前景的目的，这会影响模型的普适性。同时由于模型本身存在缺陷，比如选取 loss 函数，以及模型的构建等都会对检测视频的种类造成影响。

overpass 前景帧：

374，547-582，968，1551，2335-2836。



图片 24 overpass 视频 2400, 2507， 2697 示例帧原图及前景提取图



图片 25 overpass 视频 374，968，1551 干扰帧原图及前景提取图

对于动态干扰背景视频 overpass，我们准确的追踪并提取到了前景目标，而且对干扰具有足够的鲁棒性。

问题 5

为了测试能否从多个不同角度拍摄的监控视频提取前景目标，我们采集了一组视频 campus7, 如图 26 所示：

(a)campus7_c0 (b)campus7_c1 (c)campus7_c2

图片 26 三组不同角度拍摄的相似场景视频 campus7

我们首先对三组视频分别训练、分别检测，结果如下：

图片 27 campus7_c0 视频 75，1350,2400 示例帧原图及前景提取图

图片 28 campus7_c1 视频 1263，2983，4368 示例帧原图及前景提取图

图片 29 campus7_c2 视频 408，1055，4480 示例帧原图及前景提取图

为了试验能否提取相似场景，我们做了两个实验，第一个试验是抽取视频 campus7_c0 极小部分帧进行训练，然后把训练模型加到另外两组视频上面进行测试。第二个实验是分别采样三组视频帧共同组合训练一个模型，采样时主要采样其中一组视频，另外两组视频采样较少帧数，最后测试时再加到三组视频上面

进行测试。两个实验测试的结果分别与之前单独测试的结果进行对比如下：

实验一：

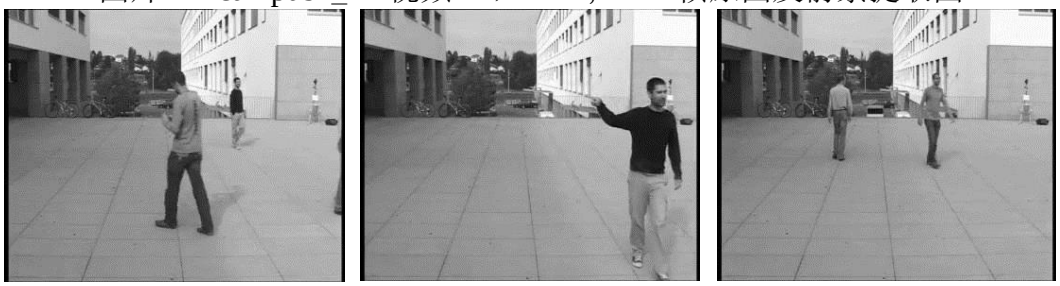
图片 30 campus7_c1 视频 1263，2983，4368 帧原图及前景提取图

图片 31 campus7_c2 视频 408，1055，4480 示例帧原图及前景提取图

从结果中可以看出，尽管模型能提取前景，但是受到背景干扰的影响也极大。从两组测试数据来看，单一场景训练出的模型的抗干扰能力不够强，相似的背景虽然能够移植，但是检测的精度也收到影响。

实验二：

图片 32 campus7_c0 视频 75，1350, 2400 帧原图及前景提取图



图片 33 campus7_c1 视频 1263，2983，4368 帧原图及前景提取图

图片 34 campus7_c2 视频 408，1055，4480 帧原图及前景提取图

从结果可以看出，实验二的检测效果明显好于试验一，这是由于在模型训练过程中加入了少量相关背景的训练，这样即使视频发生了突变，模型依旧能够提取目标前景，并且不受背景的干扰，检测精度高。

