文章编号:1671-6833(2022)06-0001-07

一种面向 UI 手稿识别的数据集制作方法

杨 起[1], 刘牧耕[2], 马 郓[3]

(1. 北京大学深圳研究生院 信息工程学院,广东 深圳 518055;2. 北京大学 计算机学院,北京 100871;
3. 北京大学 人工智能研究院,北京 100871)

摘 要: UI 手稿识别是图像目标检测技术在软件工程领域的重要应用。由于 UI 手稿图像与自然图像有着较大的差异,而且主要依靠人工绘制,所以制作用于深度学习模型训练的 UI 手稿数据集往往比较困难,耗费大量人力。针对此问题,通过对 UI 手稿数据集的制作流程进行优化改进,提出了一种 UI 手稿数据集快速制作方法 UIsketcher。在 UIsketcher 方法中,用户只需要完成一些基础 UI 组件的绘制,不需要任何框选标注,即可自动生成用于深度学习模型训练的数据集。与传统方法进行对比实验,结果表明:用户只需要绘制相对于传统方法 25%的组件数量,即可得到相似的训练效果;若绘制传统方法 75%的组件数量,训练效果将更好,可达到比传统方法更高的准确率。

关键词: 软件智能化开发服务; UI 手稿; 识别; 目标检测; 数据集; 数据增强

中图分类号: TP311;O244 **文献标志码**: A **doi**:10.13705/j.issn.1671-6833.2022.06.009

0 引言

图像目标检测旨在从一张图像中寻找目标物体,并给出目标物体的所属区域。图像目标检测应用场景广泛,如在农业领域实现蔬菜的识别检测[1]、工业领域实现产品瑕疵检测[2]、电商领域实现图片文字提取[3]等。随着以深度学习为代表的人工智能技术的快速发展,基于深度神经网络的图像目标检测算法在算法性能方面取得了巨大突破。

在软件工程领域,利用图像目标检测技术可以实现从设计师绘制的 UI 手稿图像中识别 UI 组件及其位置,进而通过代码生成技术直接生成 UI 界面程序,提高软件开发效率。近年来,学术界和产业界均基于 UI 手稿识别算法开发了多种界面自动生成工具[4-6],并组织了相关竞赛[7]。

与普通图像相比,UI 手稿没有颜色特征,仅有纹理特征,且 UI 手稿对坐标的精准度要求较高[8]。这些特点导致了 UI 手稿识别任务需要大量特定的训练数据来获得高鲁棒性的模型。然而,UI 手稿一般由设计师个人自行保存,难以从互联网大规模获取,训练模型所需的数据集仅为少数志愿者绘制的少量手稿,导致 UI 手稿识别的准确率始终与其他图像识别领域的准确率相差甚大。

针对这一问题,本文设计了一种能够大幅提高手稿数据集制作效率并显著提升原数据集性能的方法——UIsketcher。该方法取消了人工标注数据集的环节,不要求绘制者完成整幅手稿设计图的绘制,而仅需要绘制特定的 UI 组件,即可完成数据集制作的全部人工工作。

1 相关研究

1.1 UI 手稿识别

UI 手稿识别的本质为图像目标检测。当前目标检测任务主要通过深度学习技术实现,而深度学习模型需要由标注过的数据集进行训练。通过对大量已标注的手稿图片的训练,实现对手稿元素的分类和定位。因此,对于不同需求的手稿识别应用,其需要识别的目标分类也是不同的,这就涉及数据集的定制化。

1.2 手稿绘制

相对于其他类型的图像,UI 手稿图像难以从互联网中采集到原始数据,因此制作 UI 手稿数据

收稿日期:2022-01-27;**修订日期**:2022-05-19

基金项目:国家自然科学基金资助项目(62102009)

通信作者:马郓(1989—),男,山东郓城人,北京大学研究员,博士,博士生导师,主要从事智能化系统软件、Web 系统、移动计算、服务计算研究,Email:mayun@pku.edu.cn。

集通常需要由专业的UI设计师来完成若干种不同布局样式的手稿图绘制,如微软的sketch2code[4]工具使用了数百个UI手稿。

1.3 标准化处理

由于UI手稿通常是由画笔在纸张上绘制而成,可能存在画笔的颜色不同、粗细不同、纸张的颜色差异等情况。同时,在拍摄成图片输入至识别应用之前,还会存在拍摄倾斜和拍摄摇晃导致的模糊、拍摄时光线不均匀造成的画面明暗不一等各种影响识别准确率的因素。因此通常会在识别之前进行标准化处理,即对原始图片进行旋转、剪裁、灰度处理等操作,尽可能减少外部因素的干扰。

