基于可视水印检测识别的数字媒体溯源应用系统 *

刘子豪
中国传媒大学 信息与通信工程学院，中国，北京，100024

摘　　要

一直以来，图像和视频都是人们最常接触的数字媒体，随着近年来版权意识的发展，国内数字媒体版权保护产业逐渐兴起。尽管目前国内外通过嵌入不可见水印来保护版权的应用已十分广泛，然而目前绝大多数用户生成内容（UGC）平台还是保留了传统嵌入可视水印的做法，这种方法可以使人方便地识别出其来源，但面对信息时代的海量数据，如何快速、智能地溯源成为了一个值得研究的问题。因此，本文提出一种基于可视水印检测识别的数字媒体溯源应用系统，并构建了一个新的大规模常见水印图像数据集（Large-scale Common Watermark Dataset, LCWD）。本文提出的系统能够有效检测网络上常见的图像和视频中的可视水印、识别出其来源并为用户提供来源的相关信息。本文代码已开源于 GitHub，数据集开源于 Kaggle。

关键词：数字水印，媒体溯源，版权保护

1　引言

物理水印技术最早在 1282 年就出现于意大利的法布里亚诺 [1]，当时的人们通过改变纸张的厚度来嵌入水印图案，这项技术被广泛用于邮票、纸笔、书籍和重要文件的防伪和溯源。在数字媒体出现之后，人们也把水印概念引入到了图像上，发明了数字水印。和物理水印一样，数字水印一般需要尽可能少的影响到内容，而把内容提供者的 logo 放在图片的某个地方成为了最流行的方法，直到现在也十分常见。同时，一些学者 [2] 也发明出了肉眼无法看出来的不可见水印，这种水印不会影响内容，同时使用对应的提取算法也能恢复出水印。随着多年来的发展，不可见水印的鲁棒性、不可感知性、容量都得到显著的提升，其在数字内容版权保护产业的应用也十分广泛。然而，目前许多种类的数字媒体（包括电视节目、网络视频等）为了让观看者也能肉眼认识到媒体来源，还是保留了嵌入可视的数字水印的做法。并且，目前嵌入不可见水印的算法有很多，例如基于 LSB（Least Significant Bits）和 DCT（离散余弦变换）的方法 [3]，而这些方法又有很多种变种，要提取出水印，就必须使用嵌入水印相对应的提取方法，这导致在未知水印嵌入的算法时，溯源几乎不可能。而嵌入可视水印的图片就不存在这个问题，并且其鲁棒性也比大多数不可见水印更高。所以，利用可视水印来进行数字媒体溯源在当下的环境是具有可行性的。

*本应用系统将作为田佳音老师、杨成老师的媒体安全技术课和于瀛老师的计算机视觉课的结课设计，该设计交叉融合了这两门课程以及数字媒体技术系其他课程的知识，同时，在撰写两门课的结课论文（设计书）中，作者将根据要求分别体现不同的重点与特色，以避“一稿多投”之嫌。

图 1: 系统总框架图。水印检测模型使用 YOLOv5，并使用本文构建的水印图片数据集（Watermark Dataset）进行微调训练，带有水印的图片或视频被送到模型中进行检测，检测到的水印区域经过进一步增强被提取出来，最后利用水印中的信息进行溯源。

检测图片中的水印是一个具有挑战性的任务，因为（1）水印图种类多样，（2）嵌入的位置、透明度多样，（3）水印与背景颜色相似时较难分辨，（4）水印嵌入数量多样。检测出水印后，识别水印来源充满挑战性：（1）内容来源多样，较难构建一个完整的来源数据库，（2）一些水印中会包含详细信息，如新浪微博图片水印中包含上传用户的 ID，此时能识别到用户级别最佳。