提出的 MSCNN 模型对于处理相似背景的监控视频有着较强的学习能力，能够处理多个角度切换的问题。

问题 6

尝试提出利用 C3D 学习网络对这一问题进行解决。

C3D 深度学习网络的核心思想是网络可以学习到连续图片的时域信息和空间信息。使用 MSCNN 模型提取到监控视频的目标前景之后，再将这些前景图像输入到 C3D 网络中。一次性输入多张图片，然后进行 3D 卷积操作，通过 3D 卷积操作，以及后序的池化，非线性激活函数，全连接层等可以提取并学习出的目标前景在时域上的动作变化规律，从而可以预测出异常事件。该问题在 C3D 上面是个创新点，这方面还没有人做，标签也没有加，监督学习开展受阻，而人工加标签又极耗费时间，竞赛时间有限，并没有实际解决该问题。

四、模型评价

本文提出的 MSCNN 网络模型，能够客服动态背景，镜头晃动，多角度拍摄等的干扰，精确地提取出监控视频中的目标前景，同时提取到的目标轮廓能够准确的反应出目标的特征。因此对后面目标前景的异常动作分析奠定了良好的基础，而且该网络模型的预测时间与传统方法比较短。该模型的主要缺点是需要少量人工参与提取视频帧进行训练，而且随着迭代次数的增加，训练得到的模型越好，同时花费的时间代价也越高。

五、参考文献

[1] Bouwmans T. Traditional and recent approaches in background modeling for foreground detection: An overview. Computer Science Review, s 11–12.pp.31-66,2014.

[2] Lipson H. How transferable are features in deep neural networks. Eprint Arxiv, pp.3320-3328,2014.

[3] Lowe D G. Object Recognition from Local Scale-Invariant Features. IEEE Computer Society, pp.1150,1999.

[4] Wang Y, Luo Z, Jodoin P M. Interactive Deep Learning Method for Segmenting Moving Objects. Pattern Recognition Letters, 2016.

[5] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks.nternational Conference on Neural Information Processing Systems. Curran Associates Inc.pp. 1097-1105,2012

[6] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation.Computer Vision and Pattern Recognition.pp. 3431-3440, 2015.