1.4 数据集标注

在得到电子档的手稿图片数据集之后,还需要对其进行标注工作。对于目标检测任务,通常采用框选目标对象的最小外接矩形的方式来确定该元素的位置,并标注该区域所属的分类。

2 UIsketcher方法设计

在大规模数据集的制作中,随着手稿数量的大量增加,分类的数量也随之增加。这就需要多个志愿者来绘制不同的手稿以满足数据集的多样性要求。然而,传统的绘制方法有两个缺点:一是志愿者需要学习UI组件如何构成前端页面,且还需要尽可能地不重复绘制,以实现随机效果,这就要求参与数据集绘制的人员付出额外的学习成本与精力;二是手稿的背景参差不齐,有些可能存在较为明显的阴影,有些可能绘制在带有特殊纹理的纸张上。虽然丰富的背景噪声可以提高模型的鲁棒性,但可能存在同一个志愿者绘制的手稿背景都是一种样式的情况,这样反而减少了背景噪声的多样性,也就是说,通过传统方法制作的手稿数据集中,一种风格的背景噪声往往对应一种风格的笔迹。

针对上述两个问题,本文提出了一种面向UI手稿识别的数据集高效制作方法——UIsketcher。UIsketcher不要求绘制者绘制完整的UI元素组合,而是仅仅根据数据集分类标签绘制整页的UI组件。通过UIsketcher的组件图像分割工具分割提取出组件,并放入素材库中待用;同时,支持给数据集添加纸张背景库,放入不同样式的背景噪声图案。以上准备工作完成后,生成工具会自动变换素材并生成带有标注信息的数据集。

图1为UIsketcher工作的完整流程,包括绘制UI组件、组件拆分、组件增强、训练数据集生成等步骤。

2.1 绘制UI组件

传统方法的难点之一在于需要绘制完整的前端页面并进行标注。实际上,对于机器学习的模型来说,训练时主要依赖的是图片的纹理特征、色彩特征等。目标在画面中的任意位置均应被AI识别。因此,本文方法直接忽略掉页面中各个组件的位置关系,只需要通过组件模板库任意组合出各种组件元素即可。

同时,设计了一种将画面中的不同位置的元素进行最小外接矩形切割的算法。该算法只需要在一个页面上绘制 n 个K类型的组件,并为该页面进行唯一一次标注,将其标注为K类型,从而得到 n 个K类型的独立组件图片,如图1中的原始数据集。通过这种整页绘制相同组件的方式,

