中国矿业大学计算机学院

2019 级本科生课程大作业

课程名称 《计算智能》

报告时间 2022.04.30

学生姓名 王杰永

学　号 03190886

专　业 计算机科学与技术

任课教师 刘佰龙

《计算智能》课程大作业评分表

姓名：王杰永　学号：03190886　班级：计科 19-3 班

序号	毕业要求	课程教学目标	考查方式与考查点	占比	得分点	应得分	实际得分	目标汇总
1	1.4	**目标 1：**了解计算智能在实际应用领域中的应用背景、现状以及趋势。	针对某个实际问题，能够总结适用该问题计算智能方法的背景和应用领域。	20%	0	10		
					1	5		
					8	5		
2	2.1	**目标 2：**能够将实际问题抽象成数学模型，并通过具体的计算智能方法原理分析，对此模型进行求解。	针对具体问题，抽象出数学模型，并能够求解。	40%	2	15		
					3	10		
					9	15		
3	4.3	**目标 3：**根据所选计算智能方法，对实际问题对应的模型求解方案加以实验验证，并对方法的有效性、合理性以及优缺点进行分析，并与其他方法进行对比，得到有效结论。	对所选方法以仿真方式进行实现，并分析方法的优缺点，与其他方法对比，得出结论。	40%	4	15		
					5	10		
					6	5		
					7	10		
总分				100%	-	100		

得分点编号具体内容见考核要求。

关于 U-Net 网络结构的改进
以及神经网络可视化的研究

摘要：

深度学习目前已成为发展最快、最令人兴奋的机器学习领域之一。特别是在计算机视觉领域的应用，加速了计算机视觉的发展。语义分割是指将图像中的每个像素链接到类标签的过程，即像素级别的图像分类。语义分割在自动驾驶、医学影像、机器人技术等多领域均有应用。

本文在上述背景下，对传统的语义分割网络 U-Net 的结构进行优化，给出了**改进的 U-Net** 网络结构，解决了包括传统 U-Net 网络的输出与输入图像尺寸不匹配在内的多个问题。同时，网络的上采样部分使用**双线性插值**算法替代原网络中的转置卷积，减少了模型的参数数量，提高了模型的速度。最后，基于 PASCAL VOC 数据集完成了网络的训练，在测试集上的 MIOU 提升了 4%。

同时，本文通过使用 **Grad-CAM** 对被视作“黑盒”的神经网络可视化，展示了网络在做出预测结果时所依赖的原图像中的关键性特征，最后以热力图的形式可视化展示了卷积神经网络在特征提取上的有效性。

关键词：语义分割；改进的 U-Net 网络；双线性插值；Grad-CAM

A research on improving U-Net structure and the visualization of neural network

Abstract:

Deep learning is currently one of the fastest growing and most exciting fields of machine learning. Especially in the field of computer vision, deep learning accelerates the development of computer vision. Semantic segmentation refers to the process of linking each pixel in an image to a certain class label, that is, pixel-level image classification. Semantic segmentation is used in many fields such as autonomous driving, medical imaging, and robotics.

Under the above background, we optimizes the structure of the traditional semantic segmentation network U-Net, and presents an **improved U-Net network structure**, which solves the problem that the output of the traditional U-Net network does not match the size of the input image. At the same time, the **bilinear interpolation algorithm** is used in the upsampling part to replace the transposed convolution in the original network, which reduces the number of parameters of the model and improves the speed of the model. Finally, the network training is completed based on the PASCAL VOC dataset, and the MIOU on the test set is improved by 4%.

By using **Grad-CAM** to visualize the neural network which is regarded as a "black box", it shows the key features in the original image that the network relies on when the predictions results are shown, and finally the effectiveness of convolutional neural network in feature extraction is demonstrated in the form of a heat map.

KeyWords: Semantic Segmentation; Improved U-Net Structure; Bilinear Interpolation; Grad-CAM

目录

1 问题描述

计算机视觉旨在识别和理解图像/视频中的内容。经过多年的发展，目前计算机视觉研究的方向包括图像分类(Image Classification)、目标定位(Object Localization)、目标检测(Object Detection)、语义分割(Semantic Segmentation)等。

同时，基于卷积神经网络的特征自动提取虽然取得了不错的效果，但其提取到的特征和规则很难用人类理解的方式来呈现（相对于传统机器学习算法，例如决策树或者逻辑回归等），因此被很多人认为是“黑盒”。图像中哪些位置的像素对输出有着强烈的影响成为一个研究的热点问题。

本文基于 Pascal VOC 数据集，使用改进的 U-Net 网络，成功训练了目标检测模型。并使用 Grad-CAM 算法，探究 U-Net 网络预测结果的形成原因，使用热力图的形式可视化展示了网络的输出结果对输入图像敏感像素的位置。

1.1 问题背景

图像分类

图像分类是计算机视觉基本任务之一。顾名思义,图像分类即给定一幅图像,计算机利用算法找出其所属的类别标签。图像分类的过程主要包括图像的预处理、图像的特征提取以及使用分类器对图像进行分类。传统的图像分类算法提取图像的色彩、纹理和空间等特征(孙君顶和赵珊,2009),其在简单的图像分类任务中表现较好,但在复杂图像分类任务中表现不尽人意。在 2012 年之后，Alex Net 大幅提升了图像分类任务的性能，使深度卷积神经网络逐渐成为计算机视觉领域研究的热点[1]。

从早期的 LeNet 到 GoogleNet 与 VGG，再到 ResNet，分类网络的发展十分迅速。同时，也为视觉领域的其他任务提供了各种高性能的 backbone。可以说，分类任务是视觉领域的基石。

目标定位

在图像分类的基础上，我们还想知道图像中的目标具体在图像的什么位置，通常是以包围盒的(bounding box)形式呈现。

目标检测

在目标定位中，通常只有一个或固定数目的目标，而目标检测更一般化，其图像中出现的目标种类和数目都不定。因此，目标检测是比目标定位更具挑战性的任务。

目标检测算法可以分为两类——两阶段(two-stage)模型与单阶段(one-stage)模型。RNN、FAST R-CNN、FASTER R-CNN 等是基于候选区域(region proposal)的两阶段目标检测算法，准确度高但两阶段限制了其检测速度，所以通常检测速度较慢；而 yolo v1-v5、SDD 等单阶段目标检测算法，虽然检测精度略低于两阶段，但其检测速度远胜于 R-CNN 系列。

语义分割

语义分割结合了图像分类、目标检测和图像分割，通过一定的方法将图像分割成具有一定语义含义的区域块，并识别出每个区域块的语义类别，实现从

底层到高层的语义推理过程，最终得到一幅具有逐像素语义标注的分割图像。

图像语义分割方法有传统方法和基于卷积神经网络的方法，其中传统的语义分割方法又可以分为基于统计的方法和基于几何的方法。随着深度学习的发展，语义分割技术得到很大的进步，基于卷积神经网络的语义分割方法与传统的语义分割方法最大不同是，网络可以自动学习图像的特征，进行端到端的分类学习，大大提升语义分割的精确度。

1.2 问题重述

PASCAL VOC 挑战赛是视觉对象的分类识别和检测的一个基准测试，提供了检测算法和学习性能的标准图像注释数据集和标准的评估系统，主要有 Classification、Detection、Segmentation、Person Layout、Action Classification 这几类子任务，一共有 20 个类别，包括：

person

bird, cat, cow, dog, horse, sheep

aeroplane, bicycle, boat, bus, car, motorbike, train

bottle, chair, dining table, potted plant, sofa, tv/monitor

本文从 VOC2007 至 VOC2011 年的数据集中挑选部分训练/测试数据，构成了 VOC 拓展数据集，进行语义分割模型的训练。其中，训练集共 10582 张 RGB 图片及其对应的分割标注灰度图片，测试集共 1449 张图片。