六、附录

```
MSCNN: based on MatConvNet
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MSCNN ms_regular_training(video,method,frames)
%training network for train dataset
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function regular_training(video, method, frames)

opts.expDir = ['net/' method '/' num2str(frames) '/' video] ;

opts.train.batchSize = 5 ;
opts.train.numEpochs = 20;
```

```
opts.train.continue = false ;
opts.train.useGpu = true ;
opts.train.learningRate    = 1e-3;
opts.train.expDir = opts.expDir ;


%
-------------------------------------------------------------------------
%                                                    Prepare data
%
-------------------------------------------------------------------------


imgDir = ['../basic/' video '/input'];
labelDir = ['../basic/' video '/GT'];
imdb = getImdb(imgDir,labelDir);


mask = imread(['../basic/' video '/ROI.bmp']);
mask = mask(:,:,1);
A = max(max(mask));
mask(mask == A) = 1;
if size(mask,1) > 400 || size(mask,2) >400
    mask = imresize(mask, 0.5, 'nearest');
end
imdb.mask = single(double(mask));

imdb.half_size = 15;

%%%%%%Yi%%%%%% redefined the net
load('net');

net.layers{end-1} = struct('type', 'conv', ...
    'filters', 0.1*randn(1,1,64,1, 'single'), ...
    'biases', zeros(1, 1, 'single'), ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end} = struct('type', 'sigmoidcrossentropyloss');

load('meanPixel.mat');
imdb.meanPixel = meanPixel;

[net,info] = cnn_train_adagrad(net, imdb, @getBatch,...
    opts.train,'errorType','euclideanloss',...
```

```
    'conserveMemory', true);

end

function [im, labels] = getBatch(imdb, batch)
%
--------------------------------------------------------------------

half_size = imdb.half_size;
meanPixel = imdb.meanPixel;

for ii = 1:numel(batch)
    imagename = imdb.images.name{batch(ii)};
    im_ii = single(imread(imagename));

    labelname = imdb.images.labels{batch(ii)};
    roi = imread(labelname);
    labels_ii = zeros(size(roi,1),size(roi,2));
    labels_ii( roi == 50 )  = 0.25;      %shade
    labels_ii( roi == 170 ) = 0.75;      %object boundary
    labels_ii( roi == 255 ) = 1;         %foreground

    % resize the image to half size
    if size(im_ii,1) > 400 || size(im_ii,2) >400
        im_ii = imresize(im_ii, 0.5, 'nearest');
        labels_ii = imresize(labels_ii, 0.5, 'nearest');
    end
    im_large = padarray(im_ii,[half_size,half_size],'symmetric');
    im_ii = bsxfun(@minus, im_large, meanPixel);

    im(:,:,:,ii) = im_ii;
    labels(:,:,1,ii) = labels_ii;
    labels(:,:,2,ii) = double(imdb.mask);

end
end

function imdb = getImdb(imgDir, labelDir)

files = dir([imgDir '/*.jpg']);
label_files = dir([labelDir '/*.jpg']);
names = {};labels = {};

for ii = 1:numel(files)
```

```
    names{end+1} = [imgDir '/' files(ii).name];
    labels{end+1} = [labelDir '/' label_files(ii).name];
end


imdb.images.set = ones(1,numel(names));
imdb.images.name = names ;
imdb.images.labels = labels;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MSCNN ms_regular_testing(video,method,frames)
%testing network for test dataset
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function regular_testing(video, method, frames)

load(['net/' method '/' num2str(frames) '/' video '/net-epoch-20']);

net.layers{end} = struct('name', 'data_hat_sigmoid', ...
    'type', 'sigmoid'        );

net = vl_simplenn_move(net,'gpu');
load meanPixel;

half_size = 15;

imgDir = ['../basic/' video '/test'];
resDir = ['Result/' method '/' num2str(frames) '/' video];

mkdir(resDir);

images = dir([imgDir '/*.jpg']);

for kk = 1 : numel(images)
    imagename = images(kk).name;
    im = single(imread([imgDir '/' imagename]));

    if size(im,1) > 400 || size(im,2) >400
        im = imresize(im, 0.5, 'nearest');
    end

    im_large = padarray(im,[half_size,half_size],'symmetric');
    im_large = bsxfun(@minus, im_large, meanPixel);
    im_large = gpuArray(im_large);
```

```
    %tic;
    A = vl_simplenn(net, im_large);