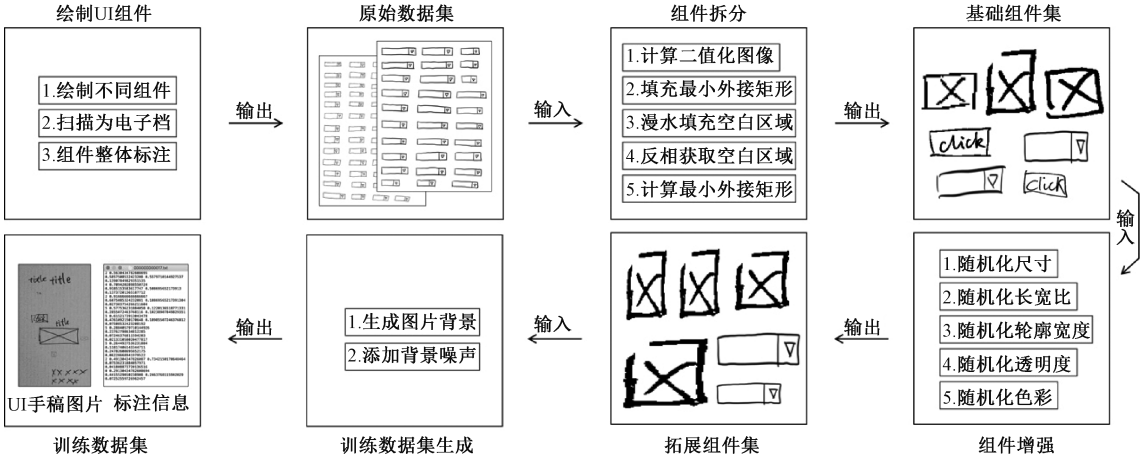


图1 UIsketcher工作流程

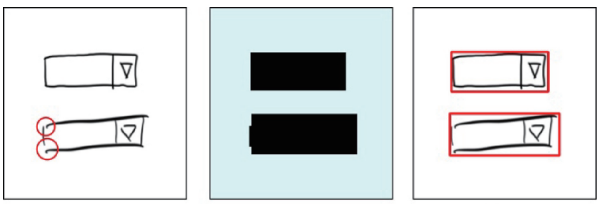
Figure 1 Workflow of UIsketcher

减少人工绘制工作量。这样获得的原始数据集将作为 UIsketcher 的输入。

2.2　组件拆分

在完成手稿绘制后,由于已知原始手稿中每张图所绘制的组件类型,则可以将组件从图像中分离出来,构建一个组件库,以便进行数据集的手稿图片的最终合成。

考虑原始组件的拆分需要保证尽可能高的精准度,例如其准确的最小外接矩形,因此,设计了一种专门用于手稿原始图像的组件拆分算法。在原始数据集无可见重叠,拍摄图片无明显阴影的情况下,该算法可以准确地分割出原始图片中的每一个组件,其原理为通道蒙版抠图,而非现在各类图像处理应用上常见的魔棒抠图。魔棒抠图的主要流程为指定图像中一个起始像素点,根据像素的颜色阈值范围,逐个搜索周围相似的像素点,并清空其色彩值,直至搜索完画面中所有符合条件的像素点。这种方式主要针对纹理单一、图像造型规则的情况。但由于手稿图像时常会出现未闭合的线框和线段,普通的魔棒抠图法对抠图区域判断的准确性会受到明显影响。因此,通过更具鲁棒性的设计对该问题进行改进。与魔棒抠图法最大的不同在于通道蒙版抠图对画面中所有层级的纹理轮廓都进行了一次搜索,并通过其最小外接矩形的并集得到每个组件真实的外边界。图 2(a)为一个典型的非封闭图形的例子,如果直接采用魔棒抠图或漫水填充,则无法忽略掉红色圈出的缺口。因此,将图片处理为图 2(b)所示的形式,再进行分离操作,就很容易获取目标组件的真实位置,如图 2(c)所示。



(a) 边界缺口　(b) 填色处理　(c) 分离结果

图 2　非封闭图形举例

Figure 2　Examples of unclosed shape

手稿识别技术通常与画笔颜色、纸张颜色无关,因此首先对图片进行灰度处理。由于笔迹的颜色通常和纸张是有明显区分的,手稿输入的内容本质上是一张二值化图像。因此,通过式(1)将图像转为灰度图后,再进行二值化处理。根据式(2),将图像的像素点取值由$\{(R,G,B) \mid R,G,B \in [0,255]\}$转化为$\{0,1\}$。

$$f_{\text{grey}}(R,G,B) = 0.299R + 0.587G + 0.114B; \quad (1)$$

$$f_{\text{binary}}(Pixel) = \begin{cases} 1, & Pixel \geqslant \lambda; \\ 0, & Pixel < \lambda; \end{cases} \quad (2)$$

$$BMap = \{(f_{\text{grey}} \circ f_{\text{binary}})(R_i,G_i,B_i) \mid R_i,G_i,B_i \in Pixel_i, Pixel_i \in Image\}。 \quad (3)$$

在处理过程中,λ 为可变参数,默认值为 127,可根据图像的实际情况进行调整。由于图像中笔迹通常是偏深色的,背景通常是偏浅色的,因此取图像灰度值的中位数与笔迹灰度值的中间值,可得到适用的 λ 值。需要注意的是,如果是纸张为深色,笔迹为浅色的情况,还须进行一次反色处理。式(3)表示图像中的每一个像素通过灰度值进行二值化处理,可得到一个二值化映射表($BMap$),二维矩阵的每个元素对应原先图像的每个像素。自此,图像的内容区域与背景区域已经可以被轻松地分离。

虽然图像转为 $BMap$ 后的内容已经比较干净清晰,但是计算机并不能直接获取到每个组件的准确轮廓和轮廓断开的情况,如图 2 所示。此外,算法的目标在于获取每个组件的外轮廓,并取其最小外接矩形,即只需要获得组件的造型。因此,本文设计了一种算法来实现对组件内部的填充。当组件内部被填充后,可以轻松地获取组件的外轮廓,且不受内部内容干扰。

为了填充组件的内部区域,考虑了轮廓膨胀、高斯模糊等常见的处理方式。但是由于组件的样式各异,单一的处理并不适用于所有的情况。于是从连续轮廓的外接矩形的角度来考虑解决方案,因为只要轮廓存在相连的地方,都可以被认为是一个整体。当对这个整体取最小外接矩形时,就很容易覆盖掉缺口区域。组件拆分示例如图 3 所示,具体步骤如下。

(1)对于画面中的每个轮廓对象,计算出能够覆盖该轮廓范围的最小外接矩形。

(2)绘制该矩形,由于矩形框是一个封闭的图形,故不进行实心填色也依旧是封闭的图形,如图 3 所示。相较于图 3(a),图 3(b)的每个组件都有一个清晰的最小外接矩形。

