上海科技大学
ShanghaiTech University

本科毕业论文（设计）

题　　目：图像驱动的平面设计生成

学生姓名：陈博克

学　　号：2020533035

入学年份：2020

所在学院：信息科学与技术学院

攻读专业：计算机科学与技术

指导教师：曹迎

上海科技大学

2024 年 05 月

上海科技大学
ShanghaiTech University

THESIS

Subject: Image-Driven Graphic Design Generation

Student Name: Boke Chen

Student ID : 2020533035

Year of Entrance: 2020

School: School of Information Science and Technology

Major: Computer Science and Technology

Advisor: Ying Cao

ShanghaiTech University

Date: 05 / 2024

上海科技大学
毕业论文(设计)学术诚信声明

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名： 

日期：　2024　年 5　月 9　日

上海科技大学
毕业论文（设计）版权使用授权书

本毕业论文（设计）作者同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海科技大学可以将本毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本毕业论文（设计）。

保　密□，在___年解密后适用本授权书。

本论文属于

不保密☑。

（请在以上方框内打“√”）

作者签名：陈博克　　指导教师签名：曹迎

日期：2024 年 5 月 9 日　　日期：2024 年 5 月 12 日

图像驱动的平面设计生成

摘要

随着数字媒体的迅速发展，日常设计流程中的图像使用量正急剧增加，尤其是在广告、社交媒体和网页设计等领域。传统的图像编辑方法以及图像素材选取需要大量的人工干预，这不仅耗时耗力，而且无法迅速响应设计师对于图像的精确需求。然而计算机科学的跨学科应用为这一工作提供了助力。本文研究和探讨了利用机器学习和深度学习技术以实现图像驱动的平面设计生成。研究主要采用了计算机视觉领域中的显著性检测技术对设计师需要使用的图像部分进行了定位，随后利用图像裁剪技术将其进行切割划分，最后采用 CyberAgent 的 FlexDM 模型对切割下的图片进行设计要素的再分配，从而使得最后生成的设计满足设计者的需求。这一需求主要体现为：在较大的海报中将核心元素提取重组后形成一个更加凝练的小型海报；或反之将较小的海报放大并填充空缺部分。本文在显著性检测部分选用了一种新颖的无监督博弈论显著对象检测算法，无需使用标注的训练数据，这样可以减少缺少设计图案训练数据的不稳定性。同时在图像裁剪方面也选取了一个新模型，将图像裁剪形式化为一个列表排序问题的同时采用一种改进的视图采样。最后在生成时采用的多模态文档模型将输入的图像转化成一个多模态向量文档，并根据文档的内容进行设计元素的填充和调整。结果表明，这些新模型的利用对输入图像的利用效率，以及生成的平面设计图像的美学价值都有着积极的作用。

关键词：平面设计，深度学习，显著性检测，图像裁剪，多模态文档模型

Image-Driven Graphic Design Generation

Abstract

With the rapid development of digital media, the use of images in daily design processes is dramatically increasing, particularly in fields such as advertising, social media, and web design. Traditional image editing methods and the selection of image materials require substantial manual intervention, which is not only time-consuming and labor-intensive but also unable to quickly respond to designers' precise image needs. However, the interdisciplinary application of computer science provides a solution for this task. This paper investigates and explores the implementation of image-driven graphic design generation using machine learning and deep learning technologies. The research primarily employs saliency detection techniques from the field of computer vision to locate the parts of the images that designers need. Subsequently, image cropping techniques are utilized to segment these parts. Finally, CyberAgent's FlexDM model is employed to reallocate the design elements of the cropped images, ensuring that the final generated designs meet the designers' requirements. These requirements mainly manifest as extracting and reorganizing core elements from a large poster to form a more concise small poster, or conversely, enlarging a small poster and filling in the missing parts.In the saliency detection section, this paper selects a novel unsupervised game theory-based salient object detection algorithm that does not require annotated training data, thereby reducing the instability caused by the lack of training data with design patterns. For image cropping, a new model is chosen that formalizes image cropping as a list-ranking problem while adopting an improved view sampling method. Finally, in the generation phase, the multimodal document model used converts the input image into a multimodal vector document and fills and adjusts the design elements based on the document's content.The results indicate that the use of these new models positively impacts the efficiency of utilizing input images and the aesthetic value of the generated graphic design images.

Key words: graphic design, deep learning, saliency detection, image cropping, multimodal document model

目 录

第一章 绪论

在平面设计领域，图像选择和裁剪技术是提升工作效率和设计质量的关键工具。随着数字媒体的广泛应用，设计师面临着海量图像资源的挑战，如何快速而准确地从中挑选和调整图像，成为了一个亟待解决的问题。目前已有不少辅助进行图像裁剪的软件，但是大多需要设计师手动进行选择，并且操作繁琐，效果可能也不尽如人意。而在裁剪过后面对素材库时，业余设计师也会纠结于合适的素材选用。为满足从裁剪到产出一体化的需求，研究人员对这一流程中的每个环节都进行了创新性的研究，尤其是选择时的显著性检测，裁剪时的图像裁剪技术以及最后加入新元素的多模态文档模型。

对于一个设计师而言，他需要的是一个实时高效反应的设计辅助工具，而非一个需要多次进行参数调整或现场输入图片进行学习的模型，因此在自动图像选择和裁剪的过程中，此模型需要实时处理大量的数据和计算，可能会遇到性能瓶颈，因此需要优化算法和模型结构来进行实现，这是对显著性检测技术有所选择的原因。而平面设计所涉及的数据类型的多样性，如文本、图像和向量图形等，则需要最后的模型能够有效整合和处理这些多模态数据，这也是本文将讨论 FlexDM 模型的原因。

1.1 研究背景

在当前平面设计行业的背景下，设计师对于设计软件的要求也较之前有所提高。他们希望在这些工具的辅助下能有效提高工作效率并保证设计质量。而根据输入图像进行自动设计生成可以大幅度减少手动编辑的时间：通过智能算法快速识别和提取输入图像中的关键内容，从而使设计师可以将更多精力投入到创意和艺术表达上。市面上已经有一些软件可以提供类似的根据输入图像进行关键对象裁剪的功能。例如 Adobe Photoshop 提供了基于内容的裁剪工具[1]，该工具可以自动识别图像中的重要内容并尝试在裁剪时保留这些区域。类似地，Canva 和 Figma 这类设计软件也集成了智能裁剪功能，帮助用户快速调整图像尺寸同时保持视觉焦点的完整性。然而，尽管市面上存在这些工具，裁剪出来的结果有时会不尽人意，而且提取出关键对象后并没有相应的设计元素生成和填充功能。因此这一领域的未来研究将持续关注算法的优化，内容的生成以及更广泛的应用场景探索。

1.2 研究问题

为了解决这些设计师在日常设计体验中共同的痛点，本文的主要内容聚焦于应用深度学习和计算机视觉技术实现图像驱动的平面设计生成，旨在辅助设计师基于感兴趣的图像快速地生成设计原型，从而启发其创意过程和提升设计效率。具体而言，课题的核心内容和技术指标包括：1. 图像内容理解与分析：利用深度学习模型对输入的图像内容进行深入分析，能够识别和理解图像中的关键元素（如人物、物体、风景等），并根据设计需求对这些元素进行智能分类和标注。 2. 自动图像裁剪技术：利用算法自动确定图像中的重要内容区域，并根据预设的设计标准（如比例、构图要求等）进行裁剪，确保裁剪后的图像在视觉效果上符合设计要求。 3. 设计方案空缺部分填充： 利用多模态文档模型对非图像内容进行预测和填充并输出。将这三者按序统合之后便能开发出一套高效、智能且用户友好的设计编辑和生成工具，解决部分平面设计师的需求。

第二章 图像内容理解与分析

在平面设计中，显著性检测技术在自动图像选择和裁剪的应用中扮演了核心角色。显著性检测是通过算法分析图像，自动识别图像中最能吸引观看者注意的视觉元素或区域。这种技术可以帮助设计师在大量图像中迅速定位到最具影响力的内容，从而在设计过程中实现更高的效率和更精确的目标导向。

在计算机视觉领域中，显著对象检测是重要的预处理步骤，其目的是在图像中找出显著的对象，在本文语境下则是找到需要进行裁剪的主题图像。显著性这一特征有助于将有限的计算资源分配给图像中最具有信息性的突出物体而不是处理背景。过去已经有大量的研究试图解决这个问题，并提出了一些方法[2]。然而，对于平面设计类的图像而言，可能图像中会包含有多个对象或存在高背景杂乱的问题，对于这种图像，显著性检测仍然是一项具有挑战性的任务。

2.1 技术背景

本文选取了一种无监督博弈论显著对象检测算法[3]，这种算法是将显著对象检测问题转化为了一个非合作博弈，其被称为“显著性博弈”。算法的设计者通过利用多种线索并结合互补特征构建了一个支付函数，而显著性地图则由每个区域在纳什均衡中的策略生成。在此基础上，设计者在探索不同颜色与深度特征之间的互补关系时发现了一种新的迭代随机游走算法，这个算法通过在特征空间之间共享信息，将使用不同特征生成的显著性地图进行组合，从而检测出那些难以检测到的对象。