    B = gather(A(end).x);
    %toc;

    map_im = uint8(B * 255);
    if size(im,1) > 400 || size(im,2) >400
       map_im = imresize(im, [size(im,1),size(im,2)], 'nearest');
    end
    imagename = strrep(imagename, '.jpg', '.png');
    imwrite(map_im,[resDir '/' imagename]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MSCNN cnn_traini_adagrad_ms(net,imdb,getBatch,varargin)
%training core file for cnn
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [net, info] = cnn_train_adagrad_ms(net, imdb, getBatch,
varargin)
% CNN_TRAIN   Demonstrates training a CNN
%    CNN_TRAIN() is an example learner implementing stochastic gradient
%    descent with momentum to train a CNN for image classification.
%    It can be used with different datasets by providing a suitable
%    getBatch function.

opts.train = [] ;
opts.val = [] ;
opts.numEpochs = 300 ;
opts.batchSize = 256 ;
opts.useGpu = false ;
opts.learningRate = 0.001 ;
opts.continue = false ;
opts.expDir = fullfile('data','exp') ;
opts.conserveMemory = false ;
opts.sync = true ;
opts.prefetch = false ;
opts.weightDecay = 0.0005 ;
opts.errorType = 'multiclass' ;
opts.plotDiagnostics = false ;
opts.delta = 1e-8;
opts.display = 1;
opts.snapshot = 1;
opts.test_interval = 1;
```

```
opts = vl_argparse(opts, varargin) ;

if ~exist(opts.expDir, 'dir'), mkdir(opts.expDir); end
if isempty(opts.train), opts.train = find(imdb.images.set==1); end
if isempty(opts.val), opts.val = find(imdb.images.set==2); end
if isnan(opts.train), opts.train = []; end

%
-------------------------------------------------------------------------
----
%                                                    Network initialization
%
-------------------------------------------------------------------------
----

for i=1:numel(net.layers)
    if ~strcmp(net.layers{i}.type,'conv'), continue; end
    net.layers{i}.filtersMomentum =
zeros(size(net.layers{i}.filters), ...
        class(net.layers{i}.filters)) ;
    net.layers{i}.biasesMomentum =
zeros(size(net.layers{i}.biases), ...
        class(net.layers{i}.biases)) ; %#ok<*ZEROLIKE>
    if ~isfield(net.layers{i}, 'filtersLearningRate')
        net.layers{i}.filtersLearningRate = 1 ;
    end
    if ~isfield(net.layers{i}, 'biasesLearningRate')
        net.layers{i}.biasesLearningRate = 1 ;
    end
    if ~isfield(net.layers{i}, 'filtersWeightDecay')
        net.layers{i}.filtersWeightDecay = 1 ;
    end
    if ~isfield(net.layers{i}, 'biasesWeightDecay')
        net.layers{i}.biasesWeightDecay = 1 ;
    end
end

if opts.useGpu
    net = vl_simplenn_move(net, 'gpu') ;
    for i=1:numel(net.layers)
        if ~strcmp(net.layers{i}.type,'conv'), continue; end
        net.layers{i}.filtersMomentum =
gpuArray(net.layers{i}.filtersMomentum) ;
        net.layers{i}.biasesMomentum =
```

```
gpuArray(net.layers{i}.biasesMomentum) ;
    end
end

G_f = cell(numel(net.layers), 1);
G_b = cell(numel(net.layers), 1);

for l=1:numel(net.layers)

    if ~strcmp(net.layers{l}.type, 'conv'), continue ; end

    G_f{l} = zeros(size(net.layers{l}.filters), 'single');
    G_b{l} = zeros(size(net.layers{l}.biases), 'single');

end

%
-------------------------------------------------------------------------
----
%                                                    Train and validate
%
-------------------------------------------------------------------------
----

rng(0) ;

if opts.useGpu
    one = gpuArray(single(1)) ;
else
    one = single(1) ;
end

info.train.objective = [] ;
info.train.error = [] ;
info.train.topFiveError = [] ;
info.train.speed = [] ;
info.val.objective = [] ;
info.val.error = [] ;
info.val.topFiveError = [] ;
info.val.speed = [] ;

lr = opts.learningRate ;
res = [] ;
for epoch=1:opts.numEpochs
```

```
    % fast-forward to where we stopped
    modelPath = @(ep) fullfile(opts.expDir, sprintf('net-epoch-%d.mat',
ep));
    modelFigPath = fullfile(opts.expDir, 'net-train.pdf') ;
    if opts.continue
        if exist(modelPath(epoch),'file')