同时，为了解决 U-Net 网络输入图像与输出图像尺寸不一致问题，本文采用了改进的 U-Net 结构，解决了上述问题。

为了探究神经网络的可解释性，本文采用 Grad-CAM 算法，使用热力图的形式，对 U-Net 网络的输出结果进行了可视化展示。

2 算法以及网络结构优化

2.1 U-Net 网络结构及改进

U-Net 发表于 2015 年，属于 FCN(Fully Convolutional Networks)的一种变体。U-Net 的初衷是为了解决生物医学图像方面的问题。U-Net 网络在测试集上表现良好，因此被广泛的应用在语义分割的各个方向，比如卫星图像分割，工业瑕疵检测等。

U-Net 与 FCN 都是 Encoder-Decoder 结构，结构简单但很有效。Encoder 负责特征提取，Decoder 负责对提取到的特征解码，还原回输入图像相同的尺寸。

2.1.1 U-Net 网络结构

U-Net 网络提出时，其网络结构如图 1 所示[2]。

蓝/白色框表示$feature\ map$；根据注解，蓝色箭头表示卷积核大小为 3×3 的不带有$padding$的卷积层，随后紧跟 $ReLU$ 激活函数。由于不带有$padding$，因此通过一层卷积会使得输入特征图（或原图像）的宽和高减少 2；红色箭头表示2×2的最大池化下采样层，当特征图通过池化层后，特征图通道数加倍，

但宽高缩小为原来的一半。

绿色箭头表示卷积核大小为2×2的转置卷积层，特征图通过后，其通道数减半，但宽高变为输入特征图的两倍，从而完成了上采样的作用。

灰色箭头表示裁剪与拼接。特征图通过不断的下采样难免会丢失部分细节信息，这部分信息是无法通过转置卷积上采样恢复的。将上采样后的特征图与对应下采样倍率的特征图进行拼接，从而保留图像的细节信息，使模型的表现更好。

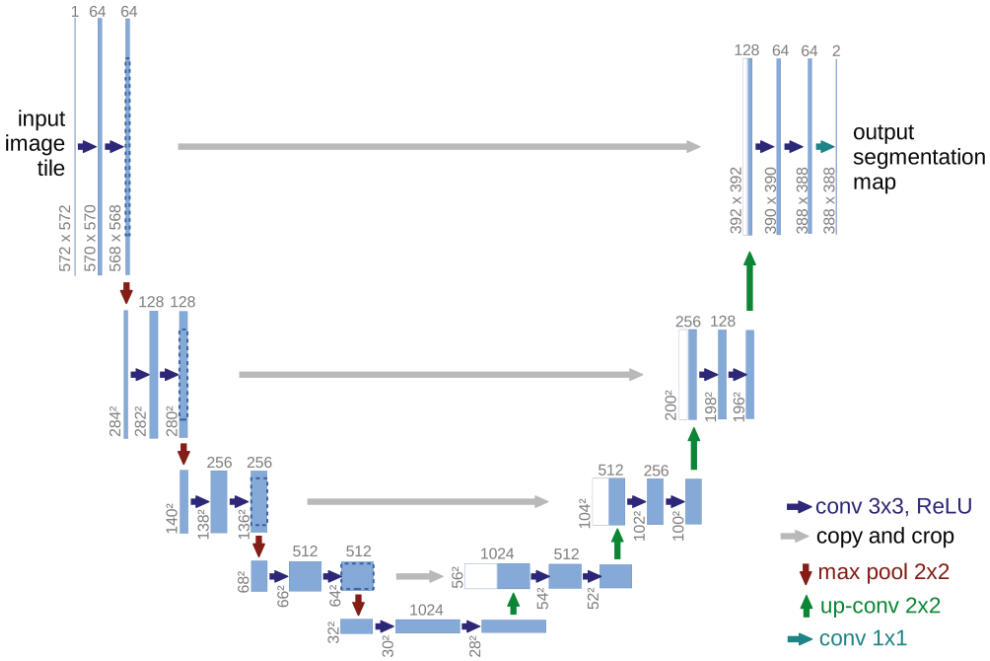

图 1 原始 U-Net 网络结构

由于 U-Net 没有使用全连接层，因此最终对像素点的逐一分类是通过1×1的卷积层实现的。这一点与 FCN 网络类似。全部由卷积操作实现的语义分割网络，可以适配输入图像的任意尺寸，这一点是含有全连接层的网络所不能够的。

不过，作者 2015 年提出的 U-Net 网络用现在的眼光看，存在着许多不足之处。

2.1.2 U-Net 网络的不足

在图 1 中可以看出，由于提取特征的卷积层$padding$设置为0，因此每一次通过卷积层得到的特征图尺寸相较于原图都会减小。进而导致最终上采样部分无法恢复原图大小。主流的语义分割网络的输出与输入图像尺寸一致，因此可以很好的对原图像的每一个像素点分类。而 U-Net 则违背了这一原则。当输入图像为572×572时，网络的输出尺寸为388×388。

为了解决这一问题，U-Net 作者简单的将原图像裁剪，使用原图像中心388×388的部分图像与网络输出配合，最终得到输入图像的部分语义分割结果。如图 2 所示。

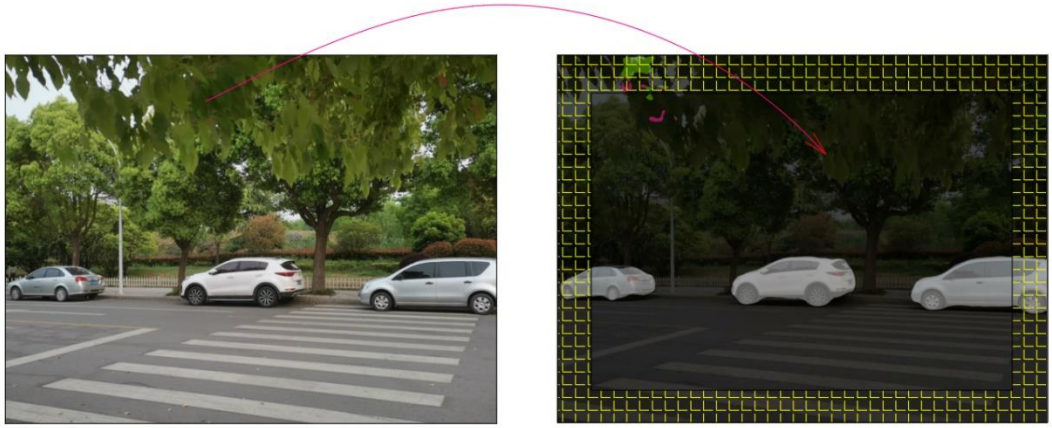

图 2 U-Net 网络的裁剪策略。以输入图像的中心为参照，仅保留部分图像，得到的语义分割结果（右图中心黑色区域）无法展现输入图像的全部。

当然，作者通过提出 Overlap-tile 策略，以空间和时间为代价，填补了网络输出对于输入的空缺部分。但如果在特征提取时，为卷积层加入$padding$，从而使通过卷积层后特征图的尺寸不变，无需进行裁剪便可以得到与输入图相同大小的网络输出了。

2.1.3 改进的 U-Net 结构

首先，解决 2.1.2 中的输入输出图像尺寸不一致的问题。我们对原始网络结构中的所有卷积层添加$padding = 1$，从而使特征图经过卷积层后尺寸不发生变化，进而解决了输入输出不匹配的问题。

U-Net 的网络结构决定了它很好的结合了图像的底层与高层信息。医学图像边界模糊、梯度复杂，需要较多的高分辨率信息；人体结构相对固定，分割目标在人体图像中的分布很有规律，语义信息简单明确，低分辨率信息很好的提供了这一信息，用于目标物体的识别。因此，对于医学图像，低级空间信息和高级语义信息都很重要；此外，医学影像的数据获取较为困难，人工标注不仅需要专业人员指导，还费时费力，且存在主观差异。因此，在医学图像分割领域中，所设计的网络模型很容易出现过拟合现象。而 U-Net 则可以和那后的解决这些问题。因此 U-Net 在医学图像领域有着很好的表现[3]。

对于自然图像而言，使用 U-Net 的特征提取网络（编码器部分）固然可以得到比较好的结果，但受限于算力以及迁移学习的影响，我们可以利用在 PASCAL VOC 数据集上已经存在的预训练权重来初始化网络。在这样一组与训练权重参数的基础上去训练我们的解码器部分，会使得网络表现大大提升。