本文之所以选取该算法，主要是考虑到其经济性。无监督方法不需要训练数据，它们的运作通常需要关于显著对象的先验假设，因此这类算法的性能严重依赖于使用的先验假设的可靠性。以近几年流行的标签传播方法为例[4]，这一算法首先根据一些先验性的知识（如边界背景假设）来选择种子，然后将标签从种子传播到未标记的区域。这种算法在大多数情况下都很有效，但是如果种子选择有误，则会导致结果不准确。譬如将图像边界作为背景种子，若显著对象接触到图像边界，则结果会产生谬误（如图 1 所示）。

图 1 错误种子选取结果（图源[3]）

但是相比于监督方法，基于启发式规则的无监督方法可能在结果上会产生更大的偏差。原因在于监督方法在海量的训练数据中，可以提取出显著对象更具有代表性的特征，比如一些基于深度学习的算法，其深度神经网络可以在训练后掌握高层的语义丰富特征，在面临具有复杂背景的图像时，该算法可以有效检测出期中的显著对象。

对于平面设计这一领域而言，其背景的复杂度并不会太高，但其图源的数量却是巨大的，而且不同设计风格或设计对象（诸如广告、海报、标识等）的图像构成也不大相同，这会导致训练的数据量产生不必要的膨胀。由于平面设计的美学需要，其图像构成往往遵循一些规律，因此直接采用无监督方法会是一种更好的选择。并且该算法的设计者也试图克服其相比于监督方法的缺陷，诸如采用两个独立的先验假设来提高算法的鲁棒性；抑或是利用预训练的全卷积网络来识别复杂场景中的显著物体；以及使用一种迭代随机游走算法对识别的结果进行优化。

2.2 选取原因

2.2.1 模型概述

如上文所说，此算法有三个创新之处，下文将对其逐一阐明。

首先是对于两个独立的先验假设的采用。在发现此算法之前，大多数算法都是将显著性检测的形式转化为对于单个能量函数的优化，该函数包含了一个显著性的先验假设。在这种方法里，不同的图像区域是通过在能量函数中添加项或顺序处理来考虑[5]。所以假如该先验存在问题或错误，则对其能量函数的优化可能会导向一个错误的结果。鉴于个中原因，此算法的设计者为每个超像素都定义了一个特定的收益函数，这个函数结合了包括空间位置先验、对象性先验以及来自其他超像素的支持等多种因素。结合上述这两个独立的先验假设，使得该算法更具有鲁棒性，因为当一个先验假设不适用时另一个仍有可能生效。在这场关于显著性目标的博弈中，该算法中的“注意力游戏”的目标是最大化每个玩家在给定其他玩家策略下的收益，这个目标可以等同于是在对多个竞争性目标函数进行最大化处理。由于该博弈的均衡状态的自动化特性，当某些图像区域由于错误的先验假设而无法被赋予其相应的显著性数值时，优化其他的目标函数便可能有助于为他们赋予正确的显著性数值。这种博弈的想法是非常贴合人脑对于图像的处理的，那些所谓“显著”的对象所呈现出的显著性，都是它们对于吸引我们注意力的竞争。

其次是对于预训练的全卷积网络的采用。深度神经网络常被用来处理复杂任务，它之所以有此能力是因为它能够对于高维语义的丰富特征进行提取和学习。全卷积网络作为深度神经网络的一种典型架构，自然能够有效处理显著性检测的问题。前人已经对该应用有了一定的研究：全卷积网络在大规模数据集（如 ImageNet）上经过训练后已经能够掌握丰富的视觉特征，包括从低层次的边缘和纹理信息到高层次的物体和场景理解[6]。这些特征使得全卷积网络在处理新任务时能够快速适应并提供高质量的特征表示，而不需要从头开始训练模型。因此，利用预训练的全卷积网络进行无监督学习方法能够显著减少计算资源的消耗和时间成本。正如上文所提到的，无监督学习方法通常不需要大量的标注数据，这对于数据获取成本较高的领域尤为重要。平面设计领域便是其中之一。通过使用预训练的全卷积网络可以直接利用其提取的特征，并构建基于这些特征的无监督算法，这不仅提高了开发效率也保证了模型性能的稳定性[7]。此外，预训练的全卷积网络在不同任务中的迁移能力使其成为无监督方法的理想选择。在显著性检测任务中，预训练的全卷积网络能够提供细粒度的特征图，通过这些特征图，算法可以更准确地捕捉到图像的显著区域。例如，一些研究利用预训练的全卷积网络提取特征，然后结合聚类算法或图论方法来实现无监督的显著性检测，并取得了较为理想的效果[2]。

最后便是使用迭代随机游走算法对识别结果的优化。上文所提到的预训练的全卷积网络固然是一个非常良好的选择，但是其包含的丰富的语义信息往往伴随着通过卷积和池化层稀释图像特征的代价。于是该算法的设计者使用了传统的 CIE-Lab 颜色特征作为深度特征的补充信息。为了充分利用这两个互补特征，从而获得更优的检测结果，他们没有直接对深度特征和 CIE-Lab 颜色特征生成的原始结果进行简单的加权处理。相反，他们提出了一种在这两个特征空间之间进行迭代随机游走的算法，以进一步细化显著性图。在每次迭代的过程中，两个特征空间的传播都会受到对方最新输出的影响和制约，从而提高显著性检测的精度。具体而言，他们选择了 FCN-32s 的特征（由于其胜任语义分割一职）以及输入图像的 Conv5 层的特征图作为特征（由于其对外观变化的鲁棒性）。