            if epoch == opts.numEpochs
                load(modelPath(epoch), 'net', 'info') ;
            end
            continue ;
        end
        if epoch > 1
            fprintf('resuming by loading epoch %d\n', epoch-1) ;
            load(modelPath(epoch-1), 'net', 'info') ;
        end
    end

    train = opts.train(randperm(numel(opts.train))) ;
    val = opts.val ;

    info.train.objective(end+1) = 0 ;
    info.train.error(end+1) = 0 ;
    info.train.topFiveError(end+1) = 0 ;
    info.train.speed(end+1) = 0 ;
    info.val.objective(end+1) = 0 ;
    info.val.error(end+1) = 0 ;
    info.val.topFiveError(end+1) = 0 ;
    info.val.speed(end+1) = 0 ;

    for t=1:opts.batchSize:numel(train)
        % get next image batch and labels
        batch = train(t:min(t+opts.batchSize-1, numel(train))) ;
        batch_time = tic ;
        fprintf('training: epoch %02d: processing batch %3d of %3d ...',
epoch, ...
            fix(t/opts.batchSize)+1,
ceil(numel(train)/opts.batchSize)) ;
        [im_ori, labels_ori] = getBatch(imdb, batch) ;
        if opts.prefetch
            nextBatch = train(t+opts.batchSize:min(t+2*opts.batchSize-1,
numel(train))) ;
            getBatch(imdb, nextBatch) ;
        end
```

```
        for ss = 1 : numel(imdb.scales)

            [im, labels] = rescale_im(im_ori, labels_ori,
imdb.scales(ss),...
                imdb.mask, imdb.half_size, imdb.meanPixel);

            if opts.useGpu
                im = gpuArray(im) ;
            end

            % backprop
            net.layers{end}.class = labels ;
            res = vl_simplenn(net, im, one, res, ...
                'conserveMemory', opts.conserveMemory, ...
                'sync', opts.sync) ;

            % gradient step
            for l=1:numel(net.layers)
                if ~strcmp(net.layers{l}.type, 'conv'), continue ; end

                g_f = (net.layers{l}.filtersLearningRate) * ...
                    (opts.weightDecay * net.layers{l}.filtersWeightDecay)
* net.layers{l}.filters + ...
                    (net.layers{l}.filtersLearningRate) / numel(batch) *
res(l).dzdw{1};
                g_b = (net.layers{l}.biasesLearningRate) * ...
                    (opts.weightDecay * net.layers{l}.biasesWeightDecay) *
net.layers{l}.biases + ...
                    (net.layers{l}.biasesLearningRate) / numel(batch) *
res(l).dzdw{2};

                G_f{l} = G_f{l} + g_f .^ 2;
                G_b{l} = G_b{l} + g_b .^ 2;

                net.layers{l}.filters = net.layers{l}.filters - lr ./
(opts.delta + sqrt(G_f{l})) .* g_f;
                net.layers{l}.biases  = net.layers{l}.biases - lr ./
(opts.delta + sqrt(G_b{l})) .* g_b;
            end
        end

        % print information
        batch_time = toc(batch_time) ;
```

```
        speed = numel(batch)/batch_time ;
        info.train = updateError(opts, info.train, net, res, batch_time) ;

        fprintf(' %.2f s (%.1f images/s)', batch_time, speed) ;
        n = t + numel(batch) - 1 ;
        switch opts.errorType
            case 'multiclass'
                fprintf(' err %.1f err5 %.1f', ...
                    info.train.error(end)/n*100,
info.train.topFiveError(end)/n*100) ;
                fprintf('\n') ;
            case 'binary'
                fprintf(' err %.1f', ...
                    info.train.error(end)/n*100) ;
                fprintf('\n') ;
            case 'euclideanloss'
                fprintf(' err %.1f', info.train.error(end) / n);
                fprintf('\n') ;
        end

        % debug info
        if opts.plotDiagnostics
            figure(2) ; vl_simplenn_diagnose(net,res) ; drawnow ;
        end
    end % next batch


    % evaluation on validation set
    if epoch == 1 || rem(epoch, opts.test_interval) == 0 || epoch ==
opts.numEpochs
        for t=1:opts.batchSize:numel(val)
            batch_time = tic ;
            batch = val(t:min(t+opts.batchSize-1, numel(val))) ;
            fprintf('validation: epoch %02d: processing batch %3d