为了使特征提取网络可以使用预训练权重，我们必须更改编码器结构——使用目标检测或语义分割常用的$backbone$，VGG 系列或 ResNet 系列。

使用 ResNet 系列的$backbone$可以有效解决深度网络中的梯度消失或梯度爆炸问题。通过引入特定的残差模块，网络的深度可以大大加深，从而提升网络表现。但**受限于算力问题**，很难训练如此深层次的网络结构。最终，本文选择了 VGG16 作为特征提取网络（编码器）的$backbone$骨架网络[4]。

VGG16 的网络结构如图 3 所示。

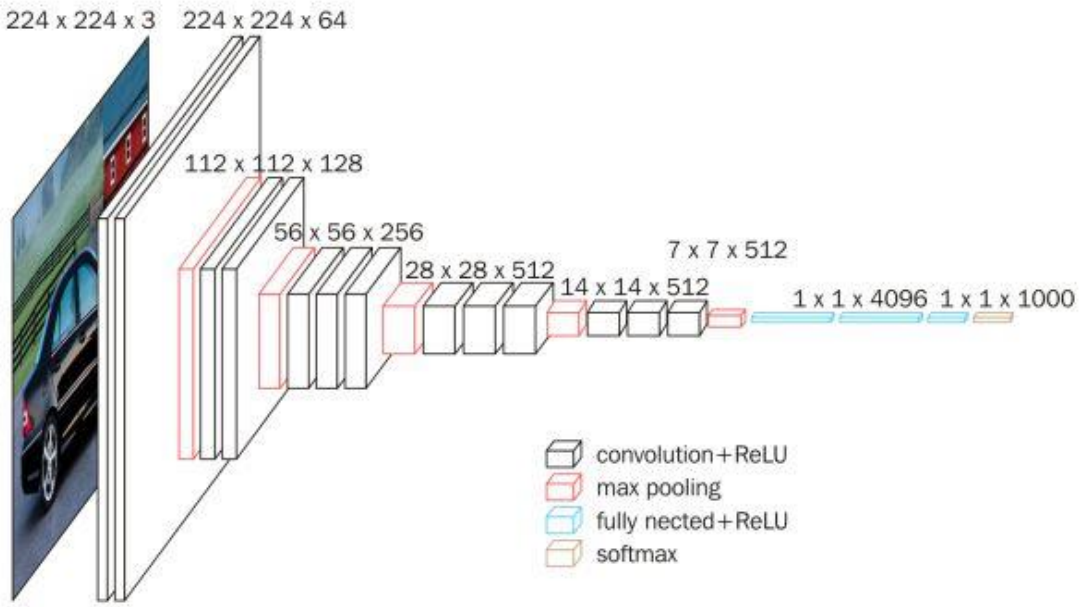

图 3 VGG16 网络结构图

去除掉最后的$softmax$层与 3 个全连接层，剩余卷积层+最大池化下采样层的组合可以作为 U-Net 的$backbone$。可以看出，VGG16 经过 5 个最大池化下采样层，使得输入图像下采样 32 倍。因此，在 U-Net 解码器部分我们可以通过 5 个 2 倍率的上采样层，将网络输出还原为原图尺寸大小。

接下来考虑上采样问题。

目前，常用的上采样方式主要有两种：双线性插值以及转置卷积。U-Net 网络中作者采用转置卷积进行上采样。尽管转置卷积在图像分割领域中应用广泛，但其带有需要学习的参数，这在算力不足的时候难以实现。双线性插值算法则不存在需要学习的参数，且前向传播的速度远大于转置卷积。综合考虑，本文采用双线性插值算法替代原始结构中的转置卷积完成上采样。

2.2 双线性插值算法

插值是指通过一系列已知的数据点，来"猜测"未知点的像素值。当我们需要对图像上采样一倍时，图像宽高方向的像素个数分别增加一倍，即生成了若干未知点。这些未知点的像素值就可以通过双线性插值确定。

2.2.1 单线性插值

单线性插值是通过距离未知点最近的两个点的像素值推断未知点像素值的。

如图 4 所示，已知点$(x_0, y_0), (x_1, y_1)$，要得到某一未知点(x, y)在位置x处的y值，我们可以通过下式计算：

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \tag{1}$$

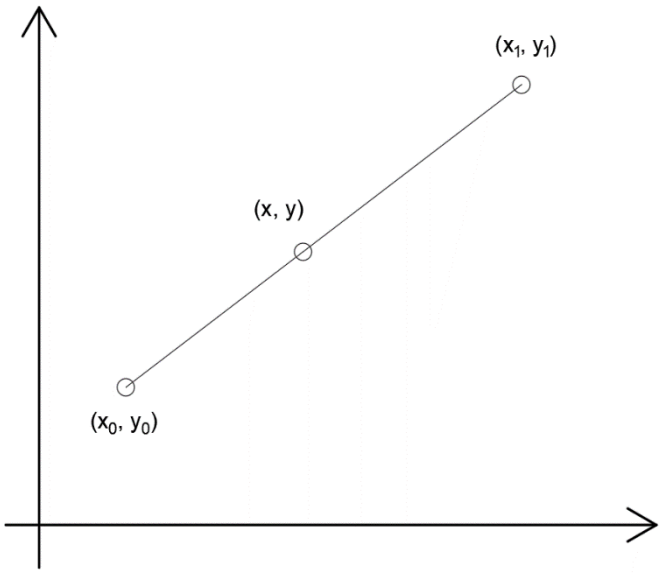

图 4 单线性插值

整理后得到：

$$y = \frac{x_1 - x}{x_1 - x_0} y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \tag{2}$$

从式(2)中可以看出，y_0与y_1的系数是x分别到x_1和x_2的距离，即使用x方向的距离作为加权系数。

单线性插值的本质是在x方向做了一次线性插值。使用$f(p)$表示像素点p的像素值，则单线性插值公式如下：

$$f(p) = \frac{x_1 - x}{x_1 - x_0} f(p_0) + \frac{x - x_0}{x_1 - x_0} f(p_1) \tag{3}$$

2.2.2 双线性插值

双线性插值是指对x方向与y方向分别进行插值，总共进行3次单线性插值。双线性插值通过距离未知点最近的四个点的像素值推断未知点像素值。

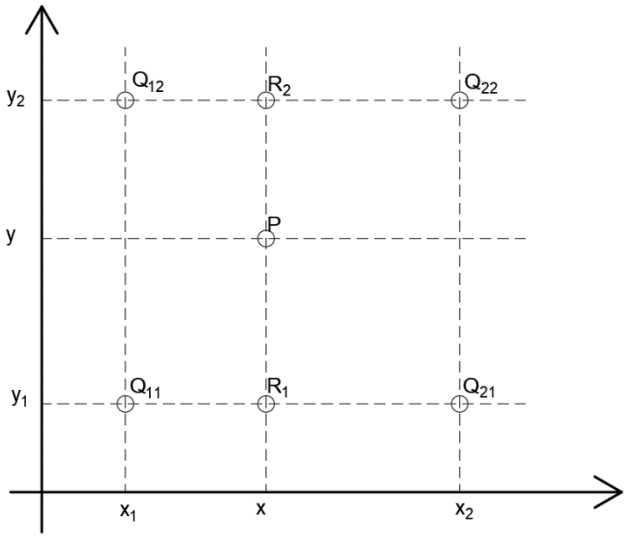

图 5 双线性插值

我们使用点$Q_{11}(x_1,y_1),Q_{12}(x_1,y_2),Q_{21}(x_2,y_1),Q_{22}(x_2,y_2)$作为已知点，点$P(x,y)$作为未知点。函数$f$为点的像素值。