2.2.2 数据支持

图 2 展示了该算法与其他算法性能的对比：

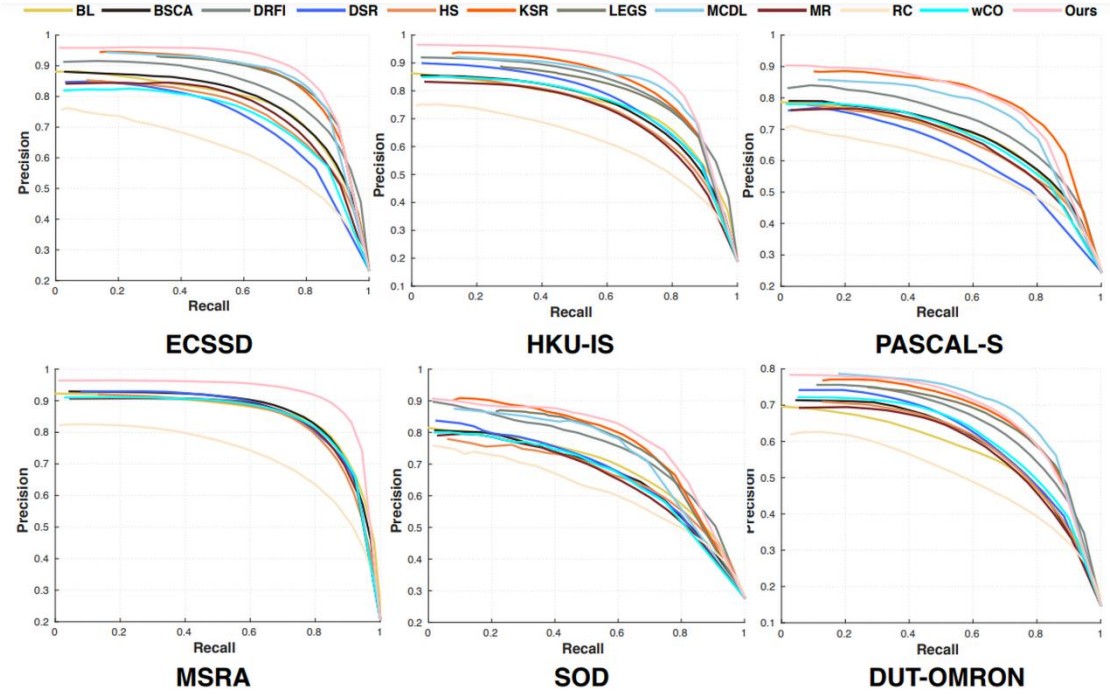


图 2 不同数据集上各种算法的准确度（图源[3]）

由图 2 可见，该算法在大多数数据集上都有着优势性的表现，且在部分数据集上具有巨大的比较优势（MSRA）。

图 3 至图 5 展示了该算法与其他算法处理复杂图片的实例：

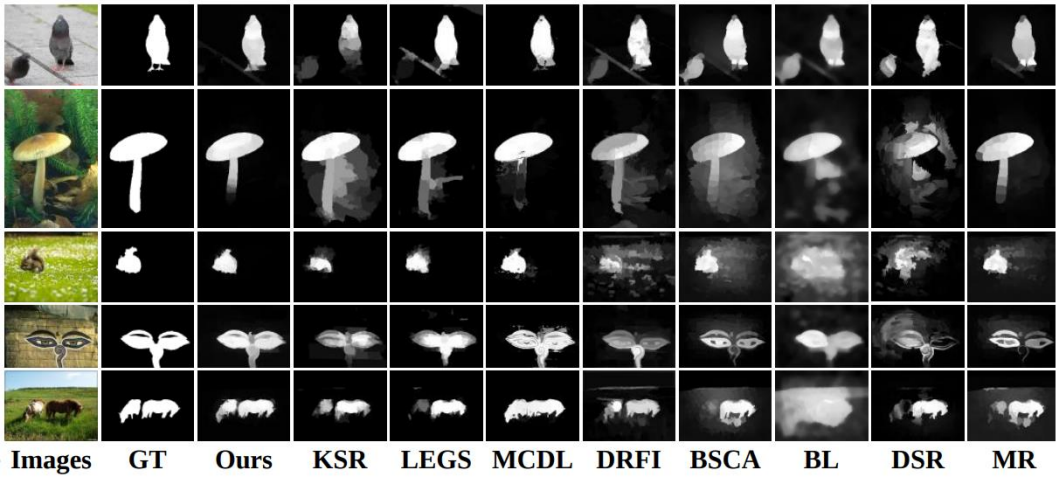


图 3 各种算法处理 ECSSD 数据集结果对比（图源[3]）

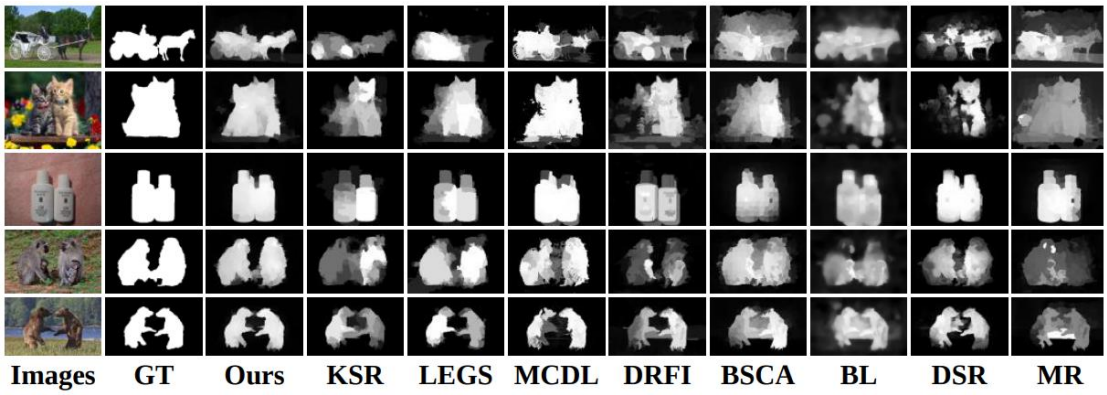


图 4 各种算法处理 HKU-IS 数据集结果对比（图源[3]）

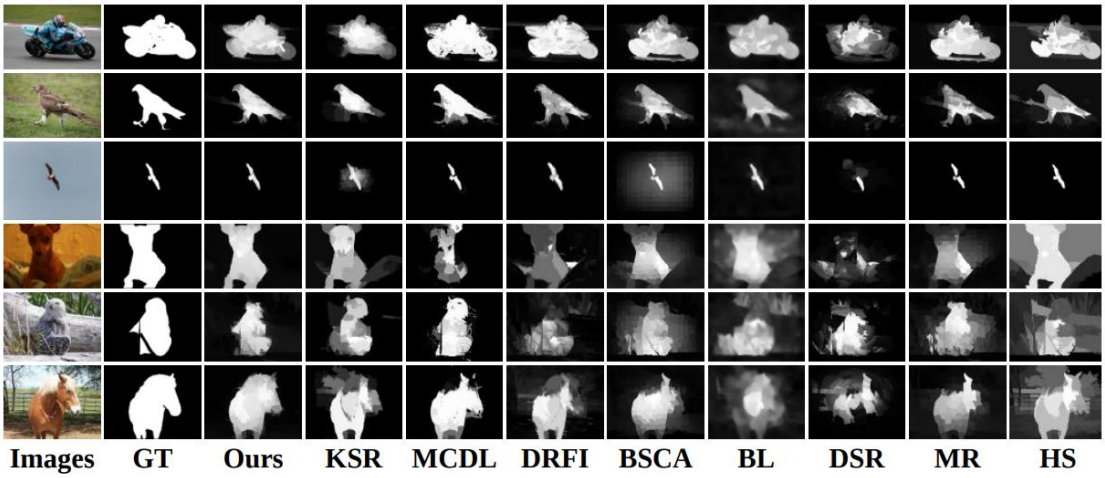


图 5 各种算法处理 PASCAL-S 数据集结果对比（图源[3]）

由此可见该算法相比于其他算法，在处理复杂图片时准确度更高，并不会将场景中的一些具有迷惑性的物体纳入显著对象的范畴，而且在检测较小物体时也有着一些优势。

图 6 至图 11 展示了输入几张平面设计图像后的输出对比：

图 6 此算法处理平面设计图例结果对比（1）

图 7 此算法处理平面设计图例结果对比（2）

图 8 此算法处理一般图例结果对比（1）

图 9 此算法处理一般图例结果对比（2）

图 10 此算法处理一般图例结果对比（3）

图 11 此算法处理一般结果图例对比（4）

由以上数据集测试结果可知，对于一般性图片，即显著性对象与背景区分度较高且不黏合在一起时，该算法的表现十分优异。但是在处理较暗的画面，以及多个显著性目标的图像时则不尽如人意。但是在平面设计领域中，有大量对于色彩运用较为大胆，或是构图较为抽象的设计作品，在对这些作品进行显著性识别时会对模型产生较大的冲击，并且结果都会与设计师所设想的相差较大。

因此该模型对于平面设计图像的适应能力还需要进一步的提高，可能需要更多的具有可识别性的平面设计图像（可识别性指画面对比度正常，且显著性目标不高度抽象）作为数据库进行训练。未来的研究和改进方向应集中于增强模型在复杂场景中的鲁棒性，特别是针对低光照条件和多目标识别的能力。通过引入更多多样化的训练数据、优化特征提取算法以及利用先进的深度学习技术，希望能够显著提升模型的综合表现，使其在各种复杂的设计作品中都能达到设计师的预期效果，从而更好地服务于平面设计领域。

第三章 自动图像裁剪技术

在平面设计中，图像裁剪技术在自动图像选择与裁剪的应用中发挥着不可或缺的作用。图像裁剪通过智能算法分析图像内容，自动确定图像中最关键的区域并进行裁剪，从而突显设计的核心元素。这种技术帮助设计师在处理大量图像时，能够迅速锁定最具视觉冲击力的部分，从而在设计流程中实现更高的效率和精确度。

图像裁剪技术在计算机视觉领域中已经取得了显著的进展，主要目标是从图像中提取最具信息量的区域。在本文的语境下，这意味着识别并裁剪出需要重点展示的设计元素。现有的图像裁剪技术不仅提升了设计作品的视觉吸引力，还能够高效分配计算资源，集中处理图像中的关键部分，而不是背景。然而，这些技术在实际应用中仍面临一些挑战，尤其是在复杂背景和多对象场景中。

当前的图像裁剪技术大多依赖于深度学习和卷积神经网络，这些方法在标准化数据集上表现良好，但在处理多样化、复杂化的设计图像时，效果往往不尽如人意。设计图像常常包含多个重要元素或高度复杂的背景，这对现有的裁剪算法提出了更高的要求。此外，许多算法在识别和裁剪时缺乏灵活性，难以适应不同设计风格和需求。

为应对这些挑战，未来的研究应聚焦于以下几个方面：首先，通过引入多模态数据和丰富的训练样本，提升算法对复杂场景的适应能力；其次，结合显著性检测和语义分割技术，实现更精细的图像裁剪；最后，开发更智能化和人性化的交互工具，使设计师能够更方便地调整和优化裁剪结果。

总体而言，尽管现有的图像裁剪技术在许多应用中已经展示了其强大功能，但其在平面设计领域的潜力尚未完全释放。通过持续的技术创新和方法改进，我们期待未来的图像裁剪技术能够更好地满足设计师的需求，助力创作出更加精美和高效的设计作品。

3.1 技术背景

图像裁剪是一个常见的图像处理手段，这个过程可以去除设计师不需要的区域以便改善整体的构图逻辑。这一技术在平面设计上的应用十分广泛，不仅在 Adobe Photoshop 中有完整丰富的裁剪手段，就连日常生活中的手机照相软件都会配备一个简单的图像裁剪工具。图像裁剪技术与上文所提的显著性检测有着密不可分的关系。通常情况下图像裁剪技术本身也会有一定的显著目标检测能力，不过其性能可能并不仅限于对于显著目标的检测，它会将目标周围的一块区域也留下，因为其主要目标是生成一张想对于原图像较小的、并更具有审美性的图像。