为解决以上问题，本文提出了如图 1 所示的系统，系统总共分成三大模块：（1）首先是基于 YOLOv5[4] 的水印检测模块，将水印的检测看做目标检测任务，并使用本文构建的 LCWD 数据集进行微调训练。数据集包含 60,000 张图片和对应的标注，每张图片包含一种水印。数据集的原始图片是 ImageNet 的一个子集，本文通过以不同方式添加水印生成了这个数据集，其详细构建方式可见第三章。（2）其次是水印增强提取模块，这个模块将背景与水印分离，以减少对下一个模块的干扰，分离的方式基于文献 [5]。（3）最后是水印溯源模块，这个模块通过百度 AI 开放平台的文字识别和品牌 Logo 识别来提取出水印中蕴含的信息来源，并结合搜索引擎进一步溯源。

本文的贡献点总结如下：

- 本文提出了一种新颖的基于可视水印检测识别的数字媒体溯源应用系统，该系统能通过识别任意图片或视频中的可视水印来追溯媒体的起源，输出起源的网页链接、简介等信息。
- 本文构建了一个大规模常见水印图像数据集，并提供了使用任意图片构建水印图片数据集的方法。本文最终使用的数据集包含 60,000 张合成的水印图像和对应的精确标注。
- 本文系统经过了来自网络的真实数据检验，其能够有效检测和识别各类水印。

2 相关工作

2.1 可视水印的去除

关于提取水印的研究和去除水印的研究常常放在一块讨论，本质上说，两者都是企图将水印层和背景层分开。传统要手动去除水印一般使用 Adobe Photoshop 软件，这种方式比较灵活但费时费力。文献 [6] 和文献 [7] 提供了半自动的方法，但用户仍然需要手动选择出水印区域。随着深度学习的发展，去除水印的任务可以看做是一个图像到图像的任务，文献 [8] 使用了类似 UNet[9] 的框架，文献 [10] 使用了多任务学习，同时进行水印检测、水印提取和水印去除的任务。然而，深度学习不可避免的带来了许多参数和计算量，同时也需要更多的数据。所以，本文借鉴文献 [5] 的方法，使用了一种快速且有效的方法提取水印。

2.2 水印图片数据集

目前存在两个开放的水印图片数据集：LVW（Large-scale Visible Watermark Dataset）和 CLWD（ Colored Large-scale Watermark Dataset）。

LVW[11]：总共包含 60,000 张图片，其中训练集的 48,000 张图片由 64 种黑白水印图案生成，验证集的 12,000 张图片由另外 16 种黑白水印图案生成，背景图选自于 PASCAL VOC2012 数据集 [12]。

CLWD[13]：总共包含 70,000 张图片，其中训练集的 60,000 张图片由 160 种彩色水印图案生成，验证集的 10,000 张图片由另外 40 种彩色水印图案生成，背景图同样选自于 PASCAL VOC2012 数据集 [12]。

这两种数据集虽然包含了大量的图片和水印素材，但是两者生成的图片和平常在网络上能看到的水印图不太一样：CLWD 数据集的水印图案多为国外公司 logo，并且生成的图像中水印面积占比非常大，实际中水印占比一般较小，如图 2 右上角所示；LVW 数据集的水印图案虽然是国内常见公司的 logo，但是 logo 数量是固定的，而现实中的水印常常不仅包含平台的 logo，还包含用户的 ID。所以，本文新构建了一个更贴合国内网络图像实际情况的大规模常见水印图像数据集，如图 2 下半部分所示。

3 算法步骤

3.1 环境配置

本应用系统基于 Python3.8 开发，于 `Ubuntu 20.04.4 LTS`和 `Windows 10`上运行测试成功，其中 YOLOv5 模型在 `Ubuntu 20.04.4 LTS`系统使用 4 张 `NVIDIA RTX 2080Ti`进行训练。主要使用的 Python 第三方库有 `PyTorch`, `mmcv`, `OpenCV`, `matplotlib`, `Pillow`, `skimage`, `scrapy`。搜索引擎相关代码基于 GitHub: zhu733756/searchengine。本系统还使用百度 AI 开放平台（百度大脑）的 logo 识别与文字识别 API。