双线性插值算法描述如下：

利用单线性插值在x方向求$f(R_1)$：

$$f(R_1)=\frac{x_2-x}{x_2-x_1}f(Q_{11})+\frac{x-x_1}{x_2-x_1}f(Q_{21}) \tag{4}$$

利用单线性插值在y方向求$f(R_2)$:

$$f(R_2)=\frac{x_2-x}{x_2-x_1}f(Q_{12})+\frac{x-x_1}{x_2-x_1}f(Q_{22}) \tag{5}$$

利用单线性插值在y方向求$f(P)$:

$$f(P)=\frac{y_2-y}{y_2-y_1}f(R_1)+\frac{y-y_1}{y_2-y_1}f(R_2) \tag{6}$$

从而求得未知点P的像素值$f(P)$。

2.3 Grad-CAM

对于深度学习网络，普遍认为是个可解释性很低的“黑盒”模型：模型为什么会这么预测、模型关注的主要特征在哪里，这些问题都比较难以解释。

Grad-CAM(Gradient-weighted Class Activation Mapping)算法借助优化算法的梯度反向传播，探究网络预测结果的形成原因，并使用热力图的形式可视化展示网络输出结果受到输入图像不同区域的影响程度大小。

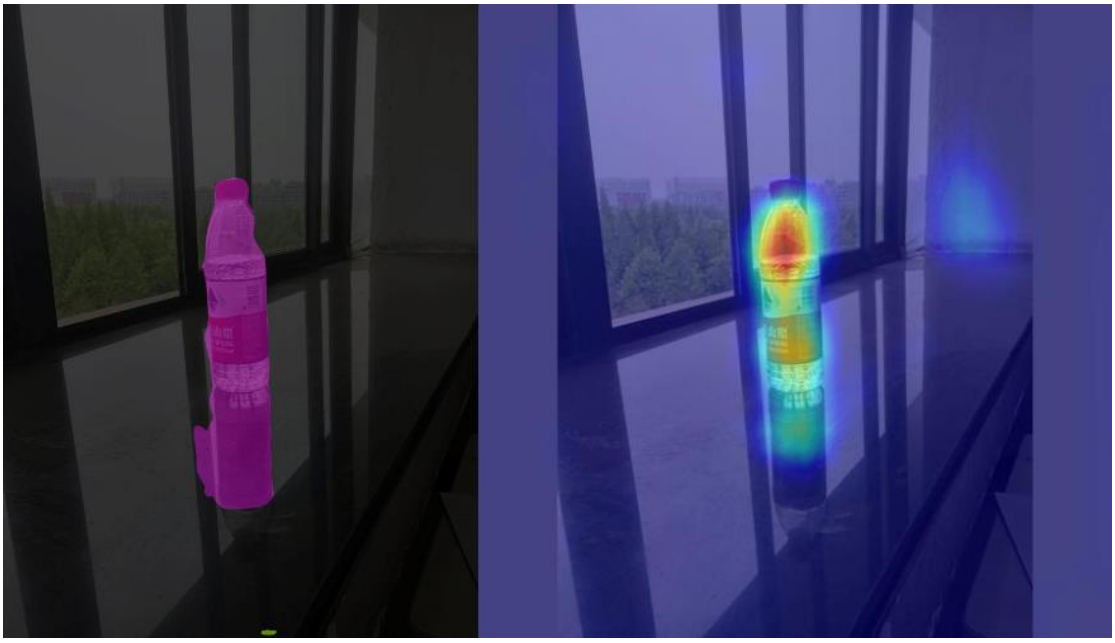

图 6 语义分割结果（左）并使用 Grad-CAM 展示网络

最深的特征图对于类别*bottle*所关注的区域（右）

Grad-CAM 是 CAM(Class Activation Mapping)的改进，相比于 CAM 更具一般性。CAM 一个致命的问题是需要修改网络结构并且重新训练后才能可视化展示热力图，而 Grad-CAM 完美解决了这一问题。

2.3.1 算法原理

Grad-CAM 算法是针对分类问题的[5]，并没有给出语义分割中算法如何应用。首先以分类任务为例，介绍 Grad-CAM 的原理。

分类任务的网络结构通常由负责特征提取的卷积层以及最终负责预测的全连接层组成。网络的输入通过一系列卷积层后，会得到表征输入图像特征的若干特征图。通过对最深的特征图展平，后接全连接层从而得到网络的预测结果。

通常认为，特征提取部分最深的特征图包含了我们所感兴趣的目标（网络分类结果类别）的众多语义信息，Grad-CAM 算法认为，通过对网络预测类别反向传播，得到反传回最深特征层的梯度信息，梯度越大则表示对预测结果的贡献越大——即梯度信息反映了众多语义信息的对结果影响大小的权重。

使用A表示某一特征层，k表示特征层A的第k个通道，c代表类别，A^k代表特征层A中通道k的数据，α_k^c代表特征图通道k的权重，则 Grad-CAM 算法的输出为：

$$L_{Grad-CAM}^{c} = ReLU\left(\sum_{k} \alpha_k^c A^k\right) \tag{7}$$

关于α_k^c的计算公式如下：

$$\alpha_k^c = \frac{1}{Z}\sum_{i}\sum_{j}\frac{\partial y^c}{\partial A_{ij}^k} \tag{8}$$

其中，y^c是网络针对类别c的预测分数，A_{ij}^k是特征层A的通道k中，坐标为(i,j)的数据，Z是特征层的宽×高。

式(7)表明，对特征层A各个通道基于预测结果反传梯度作为权重加权求和后，通过$ReLU$激活函数，得到了 Grad-CAM 的输出。由于输出与特征图的尺寸相同，我们对算法输出进行双线性插值上采样后，可以还原回输入图像的大小，进而可以对算法输出以热力图的方式可视化展示了。

最后，尽管 Grad-CAM++[6]等众多 CAM 系列的算法陆续被提出，但其效果在可视化展示的层面而言相差不多，因此本文选择 Grad-CAM 算法。

2.3.2 改进算法，应用到语义分割中

Grad-CAM 算法的核心是得到最终预测类别c对特征层A的梯度。

分类问题中，网络对类别c的输出值是维度为 1 的张量（数量）；而在语义分割问题中，网络对类别c的输出则是一张与输入图相同尺寸的图。为了将算法应用到语义分割问题中，本文将语义分割对类别c的输出（维度为 3 的张量）转换为对类别c的维度为 1 的张量，从而可以应用原始 Grad-CAM 算法绘制热力图了。

最简单的思想便是将网络对类别c的输出 3 维张量求和，得到对类别c的 1 维张量。但存在一些问题。

我们关注类别c，而原图像中并非所有像素都被分类为c。因此，我们根据网络的预测结果制作$mask$，将输入图像二值化——我们感兴趣类别对应的点像素值设置为 255，其余点设置为 0。最终，对二值化后的图像求和，得到网络对类别c的 1 维张量，从而将语义分割问题转换为与分类问题相同的形式，进而便可以应用 Grad-CAM 算法求解。如图 7 所示。

图 7 将原图（左）喂入模型，根据模型输出及所感兴趣类别$bottle$，对图像二值化（中），应用 Grad-CAM 求解（右）

3 基于 U-Net 网络的语义分割任务及 Grad-CAM 算法可视化

本节详细描述了基于 U-Net 网络进行语义分割任务所需要的步骤。如数据集的组织、数据增强、训练、预测等算法流程。最后针对改进的 U-Net 网络，使用 Grad-CAM 算法，给出了将其$backbone$最后一个卷积层所提取到的特征图所关注的特征以热力图的形式可视化的步骤。

3.1 数据集的制作与组织

在项目的工作目录下，我们建立 VOCdevkit 文件夹，全部训练、测试数据均按照如下目录结构存放。

```
.
`-- VOC2007
    |-- ImageSets
    |   `-- Segmentation
    |        |--train.txt
    |        `--val.txt
    |-- JPEGImages
    `-- SegmentationClass
```

其中，JPEGImages 存放训练/测试的原图片(RGB)；SegmentationClass 存放训练/测试的标签图片(灰度图)。ImageSets/Segmentation 中存放两个.txt 文本文件——train.txt 存放全部训练图片的文件名，val.txt 存放全部测试图片的文件名。这样的文件布局是经典的 PASCAL VOC 数据集的文件组织形式。当我们想要在自建数据集上训练时，只需将自建数据集按照 VOC 格式组织，便可以对本文的全部代码复用。