在过去的研究中，基于排名的裁剪方法往往在性能上存在缺陷，诸如使用基于对偶排序的视图查找网络来对图像进行构图并用滑动窗口来裁剪[8]。

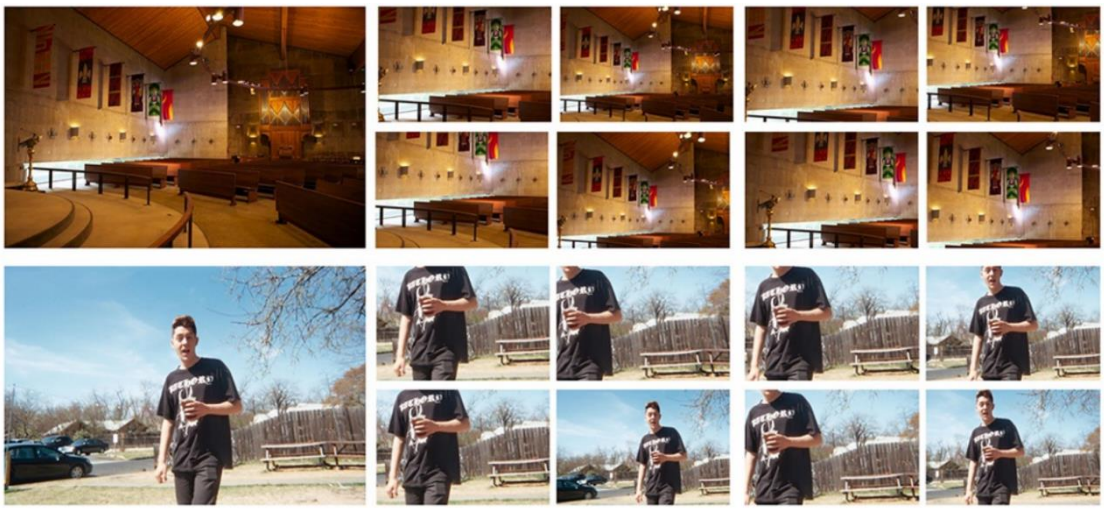


图 12 对偶排序的自动裁剪（图源[8]）

由图 12 可见，单纯使用对偶排序的自动裁剪方法会将显著对象的某一部分裁剪掉，或者干脆就找不到显著对象以及应该裁剪的部分。

这里的主要原因可能有以下两点：首先是对偶训练并不适合图像裁剪这一过程。在进行图像裁剪这一任务时，模型的主要目标是从待选的视图列表中选择最优解，因此从本质上讲，图像裁剪是一个列表排序任务，而非单纯的对偶比较。此外，对偶训练非常地依赖“对偶”的选择，因为当训练数据中样本的分布较为各异时，会导致训练的结果发生偏差，这极大地增加了计算的复杂性，并且使得训练过程不够稳定；其次是从卷积神经网络提取的粗糙特征会影响到模型学习的准确性。以往的方法通常在原始图像或特征图上进行裁剪和变形，然后为每个视图计算排名分数。在图像裁剪中，像素级的精度比对象分类更为关键，而变形操作则会降低这一精度。此外，池化层带来的缩放效应会降低采样分辨率，进而影响组合学习的效果。

因此，为了克服上述的这些问题，本文选用了一种带有精细视图采样的列表排序方法[9]。在精细视图采样的过程中，该算法的设计者提出了一种新的区域兴趣操作，用于提取候选视图的精细特征图。这样便不用再在选择视图对偶时浪费时间，而是直接利用所有标注的视图，使用列表排序损失进行训练。图 13 展示了该算法的简要流程：

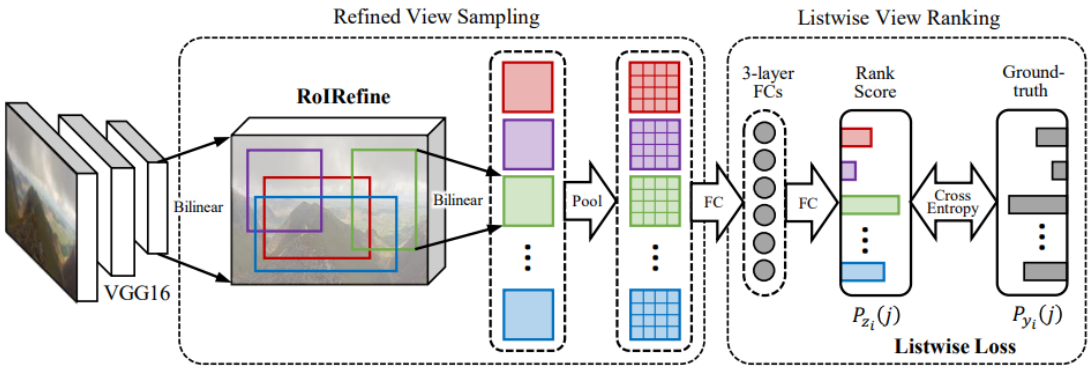


图 13 列表排序方法的图像裁剪（图源[9]）

3.2 选取原因

3.2.1 模型概述

图像裁剪技术在专业摄影中被视为一种非常有前景的自动实现美学构图的方法[10]。目前有两种主流的对图像裁剪区域的划分方式：一种是以注意力模型为主要的建模导向，这种方法即其字面意思，算法将从原始图像中找到视觉上最重要的区域，如视觉显著信息或人脸等；第二种则是以审美模型为建模导向，这种方法强调了裁剪图像给人的审美体验。本文所采用的列表排序算法更接近于第二种，将不同区域进行评分，相当于图像区域对人的审美吸引能力的竞争。

3.2.2 数据支持

这一列表排序算法在不同的数据集下都进行了测试，并且测试结果相较于先前的一些利用对偶方法或是对候选视图进行缩放的方法而言，都要更优一筹。

Methods	Avg. IoU	Avg. Disp
AesRankNet [Kong *et al.*, 2016]	0.4843	0.1400
RankSVM [Chen *et al.*, 2017a]	0.6020	0.1060
VFN+SW [Chen *et al.*, 2017b]	0.6328	0.0982
A2-RL [Li *et al.*, 2018]	0.6633	0.0892
VPN [Wei *et al.*, 2018]	0.6641	0.0858
LVRN (Ours)	**0.7100**	**0.0735**

表 1 不同方法的图像裁剪在 FCDB 数据集上的表现对比（表源[9]）

Methods	Top-1 Max IoU
Fang et al. [Fang *et al.*, 2014]	0.6998
VFN+SW [Chen *et al.*, 2017b]	0.7265
ABP+AA [Wang and Shen, 2017]	0.8100
A2-RL [Li *et al.*, 2018]	0.8204
VPN [Wei *et al.*, 2018]	0.8233
LVRN (Ours)	**0.8434**

表 2 不同方法的图像裁剪在 FLMS 数据集上的表现对比（表源[9]）

从表 1 表 2 可以看出，该算法在不同数据集上的优异表现具有鲁棒性，图像的变化并未对模型的表现产生较大的影响。

表 3 则更加直观地展示了列表排序方法相对于其他的方法，而非仅仅模型的优越性：

Ranking Loss	Avg. IoU
Pairwise + w/o selection (260W+)	0.5355
Pairwise + simple selection (130W+)	0.5568
Pairwise + careful selection	0.6080
Listwise	**0.6204**

表 3 列表排序方法与其他方法在 FCDB 数据集上的表现对比（表源[9]）

表 3 证明，此算法可以适应更宽泛的训练数据集，而不需要设计者亲自对数据集里的图像进行筛选和调整。这对于平面设计的图像而言是一个更有利的竞争条件，因为正如上文在显著性检测部分所言，针对平面设计的图像选择和裁剪会涉及较为复杂抽象的图像，而且图像之间的差别会比较大，难以用几个数据集进行统合概括。