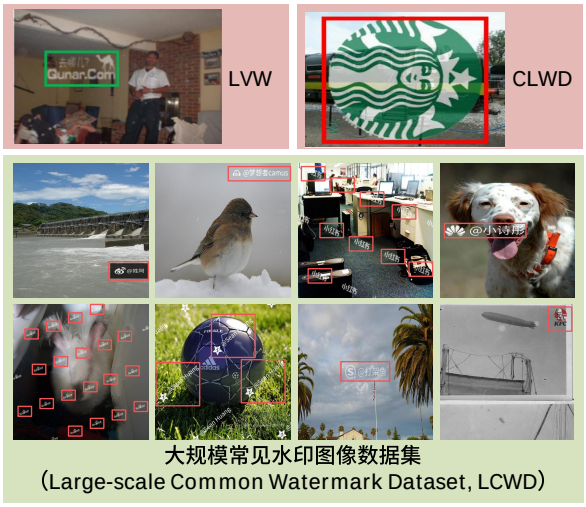

图 2: 数据集对比（建议放大观看）

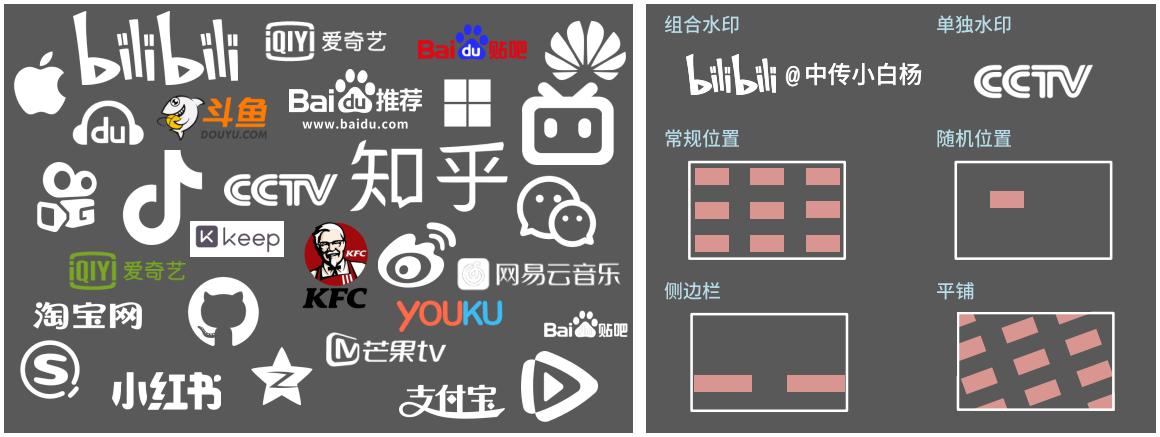

图 3: 数据集构建示意图

3.2 数据集构建

本文构建了一个大规模常见水印图像数据集（Large-scale Common Watermark Dataset, LCWD），总共包含 60,000 张图像和对应的水印检测框标注，类似于 LVW[11] 的方法，本文分成训练集和验证集两部分，分别包含 48,000 和 12,000 张图片。

如图 3 左，我们总共选取 30 个 logo 作为基础水印，其中一半是需要进行组合的水印，另一半是单独的水印，如图 3 右上。组合水印是将基础 logo 放在左边，右边是随机选取的用户名，为了使选取的用户名更接近真实情况，本文使用 Kaggle 豆瓣电影评论数据集中爬取的用户名作为随机抽取的对象。同时，本文还根据真实情况制定了几种水印位置模式（如图 3 右下），常规位置符合大多数的水印图片，这些图片中水印一般出现在边角或者正中央，而随机位置对应的是少数水印（如视频中飘动的水印），除此以外，一些图库也会使用类似侧边栏（图 2 上排第一个）和平铺的模式，其中平铺模式会在一张图片中出现多个水印，且可能带有一定的旋转角度。本数据集每张图片只有一**种**水印，在常规位置模式、随机位置模式、侧边栏模式，图片中只会有一**个**水印，而在平铺模式中，可能会出现属于**同一种**的**多个**水印。数据集只对完整的水印进行标注，超出画面的、不完整的水印将没有标注，图 2 下排前两个为平铺模式。生成数据集时，