(3)对组件的内部区域进行填充操作。倘若在图 3(b)中使用实心填色,组件内部区域并不一定是完全填满的,如果存在个别像素没有被识别为轮廓而绘制出最小外接矩形,这样的像素就会成为画面的噪点。因此,设计了另一种更为合理的方法,保证矩形内部一定能够被完全填色。

由于前面的处理工作已经对组件的外轮廓进

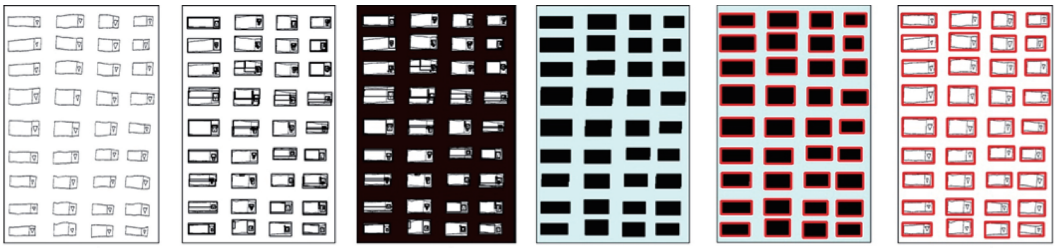

(a) 手稿 (b) 最小外接矩形 (c) 填色 (d) 反相 (e) 准确框选 (f) 叠加

图3 组件拆分示例

Figure 3 Example for splitting components

行了清晰的描边处理,并且背景区域是纯色,所以此时对轮廓外的区域进行填色会十分高效。设计的填色公式为

$$f_{\text{flood}}(Pixel)=\begin{cases}\alpha, & Pixel=0;\\ Pixel, & Pixel\neq 0。\end{cases} \tag{4}$$

通过f_{flood}对画面的非组件区域进行漫水填充操作,为与背景色区分,将填充的目标颜色以α代替。当完成填充后,组件内的空白区域还是背景色,但组件外部的区域全部都为α,如图3(c)所示。

$$f_{\text{complement}}(Pixel)=\begin{cases}\alpha, & Pixel=\alpha;\\ \beta, & Pixel\neq\alpha。\end{cases} \tag{5}$$

(4)通过式(5)对画面进行反相处理。由于在漫水填充之后,所有的背景区域都变成了α,但组件边框及其内部不为α。此时图像为三值图像,包括α、背景色、边框色。如果将不是α的值找出并进行统一填色,即可使画面重新变成二值图像,并且该二值图像不存在任何组件内纹理,如图3(d)所示。

(5)在经过上述的步骤之后,通过搜索第1层图像轮廓,即可获得准确的框选信息,且无其他干扰元素,如图3(e)所示。

(6)将矩形框的信息在原图上叠加后,得到图3(f)的效果。

2.3 组件增强

在得到每个手稿图片中的组件绘制位置的最小外接矩形和坐标后,将各类型中的组件剪裁出来,并将分类信息标注于每个组件,得到基础组件库,如图1中的基础组件集。

但这样的基础组件库中包含的组件数量并不够丰富,由于需要使数据集中的数据尽可能的多样化,组件需要进行各种维度的变化。因为组件素材可能是在任意背景平面上绘制的,这可能造成背景纹理不一。如果直接用于数据集的生成,会导致背景的割裂,从而与真实的UI手稿差异较大。因此对原始组件进行了笔迹提取处理,去除掉背景图案并保留笔迹线条。

在得到组件的笔迹线条数据后,对组件的尺寸、长宽比、深浅、粗细等进行随机变换,以增加数据的多样性,从而得到了拓展的数据集,如图1中的拓展组件集。

2.4 训练数据集生成

Ghiasi等[9]通过对实例分割数据集的抠图—复制—粘贴处理,在不需要额外增加人工工作量的情况下,实现数据集更好的训练效果。根据该思路,采用随机取出组件再排列到画布中的方式生成数据集手稿图片。

如果直接随机放置组件在画布中,可能存在组件重叠甚至完全遮挡的问题,因此设计了如下方法来实现合理的随机排布:

$$\boldsymbol{UsageMap}=\begin{bmatrix}P_{x_1y_1} & \cdots & P_{x_ny_1}\\ \vdots & \vdots & \vdots\\ P_{x_1y_n} & \cdots & P_{x_ny_n}\end{bmatrix}; \tag{6}$$

$$f_{\text{dotset}}(X_1,X_2,Y_1,Y_2)=\{Dot_{xy}\mid x\in[X_1,X_2],\ y\in[Y_1,Y_2]\}。 \tag{7}$$