图 14 展示了该算法模型与其他模型在裁剪图片实例时的表现：

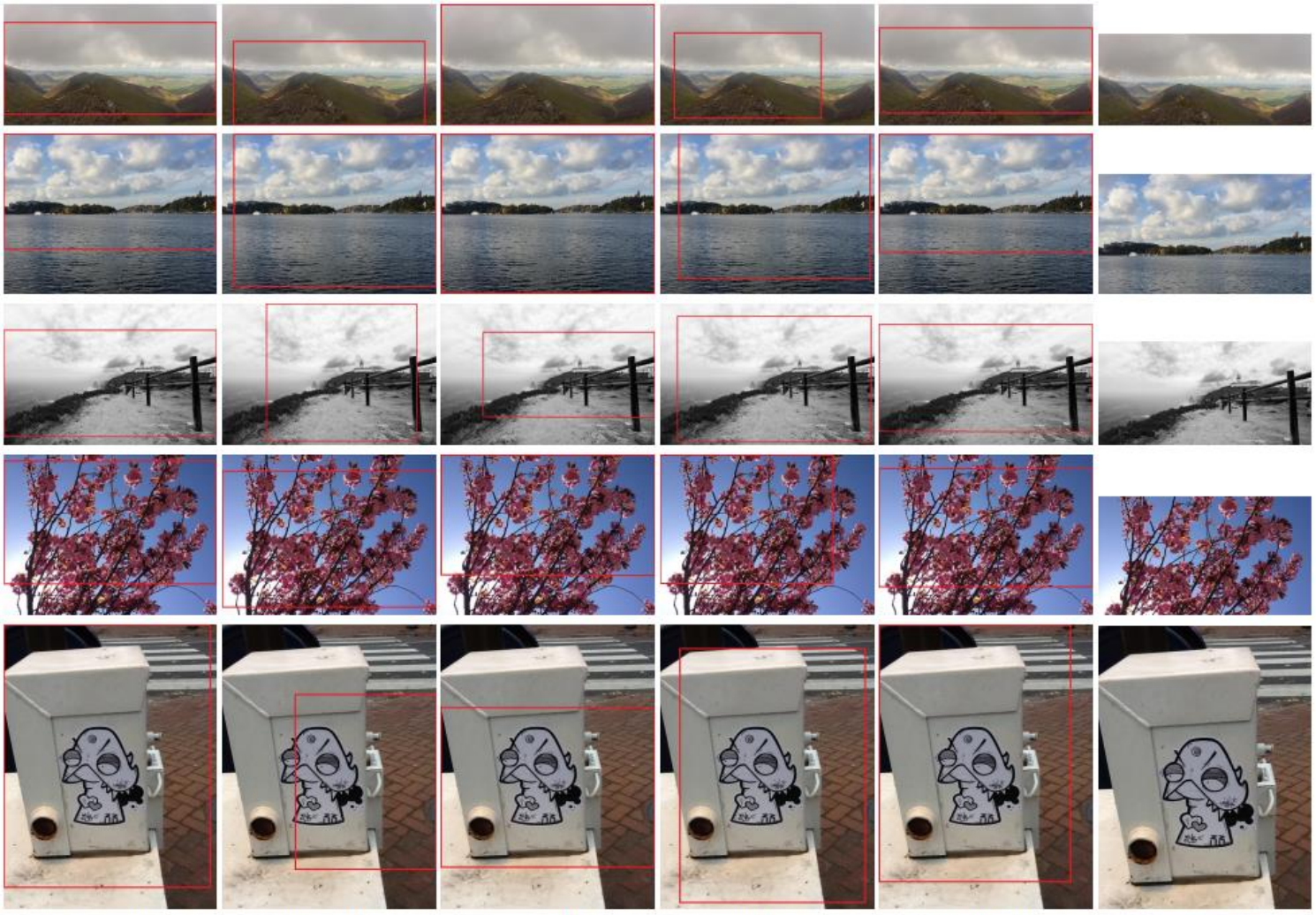


图 14 列表排序方法与其他方法的图像裁剪实例对比（图源[9]）

由图 14 可见，该列表排序方法对不同类型的图像适应度较高，并且裁剪出来的部分基本上包含了显著对象，可能会有多余的一小部分背景被选取，但是不会破坏对象的完整性。

3.3 模型改进

但是这个模型并不能完全满足本文所想要达到的效果。由于设计类的图像并不同于摄影行业的照片，其注意力模型和审美性模型与本文所选取的方法有些偏差，因此在借鉴该方法对算法进行优化时，本文考虑了上文所提到的显著性检测，将显著性检测的标准和图像裁剪的标准进行了统一，使得平面设计图像中的核心元素能够在不包含无效背景的情况下被提取出来，方便进行下一步的多模态文档模型的处理。

在实际进行操作时，我发现显著性检测和图像裁剪部分的算法和模型有重复以及冲突之处，比如图像裁剪部分的注意力模型在实际含义上就与显著性检测的检测原理相类似，因此需要将图像裁剪技术的优先级提高，用其对输入图像进行第一步处理，将可能对显著性检测产生干扰的区域排除，之后再在此区域内进行显著性检测，方能提高显著对象的检出率以及正确率。若是先进行显著性检测，再进行图像裁剪时，会使得图像裁剪的模型作废，或是不正常工作，使得显著对象的裁剪区域缩小。譬如在图像较为复杂或对比度不高时，先进行显著性检测会将关注点放在局部正确或是错误的区域，导致裁剪的结果偏离真实值。

其次，显著性检测所输出的是一个坐标地图，通过检测显著对象的边缘像素 RGB 变化生成锚点，将其勾勒出来，若检测正确则不会包含有任何的背景像素。但是图像裁剪技术却是使用了一个“取景框”，包含了可能的显著对象和周围的背景。因此结合时需要对代码进行修改，在不影响检测效果的情况下实现无背景的精确裁剪。

受原显著性检测模型所输出的黑白图片启发，首先我定义了一个辅助函数 normPRED 将预测结果归一化到[0,1]的范围内，这是通过减去最小值并除以最小值和最大值之差实现的。同时加上一个小常数避免除零错误。其次创建了一个图像裁剪的新函数，这个函数先将预测的结果去除多余的维度并转化成 numpy 数组，然后打开原始图像把其转化为 RGBA 格式（含有透明度通道），并调整预测结果的尺寸以匹配原图的尺寸。之后创建二值化掩码，使用阈值（最终版代码将阈值设置在了 0.5）将预测结果转化为二值图像。最后创建一个全透明的空图像并使用掩码将原图粘贴到空图像上。具体代码如图 15 至 16 所示：

```python
def normPRED(d):
    ma = torch.max(d)
    mi = torch.min(d)
    dn = (d - mi) / (ma - mi + 1e-8)  # 加入小常数避免除零
    return dn
```

图 15 辅助函数

```python
def save_cropped_output(image_name, pred, d_dir, threshold=0.1):
    predict = pred.squeeze()
    predict_np = predict.cpu().data.numpy()

    img_name = image_name.split("/")[-1]
    image = Image.open(image_name).convert("RGBA")
    h, w = image.size

    # 调整图片尺寸以适应原图
    predict_np_resized = transform.resize(predict_np, (w, h), mode='constant', anti_aliasing=True)

    # 从显著性地图上构建二值化掩码
    binary_map = predict_np_resized > threshold
    binary_mask = Image.fromarray((binary_map * 255).astype(np.uint8))

    # 创建空图
    empty_image = Image.new("RGBA", (w, h), (255, 255, 255, 0))

    # 粘贴掩码
    result_image = Image.composite(image, empty_image, binary_mask)

    # 保存结果
    result_image.save(os.path.join(d_dir, img_name.split(".")[0] + '_cropped.png'))
```

图 16 新裁剪函数

在此之前有一版修改很久却没有达到预期效果的代码，这里也进行一下实验误差分析。首先这段代码使用了 skimage 库读取图像而非 PIL 库，其次是采取了提取显著对象边界框的方式进行裁剪，而非使用二值化掩码。由于裁剪出的图片一直包含大量的背景并且对显著对象的完整性产生了损害（详见第三章），因此加入了 padding 值，给裁剪的边缘留下一些容错值，但是效果并不明显，依然有一部分的显著对象被裁剪掉，若是将 padding 值设置的过高（如 100+）则会裁剪入过多的背景导致与真实值背离，因此该方法是不如上文所设计的新算法模型的。具体代码如图 17 所示：

```python
def save_cropped_output(image_name, pred, d_dir, threshold=0.1, padding=0):
    predict = pred.squeeze()
    predict_np = predict.cpu().data.numpy()

    img_name = image_name.split("/")[-1]
    image = io.imread(image_name)
    h, w = image.shape[:2]

    # 调整图像尺寸以适应原图
    if predict_np.shape != (h, w):
        predict_np = transform.resize(predict_np, (h, w), mode='constant')

    # 提取显著性对象的边界框
    binary_map = predict_np > threshold
    coords = np.argwhere(binary_map)
    if coords.size == 0:
        print(f"No salient object detected in {image_name}")
        return

    y0, x0 = coords.min(axis=0)
    y1, x1 = coords.max(axis=0)

    # 添加padding
    y0 = max(y0 - padding, 0)
    x0 = max(x0 - padding, 0)
    y1 = min(y1 + padding, h)
    x1 = min(x1 + padding, w)

    # 显示坐标
    print(f"Image: {img_name}, y0: {y0}, x0: {x0}, y1: {y1}, x1: {x1}, padding: {padding}")

    # 裁剪原图
    cropped_image = image[y0:y1, x0:x1]

    # 保存图片
    aaa = img_name.split(".")
    bbb = aaa[0:-1]
```

图 17 旧裁剪函数

原先的图像裁剪模型会将物体周围的一小部分留下，这影响了设计师对于核心元素提取的这一需求。因此需要定义一个新的裁剪函数，其将沿着显著性检测结果的边缘进行裁剪，尽可能地将显著对象提取出来以便后续操作。