图 4: 水印检测结果图

常规位置模式和随机位置模式的概率为 0.6，侧边栏和平铺模式的概率都为 0.4，每个模式中组合水印和单独水印以同样概率出现，除了侧边栏模式只会出现组合水印。

本文背景图片的选择方式参考 LVW 和 CLWD 数据集，从 ImageNet 中选择了一个子集，本文具体使用的背景图片集可从在 Kaggle ImageNet mini 获取到。

3.3 水印检测模型

本文水印检测模型基于 YOLOv5[4]。所有水印都作为同一种类别进行检测，根据任务难度，本文选择了 `YOLO5-Large`模型，将输入图像缩放至 640^2 进行训练，以 `batch size=32`训练 10 轮，之后选择最佳模型。

本文选取了验证集中的部分图片作为示例，如图 4 所示，模型能够正常检测到灰度和彩色的水印（图 4-1,2），能完整检测到平铺模式下的多个水印（图 4-3），还能检测到面积占比很小的水印（图 4-4），一些人眼难以察觉的高难度水印也能轻松检测出来（图 4-5,6）。

以上都是基于图片输入的陈述，假如输入的是视频，模型将抽取 10 帧进行检测。

3.4 水印提取与增强

这个模块截取将检测到的水印截取出来，假如图片中只检测到一个水印，则不再做进一步处理，直接进入下一步，假如检测对象为视频，或者图片中检测到多个水印，则会进一步处理，利用同一种水印在不同位置的共性来增强水印区域。

水印图片的合成一般符合公式 1，其中 $\boldsymbol{J}$ 是合成后的水印图片、$\boldsymbol{W}$ 是水印标识、$\boldsymbol{I}$ 是原图，

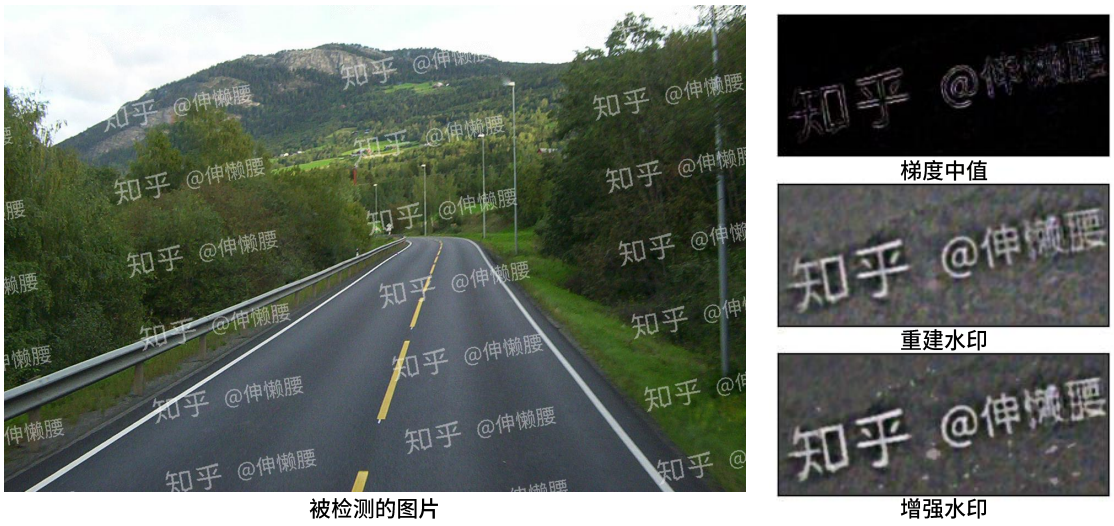

图 5: 水印增强示例图

α 是合成水印的不透明度，α 越小，水印就越不明显，也就越难进行识别。

$$\boldsymbol{J} = \alpha \boldsymbol{W} + (1-\alpha)\boldsymbol{I} \tag{1}$$

然而，对于一个视频中的多帧画面，或者平铺模式合成的水印图片的每个水印区域，$\boldsymbol{W}$ 和 α 一般是固定的，只有 $\boldsymbol{I}$ 进行改动。也就是说，我们能有 k 张水印图片，它们的背景在改变，而水印的内容不改变，而我们期望提取其中不改变的内容。