of %3d ...', epoch, ...
                fix(t/opts.batchSize)+1,
ceil(numel(val)/opts.batchSize)) ;
            [im, labels] = getBatch(imdb, batch) ;
            if opts.prefetch
                nextBatch = val(t+opts.batchSize:min(t+2*opts.batchSize-1,
numel(val))) ;
                getBatch(imdb, nextBatch) ;
            end
            if opts.useGpu
```

```
            im = gpuArray(im) ;
        end

        net.layers{end}.class = labels ;
        res = vl_simplenn(net, im, [], res, ...
            'disableDropout', true, ...
            'conserveMemory', opts.conserveMemory, ...
            'sync', opts.sync) ;

        % print information
        batch_time = toc(batch_time) ;
        speed = numel(batch)/batch_time ;
        info.val = updateError(opts, info.val, net, res, batch_time) ;

        fprintf(' %.2f s (%.1f images/s)', batch_time, speed) ;
        n = t + numel(batch) - 1 ;
        switch opts.errorType
            case 'multiclass'
                fprintf(' err %.1f err5 %.1f', ...
                    info.val.error(end)/n*100,
info.val.topFiveError(end)/n*100) ;
                fprintf('\n') ;
            case 'binary'
                fprintf(' err %.1f', ...
                    info.val.error(end)/n*100) ;
                fprintf('\n') ;
            case 'euclideanloss'
                fprintf(' err %.1f', info.val.error(end) / n);
                fprintf('\n') ;
        end
    end
end

% save
info.train.objective(end) = info.train.objective(end) /
numel(train) ;
info.train.error(end) = info.train.error(end) / numel(train)  ;
info.train.topFiveError(end) = info.train.topFiveError(end) /
numel(train) ;
info.train.speed(end) = numel(train) / info.train.speed(end) ;
info.val.objective(end) = info.val.objective(end) / numel(val) ;
info.val.error(end) = info.val.error(end) / numel(val) ;
info.val.topFiveError(end) = info.val.topFiveError(end) /
numel(val) ;
```

```
    info.val.speed(end) = numel(val) / info.val.speed(end) ;
    if epoch == 1 || rem(epoch, opts.snapshot) == 0 || epoch ==
opts.numEpochs
        save(modelPath(epoch), 'net', 'info') ;
    end

    if epoch == 1 || rem(epoch, opts.display) == 0 || epoch ==
opts.numEpochs
        figure(1) ; clf ;
        subplot(1,2,1) ;
        semilogy(1:epoch, info.train.objective, 'k') ; hold on ;
        semilogy([1 opts.test_interval : opts.test_interval : epoch epoch],
info.val.objective([1 opts.test_interval : opts.test_interval : epoch
epoch]), 'b') ;
        xlabel('training epoch') ; ylabel('energy') ;
        grid on ;
        h=legend('train', 'val') ;
        set(h,'color','none');
        title('objective') ;
        subplot(1,2,2) ;
        switch opts.errorType
            case 'multiclass'
                plot(1:epoch, info.train.error, 'k') ; hold on ;
                plot(1:epoch, info.train.topFiveError, 'k--') ;
                plot([1 opts.test_interval : opts.test_interval : epoch
epoch], info.val.error([1 opts.test_interval : opts.test_interval :
epoch epoch]), 'b') ;
                plot([1 opts.test_interval : opts.test_interval : epoch
epoch], info.val.topFiveError([1 opts.test_interval :
opts.test_interval : epoch epoch]), 'b--') ;
                h=legend('train','train-5','val','val-5') ;
            case 'binary'
                plot(1:epoch, info.train.error, 'k') ; hold on ;
                plot([1 opts.test_interval : opts.test_interval : epoch
epoch], info.val.error([1 opts.test_interval : opts.test_interval :
epoch epoch]), 'b') ;
                h=legend('train','val') ;
            case 'euclideanloss'
                plot(1 : epoch, info.train.error, 'k'); hold on;
                plot([1 opts.test_interval : opts.test_interval : epoch
epoch], info.val.error([1 opts.test_interval : opts.test_interval :
epoch epoch]), 'b') ;
                h = legend('train', 'val') ;
        end
```

```
        grid on ;
        xlabel('training epoch') ; ylabel('error') ;
        set(h,'color','none') ;
        title('error') ;