新旧裁剪函数的对比效果如图 18 所示：

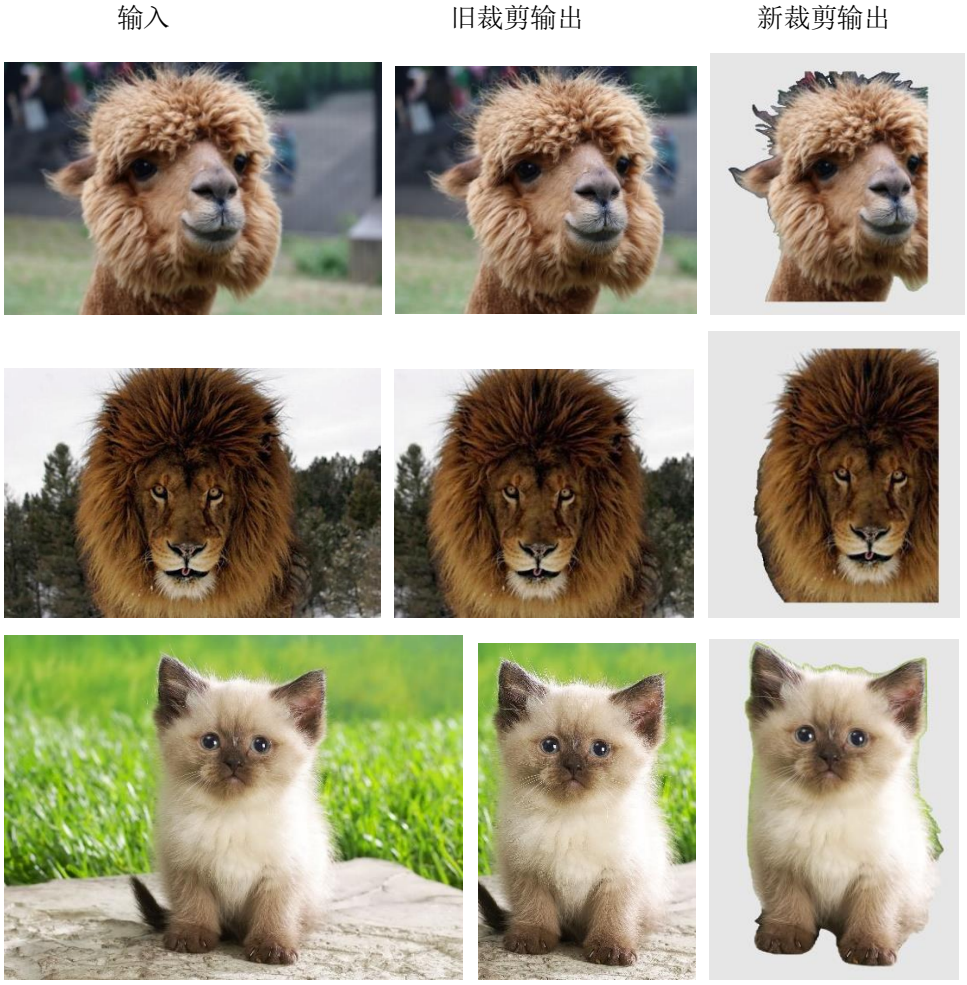


图 18 新旧裁剪函数的裁剪效果对比

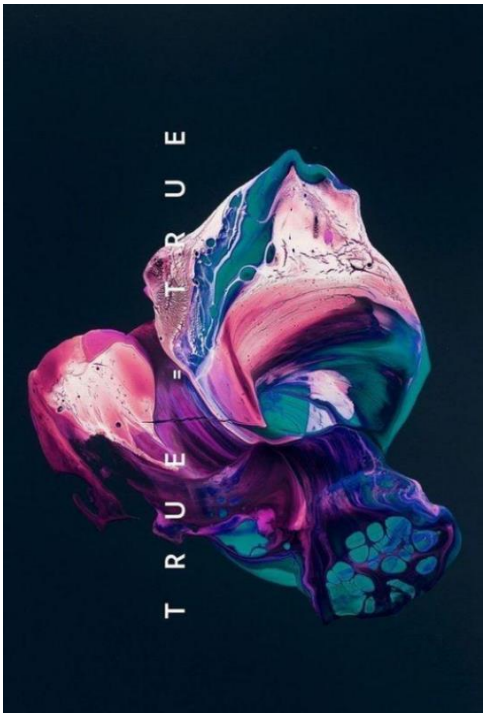
由图 18 可见，该改进算法在对一般图像进行裁剪时精确度较高，但是仍存在诸如将一小部分背景裁剪到结果图中，以及错将显著对象的一部分裁剪掉的情况，这种情况可能需要进一步对参数进行调整或对算法进行优化后才能避免。

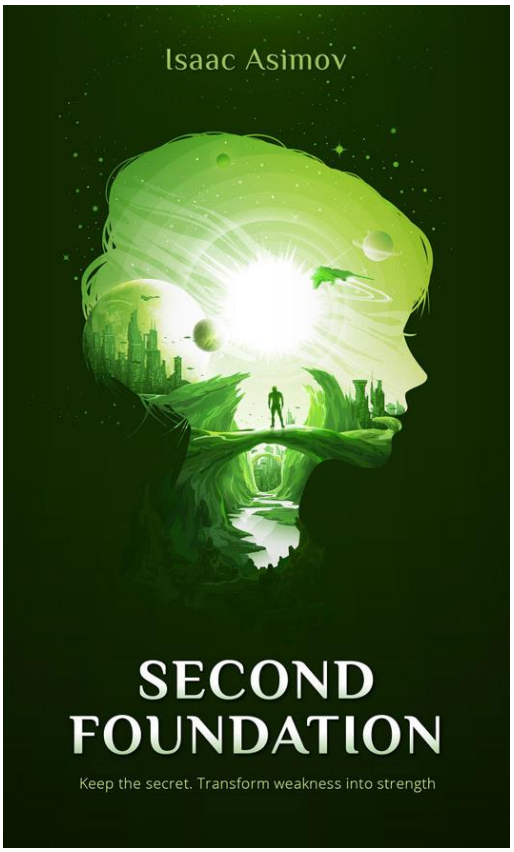
图 19 展示了该改进算法对平面设计图像的裁剪效果：


Alembic
There is no treatment
for an Indian's sweet tooth
Thankfully, there is a treatment for controlling
Post Prandial Hyperglycemia.
Vogo


strix s.a.
OLA
Oxipoder
Quitamanchas
Elimina las manchas
OLA
Oxipoder
Quitamanchas


OLA
Oxipoder
Quitamanchas


TRUE=TRUE


Isaac Asimov
SECOND
FOUNDATION
Keep the secret. Transform weakness into strength

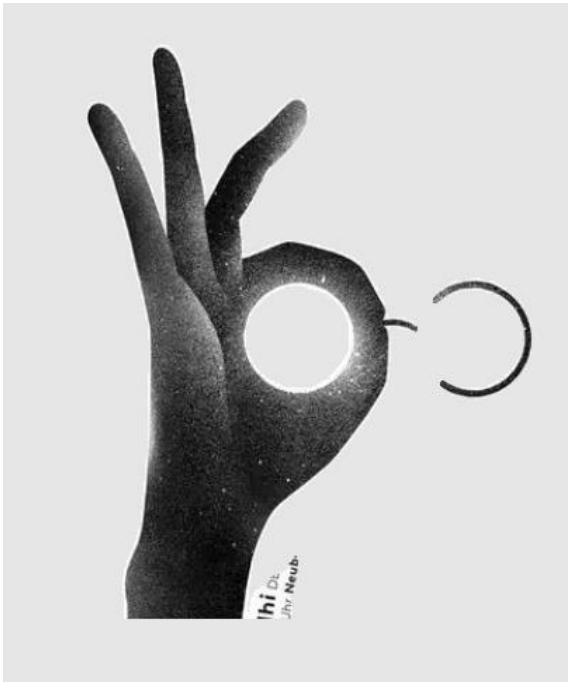


图 19 列表排序方法与显著性检测结合裁剪平面设计图例对比

由图 19 可知该改进算法在处理颜色对比度较高的图像时，可以做到基本上完美的图像裁剪，不过跟上文显著性检测时的问题一样，在第一张图中，由于人物的描边颜色较淡，在进行显著性检测的时候就将其忽略了，只留下了高颜色对比度的显著对象，但是这并不是想要的结果，对于设计师而言肯定是希望能将图像中包含人物在内的一整块区域都裁剪下来作为核心素材。同样地，在第二张图中也是对于次要显著对象产生了丢失，这部分算法还需要更深层次的研究，也许需要将排序队列中的第二或第三个元素也当作显著对象考虑。

并且该算法在处理深色的，且对比度较低的图像时，会对显著对象的检测产生缺漏，这是显著性检测这一大类算法的致命缺陷。在引入列表排序的图像裁剪方法时对图像的注意力核心进行了再确认，但是并没有很好的改善这一问题，从图上可见这两张图例由于错误的裁剪都缺失了一部分。

在后续的研究中，如何均衡显著性检测的模型和图像裁剪的注意力模型是一个主要问题。在引入主要次要显著对象之后将他们全都裁剪下来是该研究未来的首要目标。

第四章 设计方案空缺部分填充

在经过上文的两步图像处理之后，原版待处理的图像已经转化成了显著目标的裁剪图，此时为了满足设计师后续的设计需要，我们需要为这张裁剪图配上相应的文字或其他次要元素，以填充由于裁剪而空缺的部分。

Cyberagent 所研发的 Flex DM 模型便能很好地满足这个需求[11]。这个模型可以做到将元素对齐、合适的字体选择以及和谐的色彩使用等设计任务一并解决，从而达到一个辅助平面设计工作的全能工具。这个模型将矢量图形文档视作是多模态元素的集合，并学习使用统一架构预测被遮罩的字段，如元素类型、元素位置、元素样式属性以及图像或文本。通过使用多任务学习和领域内预训练，这个模型能够更好地找到不同文档字段之间的多模态关系。实验结果表明，这个模型对于不同的设计任务的处理性能与针对单一任务的模型性能相当。

4.1 技术背景

矢量图形文档是一种由文字、图像以及格式等多样化的多模态元素所构成的，是当今视觉交流的主要媒介。一个完整的矢量图形文档——即一个平面设计方案，需要设计师进行多种设计任务来完成，包括填充背景图像、更改字体的大小和颜色以及调整格式等。就算是非常熟练的设计师也需要耗费大量的人力在这些工作上，更不用说新人设计师或行业外人员了。为了能够使这一流程自动化，Cyberagent 提出了一种基于从完成的设计中学习设计知识的交互式框架，并且基于此开发了一个能够灵活切换设计任务的综合模型。