$$\boldsymbol{J}_k = \alpha \boldsymbol{W} + (1-\alpha)\boldsymbol{I}_k \tag{2}$$

本文从数字图像处理领域的图像降噪中获得了启发——在处理带有椒盐噪声的图像时，中值滤波是一个简单而有效的方法，它考虑图像中某个像素的空域邻域的其它像素，用这些像素的中位数来作为滤波结果。这个方法的原理是：图像中某个像素周围一般是和它相似的像素，而噪声是叠加在其上的以一定概率变化的数值，所以对于这个邻域内的每个像素，噪声是变化的，而像素值是看作不变的，此时用中位数表示结果就能找出邻域内原本的那个不变值。

而在水印图像的模型中也一样，多张水印图片中，水印内容不改变，背景图像改变，而我们要提取的恰好是那个不变的值，于是，我们参照文献 [5] 的做法，对多幅水印图像求取梯度 $\nabla \boldsymbol{J}_k$，然后对于每一个位置，求出其在不同水印图像中的中位数来表示成水印的梯度 $\nabla \widehat{\boldsymbol{W}}$（如图 5 右第一个）。最后，使用泊松重建（Poisson reconstruction）将水印的梯度重建为水印 $\widehat{\boldsymbol{W}}$（如图 5 右第二个）。

$$\nabla \widehat{\boldsymbol{W}}(p) = median_k(\nabla \boldsymbol{J}_k(p)) \tag{3}$$

$$\widehat{\boldsymbol{W}}(p) = reconstruct(\nabla \widehat{\boldsymbol{W}}(p)) \tag{4}$$

对于重建出来的图像，本文进一步提高其对比度：先使用 OTSU 阈值分割 [14] 对图像的三个通道分别进行二值化，之后选择这三个二值图像的并集作为感兴趣区域，提高这个区域的亮度，从而增强重建图像（如图 5 右第三个）。

3.5 水印溯源

水印溯源依靠百度 AI 开放平台和必应搜索引擎，系统将会结合 logo 识别和文字识别的结果得到来源，并通过搜索引擎获取链接和详情，具体算法可见伪代码 1。

Algorithm 1: 溯源算法

```
Data: 水印区域 watermark
Result: 水印来源 source；链接 link；详细情况 detail
初始化 BaiduAPI, BingAPI;
logo_result = BaiduAPI.detect_logo(watermark);
ocr_result = BaiduAPI.ocr(watermark);
if logo_result 有结果且置信度高于 0.7 then
    source = logo_result;
    if ocr_result 有结果 then
        link, detail = BingAPI.search(ocr_result);
    else
        link, detail = BingAPI.search(logo_result);
    end
else
    link, detail = BingAPI.search(ocr_result);
    source = ocr_result;
end
```

4 算法检验

根据上文所述方法，选取了一些结果见图 6。图 6-1 和图 6-2 分别是从知乎和微博随机选取带有水印的图片的检测识别结果。可以看到，对于第一张图片，系统准确提取出了水印的区域，并识别出了知乎的 Logo，通过搜索引擎定位到了对应用户的知乎主页。对于第二张图片，系统没有受到左边一列字的干扰，成功定位到了位于图片下方的水印，并成功溯源。图 6-3 是一个商业图库的带有多个水印的样图，系统成功定位到了大部分的水印区域，并通过水印增强使轮廓更加明显，最后成功溯源。图 6-4 是哔哩哔哩网站上随机选取的一个视频的检测结果，系统也通过多帧视频提取出了十分清晰的水印区域，并成功溯源到用户的哔哩哔哩页面。