3.2 训练

对于不同的训练任务来说，使用$pytroch$框架训练的流程基本一致。流程图

如下：

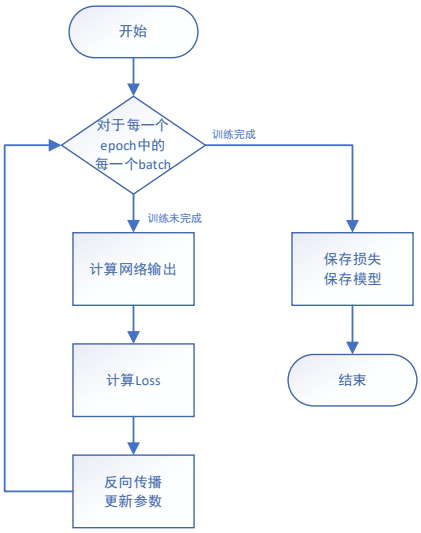

图 8 训练流程图

如果仅仅对训练集上的数据进行训练，得到的权重的泛化性通常较差，因此需要一定的数据增强；同时，训练时初始学习率的设置也对模型能否收敛到最优值有着很大的影响。

3.3 数据增强

数据增强也称为数据扩增，是一种扩充数据规模的有效方法。该技术的发展主要具有以下 3 方面重要意义[7]：

（1）丰富数据集本身。将不断发展的数据增强技术应用于各种数据集上，很好地解决了目前数据集存在的规模小、质量差、不均衡且难以获取等问题，从数量和性能方面丰富了数据集本身，这是数据增强对数据集起到的最直接的作用。

（2）提升相应分类检测系统性能。通过数据增强获得大量结构合理、种类多样的数据，很好地满足了深度学习模型对数据集数量和质量的高要求，减少了网络过拟合现象，从而得到泛化性能好的网络模型，对相应分类器、检测器等准确率的提升具有一定促进作用。

（3）拓展延伸价值。数据增强技术不断发展，现已广泛应用于众多领域。在数据至上的时代，从数据角度推动多个行业发展，这是数据增强技术带来的附加延伸价值，具有深远的意义。

简单的说，通过对图片加入噪声、模糊处理、颜色变换、反转、裁剪、缩放变形等方法，每一轮迭代的训练集会出现些许不同，由此极大的丰富了数据集，使得模型的泛化能力更强。

本文的训练过程中，对训练图片采用随机比例的缩放、随机角度的旋转、随机参数的色域变换，达到数据增强的目的。

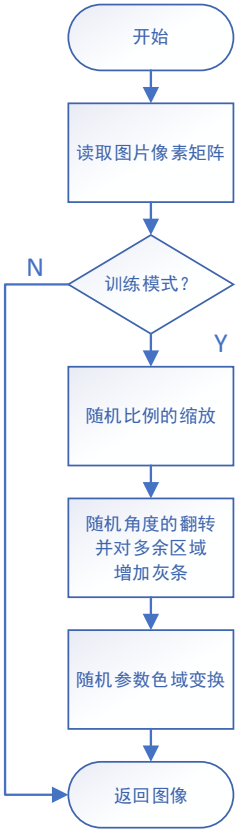

图 9 数据增强流程

3.4 测试

在训练了若干个 epoch 后，我们选择对测试集验证，查看当前模型参数的$F1_score$作为评价标准。

$F1_score$是模型准确率与召回率的调和平均数：

$$F1\_socre = \frac{2TP}{2TP + FN + FP} \tag{9}$$

其中，TP、FP、FN为混淆矩阵中的三个元素：TP为真正例个数，FP为假正例个数，FN为假负例个数。

3.5 预测

在训练完成后，我们可以对未知图片进行预测，即完成语义分割过程。预测时，我们需要注意两个问题：

1） 模型输入向量的维度

2） 模型输出向量的解码

我们的模型接受的输入向量的维度$[b, 3, input_size, input_size]$，其中$b$为$batch$大小。而在预测时，我们通常是对单张图片进行预测。但单张图片无法作为输入向量喂入网络，因为输入图片向量缺少$batch$维度，因此，我们需要为单张图片向量添加一个$batch = 1$维度。

模型的输出向量维度$[1,21, input_size, input_size]$。我们对输出向量去掉第一个维度后，得到了与输入图相同尺寸的 21 个通道图片。通道c的特征图坐标为

(x,y)的像素值代表原图中像素(x,y)属于类别c的概率。因此，我们对通道方向进行$softmax$处理后，取出每一个通道值最大的下标，该下标即为该像素点所属的类别。

最后，为了更好的展示语义分割的结果，我们对原图像与语义分割结果图像进行混合，混合后像素值满足下式：

$$img_{ij} = (1-\alpha)img_{ij}^{1} + \alpha img_{ij}^{2} \tag{10}$$

其中，img_{ij}为最终图像坐标为(i,j)处的像素值，img_{ij}^{1}为原图像坐标(i,j)处的像素值，img_{ij}^{2}为语义分割结果图像中坐标(i,j)处的像素值。α为权重系数，本文实现的算法中$\alpha = 0.7$。

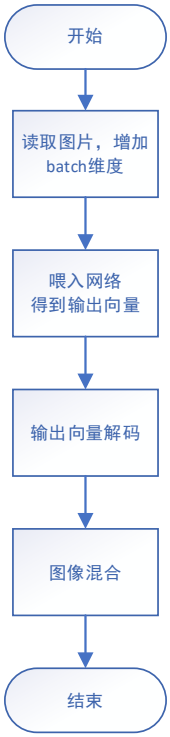

图 10 预测流程图

3.6 Grad-CAM 对 U-Net 网络可视化

2.3.2 中已经阐述了对 Grad-CAM 算法的改进，使其可以应用到语义分割网络中。首先，对于我们感兴趣的类别c与待分析特征层A，根据网络输出解码后的向量，制作标准化掩码$mask$。根据掩码$mask$与感兴趣类别c，计算网络输出向量在通道c上的有效值之和，以此作为 Grad-CAM 梯度反传的目标值。最后，根据式(7)、式(8)计算出目标特征层各通道加权求和的结果——Grad-CAM 输出。通过双线性插值上采样，并与原图像素值叠加，得到了最终的热力图。算法流程如下：

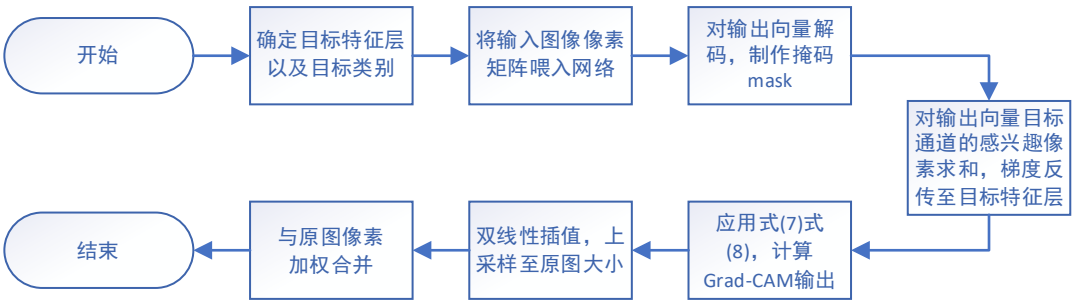

图 11 Grad-CAM 可视化算法流程图

4 实验结果

本次实验所使用的*python*解释器版本以及主要第三方库环境如下：

表 1 运行环境

解释器/第三方库	版本
python	3.7.13
numpy	1.21.5
pytorch	1.11.0
opencv	3.4.2

训练阶段所使用的初始参数如下：

表 2 训练参数

参数	值	解释
Cuda	True	是否使用 GPU
num_classes	21	PASCAL VOC 数据集有 20 个类别，语义分割任务额外增加 1 个背景类别
input_shape	[512, 512]	对喂入网络的图像缩放至 512×512
optimizer_type	adam	使用 Adam 优化器
momentum	0.9	Adam 优化器的动量
lr_decay_type	cos	学习率下降方式
Init_lr	1e-4	初始学习率
Epoch	45	模型总共训练的 epoch 数
batch_size	4	batch 大小