对于平面设计任务，其具有可操作性高（设计师可随意发挥增减设计元素）以及多模态元素交互复杂性高的特征。由于其可操作性高的特征，几乎图形文档的所有外观属性都可以进行编辑，目前已有诸如布局生成[12]、字体推荐[13]以及着色[14]等功能的实现，但是还没有一个研究开发出一个灵活的模型，其可以系统性地对多个设计任务进行处理，并自动决定平面设计任务流程。

在这个灵活的模型中，设计者将设计元素的某个特定属性（如颜色、字体等）统称为字段，并将各种设计任务统一成了被遮罩字段的预测。这个想法是受到了遮罩自编码器[15]和多任务模型[16]的启发。它的工作原理是利用遮罩模式在单个模型中切换不同的设计任务。比如元素填充这一任务可以被转化为预测新添加元素的所有属性，即字段。根据此原理，这个灵活文档模型呈现出一种编码器-解码器的结构，并通过多任务学习方式进行训练。其功能如图 20 所示：

图 20 Flex DM 可执行设计任务总览（图源[11]）

4.2 模型概述

FlexDM 模型是由三个主要模块构成的：编码器、转码模块以及解码器。

首先介绍一下矢量图形文档的概念。矢量图形在之前的研究中被当作实现高分辨率或具有艺术效果的高质量渲染的方案，但是与这些只包含了一维线条或路径的矢量图形相比，此模型的多模态特征使其要复杂得多。这种多模态的矢量图形先前只被用于文档级别的无条件生成，如 CanvasVAE 的功能[17]。但这个研究并未深入探究多模态矢量图形对于设计任务的完成能力。Doc2PPT[18]则可以根据提供的多模态矢量图形文档生成一个幻灯片，不过它只能忠于设计师提供的矢量图形，并不能对文档中缺失的内容进行推断并生成。

矢量图形的具体形式如图 21 所示：

```
{
    "tpye":text,"position":[0.1,0.6],
    "size":[0.8,0.2],"text":"CAR WASH",
    "color":navy,"font_family":"Oswald",
},
```

图 21 矢量图形代码示意图

这一代码对应的图形便如图 22 所示：

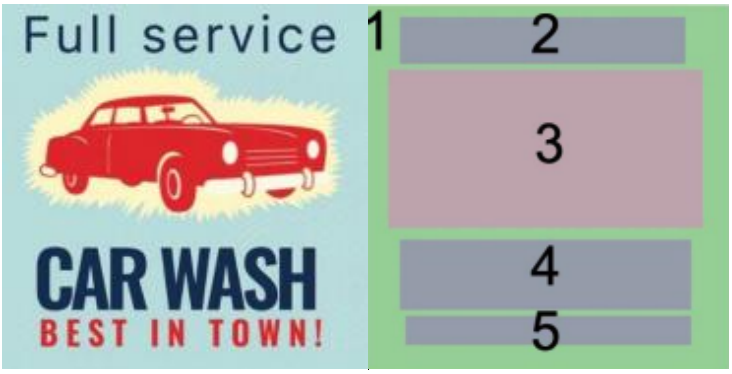


图 22 矢量图形代码所对应图形（图源[11]）

这是一种便于进行平面设计编辑的数据形式，它包含了所有人眼能从图上所观察到的视觉特质，这是一种可编辑并且便于理解的形式。

利用这种形式我们便能更好地理解人脑在进行平面设计创作时所经历的过程，该过程是一种将矢量图形拆解并同时进行多任务规划的方法。在这个流程中充满了大量具有可能性的行为，同时对各种多模态元素进行了复杂的交互

该过程如图 23 所示：

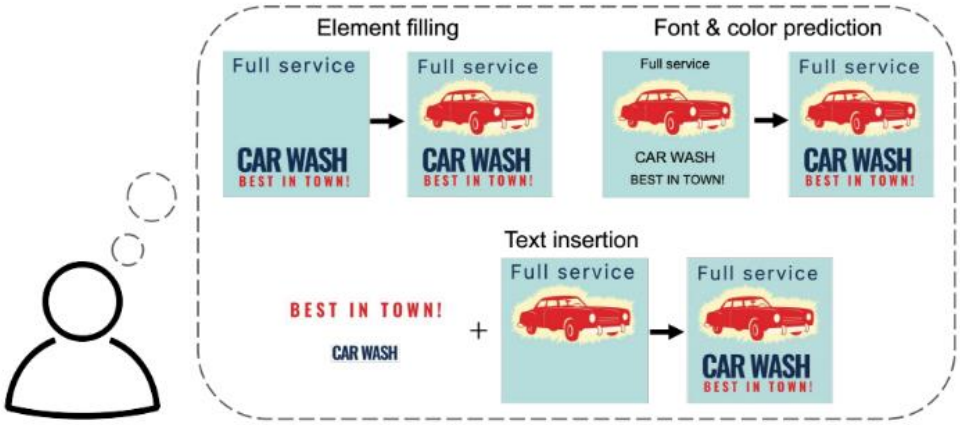


图 23 设计师进行平面设计时的元素调用（图源[11]）

因此想要模仿这种行为进行建模的话，需要将空白的、待创作的部分划分为遮盖字段（即[MASK]部分）。但是对于多种类型的区域进行编码或者解码，甚至对大面积的区域进行此操作是非常困难的。为了解决这个问题，此模型构建了一种编码转码解码系统，它可以多任务地进行这些译码工作，如图 24 所示：

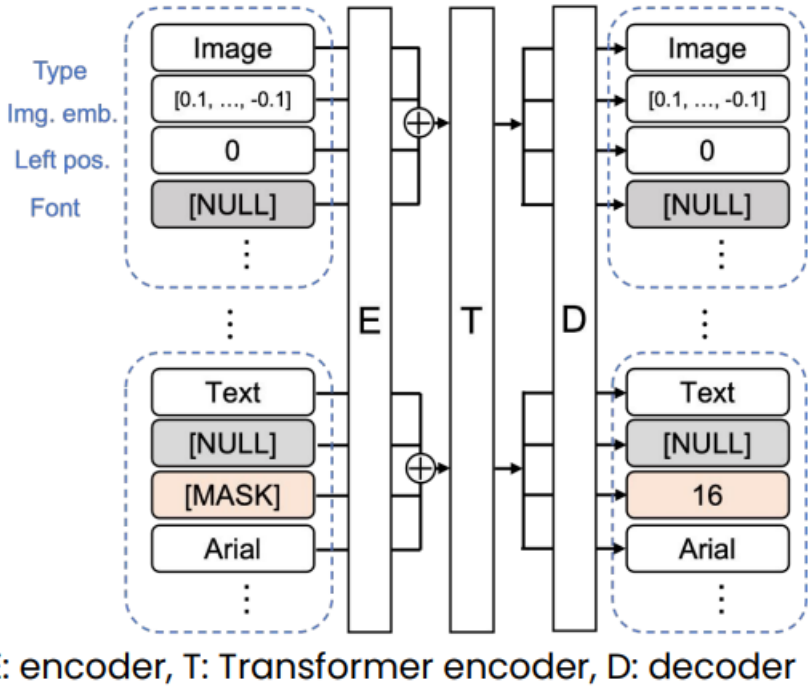


图 24 编码转码解码系统（图源[11]）

由于不同类型的字段具有不同的属性和特征，这个系统为每种字段类型设计了特定的编码器和解码器，以便更好地处理其特定属性。例如图像字段的编码器会处理图像嵌入向量，而文本字段的编码器则会处理文本内容和字体属性。为了处理大量字段的交互，该模型仅在元素级别上考虑字段间的交互。这意味着在转码的过程中，模型只处理单个元素的嵌入向量，而不处理字段内部的详细交互，从而简化计算复杂度。