其中,式(6)用于辅助储存随机信息,其结构为一个二维矩阵。对于每一张手稿图片,均生成一个等尺寸的二值画布***UsageMap***。式(7)中,$f_{\text{dotset}}(\cdot)$表示在(X_1,X_2,Y_1,Y_2)所对应的矩形区域内所包含的所有像素点。

考虑到重叠率的问题,首先,通过算法1计算出组件若放置在某个位置与当前的画布上已存在的组件的重叠率。然后,通过算法2给组件分配一个坐标,即从***UsageMap***中获取一个可以放置的区域。算法2为递归函数,表示所有组件持续地获取放置位置。

算法1 重叠率计算。

输入:组件的坐标及当前的二值画布***UsageMap***;

输出:组件的重叠率。

① FUNC getCoverage($X_1,X_2,Y_1,Y_2,$***UsageMap***)

② count=0//初始化计数器

③　FOR *Dot* in $f_{\text{dotset}}(X_1, X_2, Y_1, Y_2)$//由 4 个坐标围成的区域
④　count++//只要点包含于对应区域,则重叠点+1
⑤ RETURN　count/$(X_2 - X_1)$ $(Y_2 - Y_1)$/

算法 2　随机放置。
输入:当前的二值画布 ***UsageMap***;
输出:组件的坐标。

① FUNC getPosition(***UsageMap***)
②　X_1, X_2, Y_1, Y_2 = random(range = ***UsageMap***)//随机获取坐标信息
③ IF getCoverage(X_1, X_2, Y_1, Y_2, ***UsageMap***) $<R$ // R 为允许的重叠率
④　FOR *Dot* in $f_{\text{dotset}}(X_1, X_2, Y_1, Y_2)$//遍历区域内的像素点
⑤　***UsageMap***[X][Y] = 1//缓存区域内的每个点
⑥　RETURN (X_1, X_2, Y_1, Y_2)
⑦　END
⑧ ELSE//重叠率过高,重新获取位置
⑨ getPosition(***UsageMap***)

除此之外,由于纸张背景纹理可能作为噪声信息来影响机器学习的训练效果,并且真实的手稿也通常在纸张上绘制。因此,加入随机的纸张背景,并对背景进行随机的缩放、翻转、改变透明度等操作,与画布叠加,使手稿图片更真实。

UIsketcher 方法保存了每个组件的坐标信息、重叠率等各种标注信息,所以不需要人工标注即可得到带有丰富标注信息的数据集,即 UIsketcher 的最终输出,如图 1 中的训练数据集。

3　实验与结果分析

3.1　实验方法

由于本文的最终数据集是由单个 UI 组件生成的,因此,将使用传统方法制作的 UI 手稿数据集的训练效果与其组件拆分后生成新数据集的训练效果进行对比,并采用相同的验证集来验证本文方法生成的数据集的性能表现。

本文进行了两个维度的实验:一是假定完成传统方法工作量的 25%、50%、75%、100%,观察其训练效果;二是设定 2 倍、4 倍、8 倍、16 倍增强来观察增强幅度与数据集训练效果的关系。

3.2　实验数据集

微软 sketch2code 公开的 UI 手稿数据集共提供了 10 种不同分类的 UI 组件,其数据集为人工绘制 UI 原型图于纸张或电子手绘板上,通过扫描以图片文件形式储存并使用 json 储存标注信息。考虑到本文的研究工作是 UI 手稿组件识别的数据集,而非 OCR 手写字体识别任务,在微软 sketch2code 中不涉及文字识别的组件共有 7 种,因此,实验数据集也采用 7 种组件进行实验。

建立如下数据集作为传统方法基准数据集:82 张手稿图用于训练,35 张手稿用于测试;同时,通过删减手稿图的绘制数量来控制人工的工作量,以模拟出不同工作量下数据集的训练效果。

由于最终输出的数据集可能是任何的通用格式,如 coco2007、voc2007 等,但这些数据集都有自己的文件结构和数据标注的标准,因此,开发了一套工具并自定义了一种方便中转为其他数据集标注信息的数据格式。该数据格式将每个组件视为一个目标对象,目标对象包含了组件的尺寸、坐标信息、重叠率、分类等。在数据集生成的过程中会先转为这样的中转格式,再转换为训练所需要的数据集格式,这样方便管理,降低了出错率。