使用学校服务器的 RTX 2080 Ti 显卡进行训练。

4.1 训练损失以及模型评价

经过 45 个*epoch*训练后，训练过程中在训练集与测试集的损失函数值的变化趋势如图 12 所示。

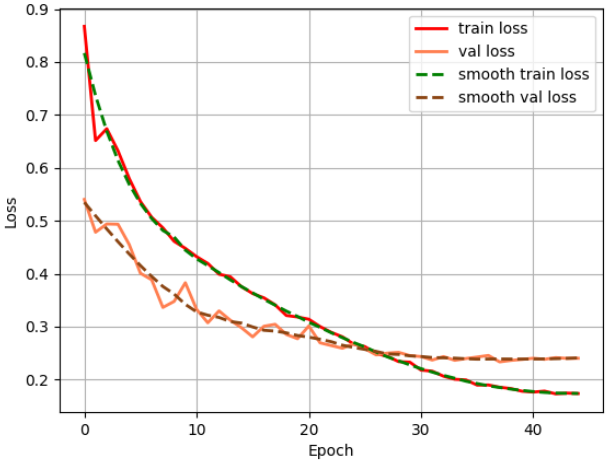

图 12 损失函数变化

可以看出，$25epoch$时损失函数在训练集与测试集上的值几乎相等；在$30\sim40epoch$的迭代中，测试集上损失函数值变化不大，而训练集上的损失函数值继续下降，存在过拟合趋势。因此，最终选择$epoch = 30$时的模型M作为训练结果。

使用模型M，对测试集预测，得到不同类别的 mIOU、mPA、准确率、召回率如下图所示：

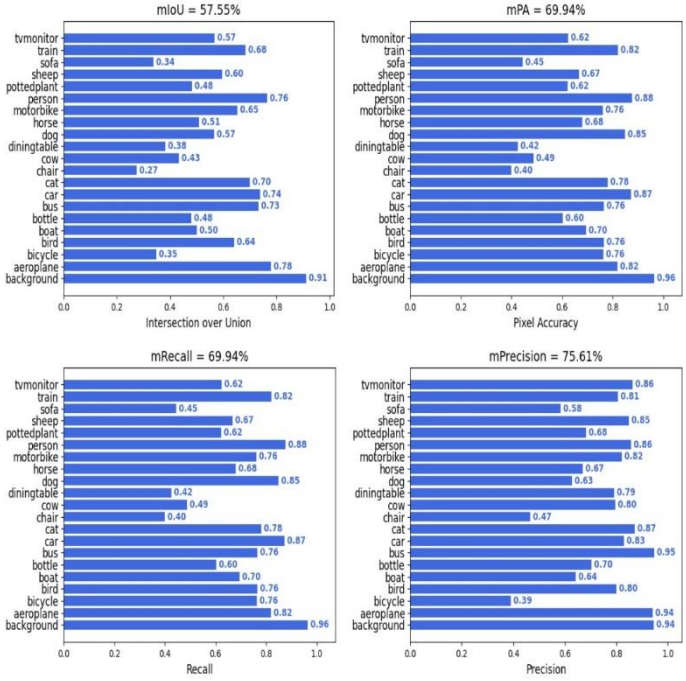

图 13 模型M在测试集上的评价

模型在测试集上的得分如下：

mIoU：57.55；

mPA：69.94；

mRecall：69.94；

mPrecision：75.61；

4.2 预测结果及 Grad-CAM 可视化

使用模型M，对在学校拍摄的照片进行语义分割，并使用 Grad-CAM 对网络关注的区域可视化展示。如图14~18所示。

图 14 语义分割结果及 Grad-CAM 可视化(1)

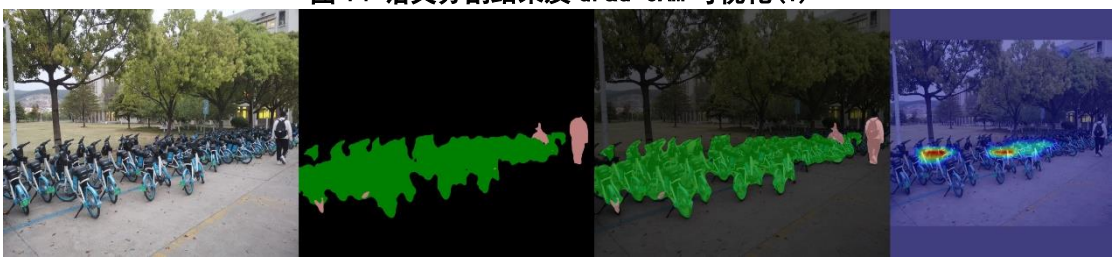

图 15 语义分割结果及 Grad-CAM 可视化(2)

图 16 语义分割结果及 Grad-CAM 可视化(3)

图 17 语义分割结果及 Grad-CAM 可视化(4)

图 18 语义分割结果及 Grad-CAM 可视化(5)

我们选择$backbone$即 VGG16 网络的最后一个卷积层，作为 Grad-CAM 的目标特征层。如图14~18所示,自左至右分别为：原图，背景像素值为 255 的语义分割结果，与原图融合后的语义分割结果，Grad-CAM 算法可视化结果。其中，Grad-CAM 算法在图14~18中设置的关注类别分别为"$person$"、"$bicycle$"、"car"、"$person$"、"cat"。

可以看出，对于所感兴趣的类别，Grad-CAM 较为准确的指出了表征感兴趣类别的特征所在区域，即很好的解释了网络预测结果形成的原因。

5 与其他算法的比较

5.1 改进的 U-Net 与 U-Net 的比较

按照 U-Net 作者给出的网络结构，搭建了传统的 U-Net 网络，在同样的训练超参数下，最终模型在测试集上的得分如下：

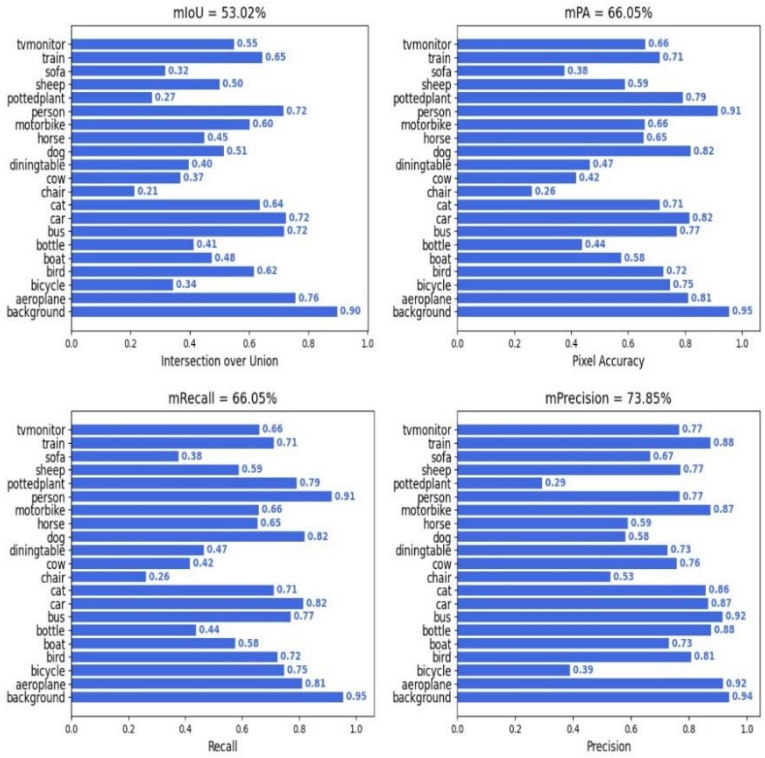

图 19 U-Net 网络在测试集上的得分

U-Net 网络改进前后的对比如下表：

表 3 网络结构优化前后对比

	U-Net	改进的 U-Net
miou	53.02%	57.55%
mPA	66.05%	69.94%
mRecall	66.05%	69.94%
mPrecision	73.85%	75.61%