        drawnow ;
        print(1, modelFigPath, '-dpdf') ;
    end

end
end

%
-------------------------------------------------------------------------
----
function info = updateError(opts, info, net, res, speed)
%
-------------------------------------------------------------------------
----
predictions = gather(res(end-1).x) ;
sz = size(predictions) ;
n = prod(sz(1:2)) ;

labels = net.layers{end}.class ;
info.objective(end) = info.objective(end) +
sum(double(gather(res(end).x))) ;
info.speed(end) = info.speed(end) + speed ;
switch opts.errorType
    case 'multiclass'
        [~,predictions] = sort(predictions, 3, 'descend') ;
        error = ~bsxfun(@eq, predictions, reshape(labels, 1, 1, 1, [])) ;
        info.error(end) = info.error(end) +....
            sum(sum(sum(error(:,:,1,:))))/n ;
        info.topFiveError(end) = info.topFiveError(end) + ...
            sum(sum(sum(min(error(:,:,1:5,:),[],3))))/n ;
    case 'binary'

        labels = labels(:,:,1,:);
        [~,predictions] = sort(predictions, 3, 'descend') ;

        predictions = predictions(:,:,1,:);
        error = ~bsxfun(@eq, predictions, labels) ;
        info.error(end) = info.error(end) + sum(error(:))/n ;
    case 'euclideanloss'
        error = euclideanloss(sigmoid(predictions), labels);
```

```
        info.error(end) = info.error(end) + error;
end
end


function [im, labels] = rescale_im(im_ori, label_ori, scale, mask,
half_size, meanPixel)

mask = imresize(mask, scale, 'nearest');

for i = 1:size(im_ori,4)
    im_ii = imresize(im_ori(:,:,:,i), scale, 'nearest');
    im_large = padarray(im_ii, [half_size, half_size], 'symmetric');
    im_ii = bsxfun(@minus, im_large, meanPixel);

    label_ii = imresize(label_ori(:,:,:,i), scale, 'nearest');


    im(:, :, :, i) = im_ii;
    labels(:, :, 1, i) = label_ii;
    labels(:, :, 2, i) = double(mask);

end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MSCNN Y=sigmoidcrossentropyloss(X,c,dzdy)
% sigmoidcrossentropyloss function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Y = sigmoidcrossentropyloss(X, c, dzdy)
%EUCLIDEANLOSS Summary of this function goes here
%   Detailed explanation goes here

%assert(numel(X) == numel(c));

sz = [size(X,1) size(X,2) size(X,3) size(X,4)] ;

if numel(c) == sz(4)
  % one label per image
  c = reshape(c, [1 1 1 sz(4)]) ;
end
if size(c,1) == 1 & size(c,2) == 1
  c = repmat(c, [sz(1) sz(2)]) ;
end
```

```
% one label per spatial location
sz_ = [size(c,1) size(c,2) size(c,3) size(c,4)] ;
assert(isequal(sz_, [sz(1) sz(2) sz_(3) sz(4)])) ;
assert(sz_(3)==1 | sz_(3)==2) ;

% Getting the weights
mass = double(c(:,:,1,:));
mass(:) = 1;
if sz_(3) == 2
  % the second channel of c (if present) is used as weights
  mass = double(c(:,:,2,:));
  c(:,:,2,:) = [] ;
end

%p    = sigmoid(c);
p_hat = sigmoid(X);

%sz = [size(X,1),size(X,2),size(X,3),size(X,4)];
%n = sz(1) * sz(2);

if nargin == 2 || isempty(dzdy)

    Y = -(c .* log(p_hat) + (1-c)*log(1-p_hat)).*mass;
    Y = sum(Y(:));

%    Y = -sum(subsref(c * log(p_hat) + (1 - c) * log(1 - p_hat),
substruct('()', {':'}))); % Y is divided by d(4) in cnn_train.m /
cnn_train_mgpu.m.
%    Y = -1 / prod(d(1 : 3)) * sum(subsref(p * log(p_hat) + (1 - p) * log(1
- p_hat), substruct('()', {':'}))); % Should Y be divided by prod(d(1 :
3))? It depends on the learning rate.

elseif nargin == 3 && ~isempty(dzdy)

    assert(numel(dzdy) == 1);

    Y = dzdy * (p_hat - c) .* mass; % Y is divided by d(4) in cnn_train.m
/ cnn_train_mgpu.m.
    %Y = dzdy / n * (p_hat - c); % Should Y be divided by prod(d(1 : 3))?
It depends on the learning rate.

end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MSCNN y=sigmoid (x,dzdy)
% sigmoid function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = sigmoid(x, dzdy)
%SIGMOID Summary of this function goes here
%   Detailed explanation goes here

y = 1 ./ (1 + exp(-x));

if nargin == 2 && ~isempty(dzdy)

    assert(all(size(x) == size(dzdy)));

    y = dzdy .* y .* (1 - y);

end

end
```