实验选用 YOLOv5 模型作为实验模型,使用 coco2007 格式作为数据集图片格式。

3.3　模型

目前,深度神经网络在目标检测领域快速发展,取得了非常好的效果,其中具有代表性的有两类。一类是以 Faster-RCNN[10] 为代表的双阶段模型,这类模型提供了一系列的待筛选目标区域,并对这些目标区域进行卷积分类,这类算法提取出大量冗余特征,训练过程较为复杂耗时。另一类是以 YOLO[11] 为代表的单阶段模型,这类模型对整张图像同时进行检测和分类。近年来,YOLO 模型在目标检测领域有了长足的发展,尤其是 YOLOv5 模型[12],因其模型小、速度快、准确率高等特点而大受欢迎。对于 UI 手稿的识别来说,YOLOv5 算法在进行端到端识别工作时更加简洁高效。因此,选用 YOLOv5 模型作为实验模型。

YOLOv5 模型包括:YOLOv5s、YOLOv5m、YOLOv5l、YOLOv5x。由于网络 UI 界面手稿具有清晰简洁、无重叠、无遮挡、背景干扰少等特点,因此选择推断速度快、平均精度(*mAP*)高的 YOLOv5s 模型进行训练。

3.4　评估指标

本文采用的评估指标包括准确率 *Precision*、召回率 *Recall*、*mAP*@0.50 和 *mAP*@0.50/0.95。准确率是指在所有预测为正例中真正例的比率,

即预测的准确性。召回率是指在所有正例中被正确预测的比率,即预测正确的覆盖率。*mAP*@0.50是将*IoU*阈值设为0.50时,计算各类别*AP*(average precision)的平均值。其中,目标检测中的交并比*IoU*是指预测框与实际框的交集除以并集。*mAP*@0.50/0.95表示在不同*IoU*阈值(0.50、0.55、0.60、0.65、0.70、0.75、0.80、0.85、0.90、0.95)上*mAP*的平均值。

3.5 实验结果

对每个数据集进行了1 024次迭代的训练并达到收敛,具体训练情况如图4所示,其中,Baseline表示传统方法100%工作量。图4(a)为选取的25%工作量在不同随机增强幅度下的训练表现,图4(b)为选取的75%工作量在不同随机增强幅度下的训练表现,可以看到随机幅度和训练效果大致为正相关。表1为不同工作量和增强幅度的训练效果。当工作量达到传统方法的75%时,不论随机幅度多少,其准确率均大于传统方法。在75%工作量8倍增强的情况下,4个评价指标均超过了传统方法。而即使仅有25%工作量的情况下,训练效果也未有明显的下滑。因此,对于想要验证想法的研究人员,仅仅需要完成25%的工作量即可进行实验,而对于采用本文方法制作的用于正式使用的数据集,在工作量仍然低于传统制作方式的情况下,可以进一步提升模型的训练效果。

除此之外,当固定增强幅度的时候,并非工作量更高的数据集的训练效果更好。如图4(c)所示,当训练不足2^4次迭代时,75%工作量所带来的训练效果明显好于100%工作量的训练效果。