图 6: 系统溯源效果图（建议放大观看）

5 结论

本文在数字版权保护领域独辟蹊径，新颖地针对可视水印进行检测、识别与溯源，构建了一个轻量快速的应用系统。在进行实验之后，本文认为针对可视水印的溯源系统在当下具有很强的实用性，可与不可见数字水印的检测结合在一起进行溯源和版权保护。此外，本文还构建并开源了一个大规模的常见水印图像数据集，以促进相关领域后续的研究工作。

6 最后的话

首先感谢田佳音老师、杨成老师和和于瀛老师这个学期的课程，更感谢老师们选择了大作业这种更灵活且能充分实践已有知识的考核形式。

在一看到两门课的结课要求时，我就在想，有没有一种方法能够交叉融合两门课所学的知识呢，于是便自作聪明地完成了这个拙作。在进行这个项目的时候也抱着想省事的心态，蹭了 YOLOv5 的模型、蹭了 github 上一个搜索引擎的爬虫代码、蹭了五年前没用深度学习的 CVPR 论文、还蹭了百度的 AI……虽然如此，但是在进行的过程中还是遇到了很多麻烦，也敲了七百多行的代码，最后为了锻炼学术能力还尝试着用 LaTeX 写了这篇论文……总之，自我感觉良好，这是个能让我自豪地承认“这是我写的”的大作业 🤗!

参考文献

[1] Philip B. Meggs. A History of Graphic Design (Third ed.) John Wiley & Sons, Inc., 1998.

[2] Anatol Z Tirkel et al. “Electronic watermark”. In: Digital Image Computing, Technology and Applications (DICTA’ 93) (1993), pp. 666–673.

[3] 刘瑞祯 and 谭铁牛. “数字图像水印研究综述”. In: 通信学报 21.8 (2000), pp. 39–48.

[4] Glenn Jocher. Ultralytics/yolov5: Yolov5 in PyTorch ; ONNX ; CoreML ; TFLite. May 2020. URL: https://github.com/ultralytics/yolov5.

[5] Tali Dekel et al. "On the effectiveness of visible watermarks". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 2146–2154.

[6] Chun-Hsiang Huang and Ja-Ling Wu. "Attacking visible watermarking schemes". In: IEEE transactions on multimedia 6.1 (2004), pp. 16–30.

[7] Soo-Chang Pei and Yi-Chong Zeng. "A novel image recovery algorithm for visible watermarked images". In: IEEE Transactions on information forensics and security 1.4 (2006), pp. 543–550.

[8] Jing Liang et al. "Visible Watermark Removal via Self-calibrated Localization and Background Refinement". In: Proceedings of the 29th ACM International Conference on Multimedia. 2021, pp. 4426–4434.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: arXiv e-prints, arXiv:1505.04597 (May 2015), arXiv:1505.04597. arXiv: 1505.04597 [cs.CV].

[10] Xiaodong Cun and Chi-Man Pun. "Split then refine: stacked attention-guided ResUNets for blind single image visible watermark removal". In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. 2. 2021, pp. 1184–1192.

[11] Danni Cheng et al. "Large-scale visible watermark detection and removal with deep convolutional networks". In: Chinese conference on pattern recognition and computer vision (prcv). Springer. 2018, pp. 27–40.

[12] Mark Everingham et al. "The pascal visual object classes challenge: A retrospective". In: International journal of computer vision 111.1 (2015), pp. 98–136.

[13] Yang Liu, Zhen Zhu, and Xiang Bai. "Wdnet: Watermark-decomposition network for visible watermark removal". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021, pp. 3685–3693.

[14] Nobuyuki Otsu. "A threshold selection method from gray-level histograms". In: IEEE transactions on systems, man, and cybernetics 9.1 (1979), pp. 62–66.