可以看出，当使用了在PASCAL VOC数据集上特征提取效果较好的VGG16作为 backbone、且上采样使用双线性插值而非转置卷积后，网络的精确度有了较大幅度的上升。

5.2 Grad-CAM在不同深度特征图上的表现对比

随着特征提取网络的加深，越深处的卷积网络提取到的特征图的感受野较大，理论上应该更好的表征原图像。VGG16作为backbone自浅至深共有13个卷积层，我们将 13 个卷积层提取到的特征图分别作为 Grad-CAM 算法的输入，结果如下；

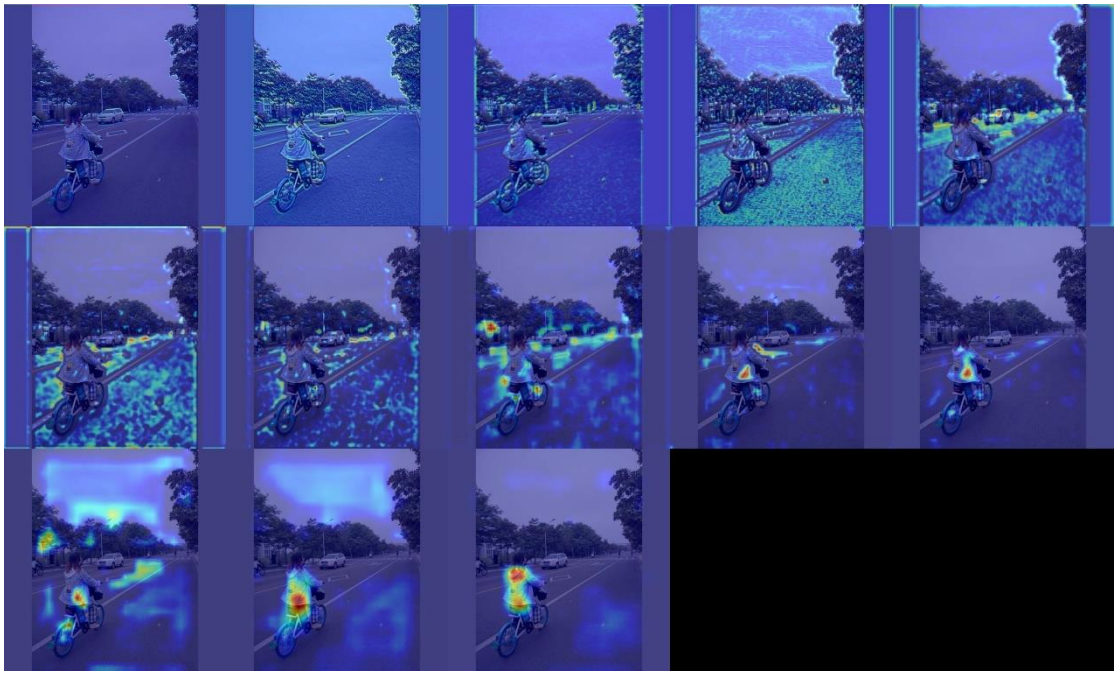

图 20 VGG16的13个特征图所关注的特征(自左向右、自上而下)

从图 20 可以清楚的看出，位置靠前的卷积层所关注的特征较为发散，不能很好的关注到表征目标(person)的关键特征；随着卷积所在位置深度的增加，其所提取到的特征图越发明确，最深处的特征图正确的关注到了表征目标的关键性特征。

6 结论

U-Net网络是一个轻量级语义分割网络。我们将U-Net特征提取及下采样部分替换为 VGG16，上采样部分使用双线性插值替代转置卷积，使得改进后的 U-Net 网络精度以及速度上都有了很大的提升(表 3)。

为了更好的探究神经网络卷积层所提取的特征图表征的原图区域，使用 Grad-CAM 算法，以热力图的形式可视化展示了卷积神经网络在特征提取上的有效性。同时侧面证明了改进后的U-Net网络结构的正确性。

不过，受限于算力等各种问题，最终模型的精度还有待提升，预测的结果并不是特别准确。

在 U-Net 被提出后，陆陆续续出现了许多的改进结构——Fusion Net、

R2U-Net、Attention UNet 等，这些网络都是类似于 UNet 的编码器-解码器结构，或者替换了 backbone，或者加入了注意力机制，或者增加了多尺度特征的融合。未来，对现有网络结构的改进或许可以通过上述方法进行。

对于 Grad-CAM，同样存在许多改进算法——GradCAM++使用了二阶导数、XGradCAM 则是对特征图进行了归一化。未来，或许可以融合梯度与二阶导数，或是使用其他可以表征特征层类别的信息完成可视化的目标。

7 心得体会

通过本门课程的学习，收获颇丰。

在模糊计算章节中，模糊聚类算法给我留下了很深刻的印象。特别是模糊相似矩阵的传递闭包是模糊等价矩阵这一定理，加深了我对模糊计算理论如何解决实际问题的理解。

在神经计算章节中，这是我在课堂上第一次系统的学习神经网络的相关知识。通过学习感知机、BP 神经网络，巩固了我对深度学习基础理论的掌握。

在进化计算章节中，在曾经多次数学建模比赛中使用遗传算法的基础上，再次加深了自身对这一算法的理解。

最后，由于目前在研究增量学习语义分割模型，因此我选择了一个轻量级的语义分割网络——U-Net 作为本次大作业的第一个内容。

增量学习语义分割问题存在两个主要问题——所有增量学习任务中都存在的灾难性遗忘(catastrophic forgetting)以及语义分割增量学习中特有的背景漂移(background shift)，而后者通常会加剧灾难性遗忘问题[8]。解决灾难性遗忘的一个较为常见的方式是增加蒸馏损失。仅仅对结果增加蒸馏损失的模型表现一般，而对特征层面增加特征蒸馏后的模型表现较好。使用特征蒸馏的方式之一是借鉴了 Grad-CAM 算法。因此，Grad-CAM 算法对卷积网络特征提取效果的可视化成为了本次作业的第二个内容。

通过完成这次大作业，我对语义分割、U-Net 网络的结构、Grad-CAM 算法都有了更深的理解。感谢本次课程。

8 参考文献

[1] 张珂,冯晓晗,郭玉荣,苏昱坤,赵凯,赵振兵,马占宇,丁巧林.图像分类的深度卷积神经网络模型综述[J].中国图象图形学报,2021,26(10):2305-2325.

[2] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation[J]. Springer International Publishing, 2015.

[3] 许鹏飞. 基于 U-Net 和聚类的医学图像分割方法研究[D].南京邮电大学,2021.DOI:10.27251/d.cnki.gnjdc.2021.001050.

[4] Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

[5] Selvaraju R R , Cogswell M , Das A , et al. [IEEE 2017 IEEE International Conference on Computer Vision (ICCV) - Venice (2017.10.22-2017.10.29)] 2017 IEEE International Conference on Computer Vision (ICCV) - Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization[J]. 2017.

[6] Chattopadhyay A , Sarkar A , Howlader P , et al. Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks[J]. IEEE, 2018.

[7] 朱晓慧,钱丽萍,傅伟.图像数据增强技术研究综述[J].软件导刊,2021,20(05):230-236.

[8] Douillard A , Chen Y , Dapogny A , et al. PLOP: Learning without Forgetting for Continual Semantic Segmentation[J]. 2020.