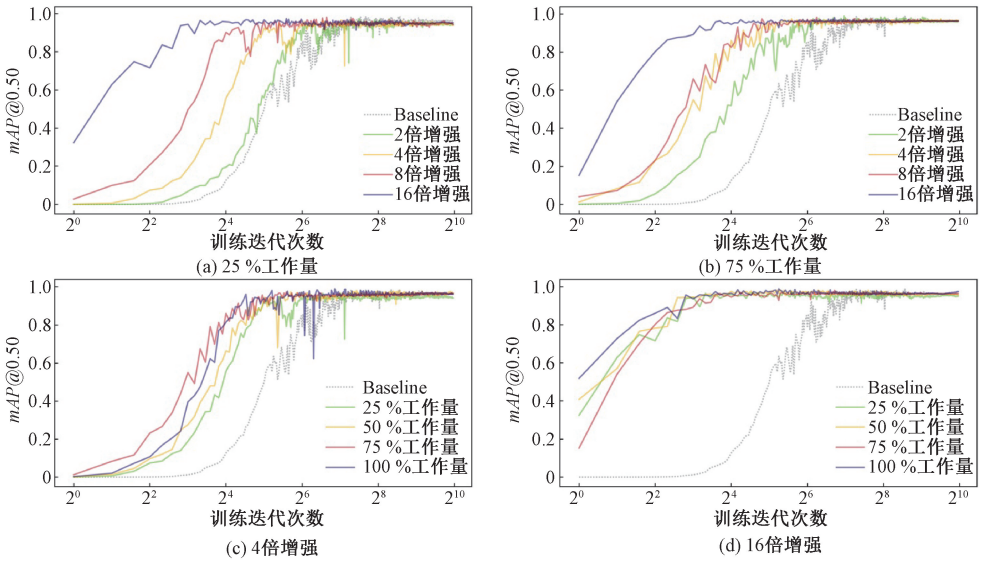


图4 不同迭代次数下的训练情况

Figure 4 Training processes in different epoch

表1 不同工作量和增强幅度的训练效果

Table 1 Results with different workload and enhancement

工作量	*Precision*	*Recall*	*mAP*@0.50	*mAP*@0.50/0.95
传统方法100%工作量	0.973 8	0.963 7	0.960 7	0.781 2
25%工作量2倍增强	0.941 3	0.941 3	0.943 9	0.744 6
25%工作量4倍增强	0.955 1	0.946 8	0.946 2	0.750 6
50%工作量2倍增强	0.966 7	0.954 5	0.958 7	**0.785 7**
50%工作量4倍增强	0.968 7	0.956 1	**0.965 4**	**0.783 6**
75%工作量2倍增强	**0.975 3**	0.961 9	0.960 6	**0.784 4**
75%工作量4倍增强	**0.974 4**	0.957 6	**0.963 2**	**0.787 9**
75%工作量8倍增强	**0.980 3**	**0.969 6**	**0.962 9**	**0.787 8**
75%工作量16倍增强	**0.981 2**	**0.968 3**	**0.963 1**	0.769 8
100%工作量2倍增强	**0.975 4**	**0.966 0**	**0.962 9**	**0.797 8**
100%工作量4倍增强	**0.979 8**	**0.969 2**	**0.964 2**	**0.794 0**
100%工作量8倍增强	**0.982 2**	**0.968 5**	**0.963 3**	**0.793 1**
100%工作量16倍增强	**0.982 3**	**0.973 6**	**0.967 3**	**0.788 1**

25%工作量的训练效果在 2^7 次迭代之前甚至优于使用传统数据集制作的 100%工作量的训练效果。因此进一步进行实验,观察在更高的随机增强幅度下,是否会出现这样的情况,结果如图 4(d)所示。

同样,在图 4(d)中仍然出现了类似的趋势,即低工作量的数据集在训练量较少的时候可以得到比传统方法制作的数据集更高的 *mAP*。这可能是由于超低工作量的数据集组件变化较少,因此 AI 模型可以更快地学习其特征,从而更快拟合,但这并不能提高其最终精度,因为鲁棒性不足。在这里还有个附加条件,即虽然数据集的绘制工作量不同,但是相同的增强幅度下,数据集的数据量几乎是一致的,同时它们的验证集也是一致的。

4　结论

(1)通过本文提出的 UI 手稿数据集制作方法,在仅需传统方法 25%的人工工作量情况下,仍能保持与传统方法制作的数据集相近的训练准确率;在达到传统方法 75%的人工工作量情况下,可超过传统方法制作的数据集的训练准确率。因此本文提出的制作方法对人工的依赖显著少于传统方法。

(2)相较于传统方法,本文提出的方法减少了数据集的标注环节,不需要专门设计不同布局样式的 UI 界面,只需要绘制者根据要求绘制单个的组件即可,通过减少定制化程度,显著降低了人工成本。

本文方法可以为 UI 手稿识别、手稿生成等领域的研究人员提供新的数据集制作思路以及数据集的增强方法。在下一步工作中,将考虑改进优化由组件库到生成最终数据集的处理方法,从而实现更佳的训练效果。

参考文献:

[1] 魏宏彬, 张端金, 杜广明, 等. 基于改进型 YOLO v3 的蔬菜识别算法[J]. 郑州大学学报(工学版), 2020, 41(2): 7-12, 31.
WEI H B, ZHANG D J, DU G M, et al. Vegetable recognition algorithm based on improved YOLOv3[J]. Journal of Zhengzhou university (engineering science), 2020, 41(2): 7-12, 31.

[2] 楼豪杰, 郑元林, 廖开阳, 等. 基于 Siamese-YOLOv4 的印刷品缺陷目标检测[J]. 计算机应用, 2021, 41(11): 3206-3212.
LOU H J, ZHENG Y L, LIAO K Y, et al. Defect target detection for printed matter based on Siamese-YOLOv4[J]. Journal of computer applications, 2021, 41(11): 3206-3212.

[3] 丁明宇, 牛玉磊, 卢志武, 等. 基于深度学习的图片中商品参数识别方法[J]. 软件学报, 2018, 29(4): 1039-1048.
DING M Y, NIU Y L, LU Z W, et al. Deep learning for parameter recognition in commodity images[J]. Journal of software, 2018, 29(4): 1039-1048.

[4] ALES Z, LUKÁS P, ANTONÍN R. Sketch2Code: automatic hand-drawn UI elements detection with faster R-CNN[EB/OL]. (2020-09-22)[2021-04-12]. http://ceur-ws.org/Vol-2696/paper_82.pdf.

[5] WIMMER C, UNTERTRIFALLER A, GRECHENIG T. SketchingInterfaces: a tool for automatically generating high-fidelity user interface mockups from hand-drawn sketches[C]//32nd Australian Conference on Human-Computer Interaction. New York: ACM, 2020: 538-545.

[6] JOÃO S F, ANDRÉ R, HUGO S F. Automatically generating websites from hand-drawn mockups[C]// International Conference on Computer Vision Theory and Applications. Vienna: VISAPP, 2021: 48-58.

[7] FICHOU D, BERARI R, BRIE P, et al. Overview of the 2020 imageCLEFdrawnUI task: detection and recognition of hand drawn website UIs[EB/OL]. (2020-09-22)[2021-04-12]. http://ceur-ws.org/Vol-2696/paper_245.pdf.

[8] CHEN J S, XIE M L, XING Z C, et al. Object detection for graphical user interface: old fashioned or deep learning or a combination[C]//Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2020: 1202-1214.

[9] GHIASI G, CUI Y, SRINIVAS A, et al. Simple copy-paste is a strong data augmentation method for instance segmentation[C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2021: 2917-2927.

[10] REN S Q, HE K M, GIRSHICK R, et al. Faster R-CNN: towards real-time object detection with region proposal networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(6): 1137-1149.

[11] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 779-788.

[12] ULTRALYTICS. YOLOv5[EB/OL]. [2021-04-12]. https://github.com/ultralytics/yolov5.

(下转第 14 页)

An Improved Seagull Optimization Algorithm with Learning

WANG Peichong[1,2], YIN Xinjie[1,2], LI Lirong[3]

(1. School of Information Engineering, Hebei GEO University, Shijiazhuang 050031, China; 2. Laboratory of AI and Machine Learning, Hebei GEO University, Shijiazhuang 050031, China; 3. School of Art, Hebei GEO University, Shijiazhuang 050031, China)

Abstract: To overcome the weakness of slow convergence, prematureness and low accuracy of seagull optimization algorithm (SOA) in solving high-dimensional problems, an improved SOA with learning (ISAOL) was proposed. A migration operator based on the difference between the $\boldsymbol{X}_i$ and the $\boldsymbol{X}_m$ was designed, this could make $\boldsymbol{X}_i$ search wider solution spaces in early stage, and a nonlinear adaptive parameter A was introduced to ensure the algorithm suitable for the search of solution space of complex problems, which could prevent the algorithm from falling into local optimum too early. In the later stage, some elite individuals executed opposition based learning(OBL) to intensify the exploration of the space around the global optimal individual to improve the accuracy of solution. Ten unconstrained test functions in CEC2017 were selected to test the performance of the ISOA and compared with HPSO-TS, V-DVGA, DADE, CMA-ES and others. Testing results of this experiments showed that the ISOAL had higher accuracy and stability than other algorithms. Finally, experiments was carried out by using the tension spring problem. The results showed that the total cost of the spring , the coil diameter and the average diameter of the spring obtained by the ISOAL were reduced by 3.5% ,5.7% and 3.5% than SOA, respectively. ISOAL had the attributes of fast convergence, high accuracy and robustness, fitting to solve higher dimensional function optimization problem and engineering optimization problems with constraints.

Keywords: seagull optimization algorithm; learning mechanism; nonlinear parameter; opposition-based learning

---

(上接第7页)

An Efficient Approach to Creating Hand-Drawn Dataset for UI Manuscript Recognition

YANG Qi[1], LIU Mugeng[2], MA Yun[3]

(1. School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen 518055, China; 2. Department of Computer Science, Peking University, Beijing 100871, China; 3. Institute for Artificial Intelligence, Peking University, Beijing 100871, China)

Abstract: UI manuscript recognition is one of the important applications of image object detection in the area of software engineering. Due to the significant difference between UI manuscript images and natural images where UI manuscript images usually need to be drawn manually, it is difficult to build UI manuscript dataset for deep learning because of the dependency on tremendous manual efforts. To address the issue, in this study an approach called UIsketcher was proposed to efficiently generate UI manuscript dataset based on optimizing the current workflow. In UIsketcher, users should just draw some basic elements without labeling, and then the dataset could be automatically generated for training deep learning model. According to the experiment with UIsketcher, only 25% drawing workload of the traditional methods could get the similar training results. If the workload was 75%, the final accuracy was even better than that of traditional methods.

Keywords: intelligent software developing service; UI manuscript; recognition; object detection; dataset; data enhancement