9 代码

附件包括全部代码文件。VOC 数据集、模型的权重文件占用磁盘空间过大，没有包含在附件中。

以下展示部分代码。

./nets/unet.py

```
import torch
import torch.nn as nn

from nets.resnet import resnet50
from nets.vgg import VGG16


class unetUp(nn.Module):
    def __init__(self, in_size, out_size):
        super(unetUp, self).__init__()
        self.conv1  = nn.Conv2d(in_size, out_size, kernel_size = 3, padding = 1)
        self.conv2  = nn.Conv2d(out_size, out_size, kernel_size = 3, padding = 1)
        self.up     = nn.UpsamplingBilinear2d(scale_factor = 2)
        self.relu   = nn.ReLU(inplace = True)

    def forward(self, inputs1, inputs2):
        outputs = torch.cat([inputs1, self.up(inputs2)], 1)
        outputs = self.conv1(outputs)
        outputs = self.relu(outputs)
        outputs = self.conv2(outputs)
        outputs = self.relu(outputs)
        return outputs

class Unet(nn.Module):
    def __init__(self, num_classes = 21, pretrained = False, backbone = 'vgg'):
        super(Unet, self).__init__()
        if backbone == 'vgg':
            self.vgg    = VGG16(pretrained = pretrained)
            in_filters  = [192, 384, 768, 1024]
        elif backbone == "resnet50":
            self.resnet = resnet50(pretrained = pretrained)
            in_filters  = [192, 512, 1024, 3072]
        else:
            raise ValueError('Unsupported backbone - `{}`, Use vgg, resnet50.'.format(backbone))
        out_filters = [64, 128, 256, 512]
```

```
36.
37.         # upsampling
38.         # 64,64,512
39.         self.up_concat4 = unetUp(in_filters[3], out_filters[3])
40.         # 128,128,256
41.         self.up_concat3 = unetUp(in_filters[2], out_filters[2])
42.         # 256,256,128
43.         self.up_concat2 = unetUp(in_filters[1], out_filters[1])
44.         # 512,512,64
45.         self.up_concat1 = unetUp(in_filters[0], out_filters[0])
46.
47.         if backbone == 'resnet50':
48.             self.up_conv = nn.Sequential(
49.                 nn.UpsamplingBilinear2d(scale_factor = 2),
50.                 nn.Conv2d(out_filters[0], out_filters[0], kernel_size
    = 3, padding = 1),
51.                 nn.ReLU(),
52.                 nn.Conv2d(out_filters[0], out_filters[0], kernel_size
    = 3, padding = 1),
53.                 nn.ReLU(),
54.             )
55.         else:
56.             self.up_conv = None
57.
58.         self.final = nn.Conv2d(out_filters[0], num_classes, 1)
59.
60.         self.backbone = backbone
61.
62.     def forward(self, inputs):
63.         if self.backbone == "vgg":
64.             [feat1, feat2, feat3, feat4, feat5] =
    self.vgg.forward(inputs)
65.         elif self.backbone == "resnet50":
66.             [feat1, feat2, feat3, feat4, feat5] =
    self.resnet.forward(inputs)
67.
68.         up4 = self.up_concat4(feat4, feat5)
69.         up3 = self.up_concat3(feat3, up4)
70.         up2 = self.up_concat2(feat2, up3)
71.         up1 = self.up_concat1(feat1, up2)
72.
73.         if self.up_conv != None:
74.             up1 = self.up_conv(up1)
75.
76.         final = self.final(up1)
77.
78.         return final
79.
```

```
    def freeze_backbone(self):
        if self.backbone == "vgg":
            for param in self.vgg.parameters():
                param.requires_grad = False
        elif self.backbone == "resnet50":
            for param in self.resnet.parameters():
                param.requires_grad = False

    def unfreeze_backbone(self):
        if self.backbone == "vgg":
            for param in self.vgg.parameters():
                param.requires_grad = True
        elif self.backbone == "resnet50":
            for param in self.resnet.parameters():
                param.requires_grad = True

```

predict.py

```
import time

import cv2
import numpy as np
from PIL import Image
import os

from unet import Unet
from predict_grad_cam import get_cam

if __name__ == "__main__":
    unet = Unet()
    mode = "dir_predict"
    test_interval = 100
    dir_origin_path = "img/"
    dir_save_path   = "img_out/"

    if mode == "predict":
        while True:
            img = input('Input image filename:')
            try:
                image = Image.open(img)
            except:
                print('Open Error! Try again!')
                continue
            else:
                class_name = input("Input grad-cam class: ")
                r_image = unet.detect_image(image)
```

```
                get_cam(unet, image, unet.input_tensor, unet.real_out,
class_name, img.split("/")[2])
                # print(unet.input_tensor.shape, unet.real_out.shape)
                print(img)
                r_image.show()
                r_image.save(os.path.join(dir_save_path,
img.split("/")[2]))


    elif mode == "dir_predict":
        import os
        from tqdm import tqdm

        img_names = os.listdir(dir_origin_path)
        for img_name in tqdm(img_names):
            if img_name.lower().endswith(('.bmp', '.dib', '.png',
'.jpg', '.jpeg', '.pbm', '.pgm', '.ppm', '.tif', '.tiff')):
                image_path  = os.path.join(dir_origin_path, img_name)
                image       = Image.open(image_path)
                r_image     = unet.detect_image(image)
                # print(unet.input_tensor.shape, unet.real_out.shape)
                # 得到grid_cam图
                get_cam(unet, image, unet.input_tensor, unet.real_out,
"person", img_name)
                # print(unet.input_tensor)
                if not os.path.exists(dir_save_path):
                    os.makedirs(dir_save_path)
                r_image.save(os.path.join(dir_save_path, img_name))

    else:
        raise AssertionError("Please specify the correct mode:
'predict', 'video', 'fps' or 'dir_predict'.")
```

predict_grad_cam.py

```
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
import torch
import numpy as np
from PIL import Image
from pytorch_grad_cam.utils.image import show_cam_on_image,
preprocess_image
from unet import Unet
from pytorch_grad_cam import GradCAM

```

```
def reshape_image(image, size):
    iw, ih = image.size
    w, h = size
    scale = min(w / iw, h / ih)
    nw = int(iw * scale)
    nh = int(ih * scale)

    image = image.resize((nw, nh), Image.BICUBIC)
    new_image = Image.new('RGB', size, (128, 128, 128))
    new_image.paste(image, ((w - nw) // 2, (h - nh) // 2))
    return new_image

class SemanticSegmentationTarget:
    def __init__(self, category, mask):
        self.category = category
        self.mask = torch.from_numpy(mask)
        if torch.cuda.is_available():
            self.mask = self.mask.cuda()

    def __call__(self, model_output):
        return (model_output[self.category, :, :] * self.mask).sum()

def main(img_name, class_name):
    model = Unet()
    model.net = model.net.eval()
    img = Image.open("./img/" + img_name)
    model.detect_image(img)
    get_cam(model, img, model.input_tensor, model.real_out,
class_name, img_name)

def get_cam(model, img, input_tensor, output_tensor, class_name,
img_name):
    image = np.array(reshape_image(img, [512, 512]))
    rgb_img = np.float32(reshape_image(img, [512, 512])) / 255

    normalized_masks =
torch.nn.functional.softmax(output_tensor.float(), dim=1).cpu()
    sem_classes = [
        '__background__', 'aeroplane', 'bicycle', 'bird', 'boat',
'bottle', 'bus',
        'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse',
'motorbike',
        'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor'
    ]
    sem_class_to_idx = {cls: idx for (idx, cls) in
enumerate(sem_classes)}

    car_category = sem_class_to_idx[class_name]
```

```
    car_mask = normalized_masks[0, :, :, :].argmax(axis=0).detach().cpu().numpy()
    car_mask_uint8 = 255 * np.uint8(car_mask == car_category)
    car_mask_float = np.float32(car_mask == car_category)

    both_images = np.hstack((image, np.repeat(car_mask_uint8[:, :, None], 3, axis=-1)))
    target_layers = [model.net.vgg.features[28]]
    targets = [SemanticSegmentationTarget(car_category, car_mask_float)]
    with GradCAM(model=model.net,
                 target_layers=target_layers,
                 use_cuda=torch.cuda.is_available()) as cam:
        grayscale_cam = cam(input_tensor=input_tensor,
                            targets=targets)[0, :]
        cam_image = show_cam_on_image(rgb_img, grayscale_cam, use_rgb=True)

    Image.fromarray(cam_image).save("./img_out_grid/" + class_name + "_" + img_name)



if __name__ == '__main__':
    main("30.jpg", "cat")

```