图 25 展示了一些 FlexDM 模型生成的结果，其中第一、二行是字体颜色预测的结果，第三、四行是字体类型预测的结果，第五、六行是元素填充生成的结果。

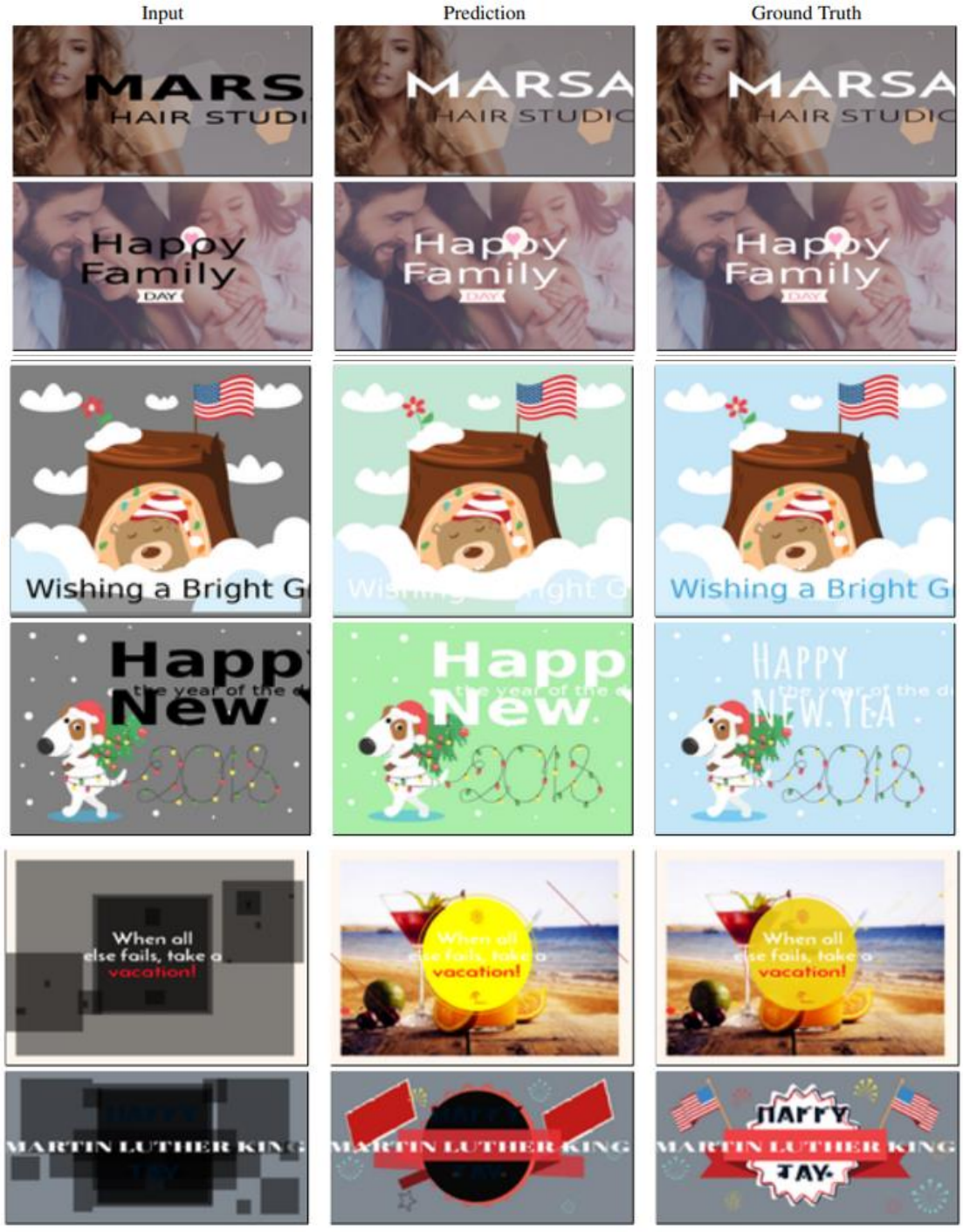


图 25 模型生成结果展示对比（图源[11]supplementary）

由该模型生成的预测图与真实值的对比可发现，生成的内容从各个模态的角度都非常贴近于真实值，分歧主要产生在部分字体以及背景颜色上，主体的页面布局是相同的，甚至某些预测结果比真实值更加符合观看者的审美。

将这个模型与之前的显著性检测以及图像裁剪统合之后，便能得到一个图像驱动的平面设计生成模型。

在经过显著性检测和图像检测的基础上，需要为代码添加一条管道，将其导入多模态文档模型中，并且将预训练的模型加载进来，如图 26 所示：

```python
import os
from PIL import Image
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model # type: ignore

# 定义一些参数
prediction_dir = './test_data/test_results/'
model_path = '../results/crello/ours-exp-ft/checkpoints'
input_size = (128, 128)

# 加载裁剪后的图像
def load_and_preprocess_image(image_path, target_size):
    img = Image.open(image_path).convert('RGB')
    img = img.resize(target_size, Image.ANTIALIAS)
    img_array = np.array(img) / 255.0  # 归一化
    return img_array

# 获取所有裁剪后的图像路径
cropped_images = [os.path.join(prediction_dir, f) for f in os.listdir(prediction_dir) if f.endswith('_cropped.png')]

# 加载模型
model = load_model(model_path)
```

图 26 模型串联代码（1）

成功导入素材图片后，便可唤起模型进行生成，代码如图 27 所示：

```python
# 进行预测并可视化结果
def visualize_prediction(model, image_paths, input_size):
    for image_path in image_paths:
        # 加载和预处理图像
        img = load_and_preprocess_image(image_path, input_size)
        img = np.expand_dims(img, axis=0)  # 扩展维度以匹配模型输入

        # 进行预测
        pred = model.predict(img)

        # 可视化预测结果
        print(f"Predictions for {os.path.basename(image_path)}:")
        print(pred)

# 对所有裁剪后的图像进行预测
visualize_prediction(model, cropped_images, input_size)
```

图 27 模型串联代码（2）

至此，图像驱动的平面设计生成这一流程便可总结概括为图 28 这张流程图：

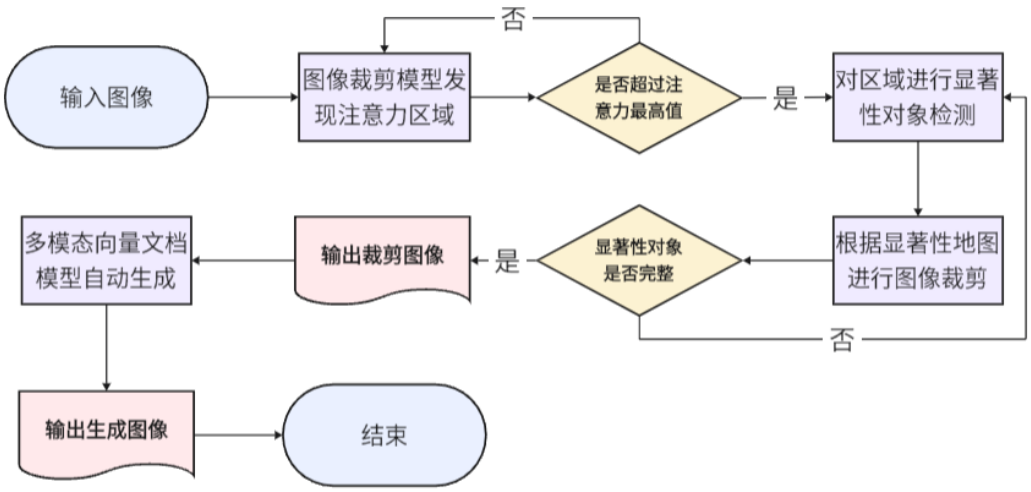


图 28 图像驱动的平面设计生成系统流程图

输入裁剪图像时，需要将 FlexDM 的模式调整为元素预测状态，并且在这一过程中，生成的编码器里 Type 字段已知为 Image，因为是图片所以 Font 字段为 NULL，而 img.emb 以及 Left pos.等字段都为遮盖字段（MASK），这些需要模型去预测它的值。当然也可以手动将图像的位置置于中间（更改 Left pos.的值），这取决于设计师的需求。处理这些 MASK 字段的开头部分代码如图 29 所示：

```
target_task = "elem"  # choose from: elem, pos, attr, txt, img
column_names = {
    "txt": ["gt-layout", "gt-visual", "input", "pred"],
    "img": ["gt-layout", "gt-visual", "input", "pred"],
    "attr": ["gt-layout", "gt-visual", "input", "pred"],
    "pos": ["gt-layout", "gt-visual", "pred-layout", "pred-visual"],
    "elem": ["gt-layout", "gt-visual", "input-layout", "input-visual", "pred-layout", "pred-visual"],
}

def visualize_reconstruction(
    models: List[tf.keras.Model],
    example: Dict,
    dataspec: DataSpec
):
    svgs = []
    items = dataspec.unbatch(example)
    svgs.append(list(map(builders["layout"], items)))
    svgs.append(list(map(builders["visual"], items)))
    if target_task == "txt":
        svgs.append(list(map(builders["visual_wo_text"], items)))
    elif target_task == "img":
        svgs.append(list(map(builders["visual_wo_image"], items)))
    elif target_task == "attr":
        svgs.append(list(map(builders["visual"], [set_visual_default(x) for x in items])))

    seq_mask = get_seq_mask(example["length"])
    mfp_masks = get_initial_masks(input_columns, seq_mask)

    for key in mfp_masks.keys():
        if not input_columns[key]["is_sequence"]:
            continue
        mask = mfp_masks[key].numpy()

        if target_task == "elem":
            target_indices = [0]  # hide first
            for i in range(len(target_indices)):
```

图 29 多模态文档模型代码部分展示

第五章 实验

根据本文在上文所罗列的三个算法模型，对其进行修改串联后便可得到一个完整的图像驱动的平面设计生成模型。该完整模型已在第四章展示，此处便不再赘述。

借由我的自选数据集所训练的模型，将测试图像输入该系统中所产生的成功样例输出如图 30 所示：

输入图像　　裁剪图像　　输出设计

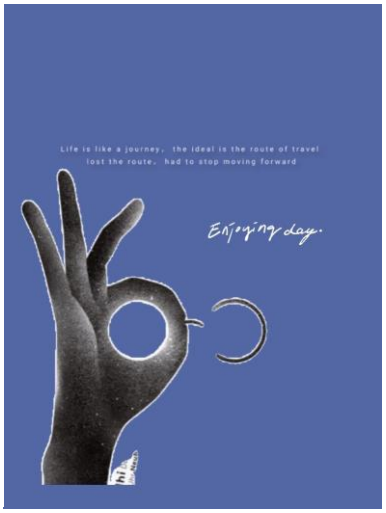


图 30 图像驱动的平面设计生成全流程生成结果展示

这里选取了一些生成效果比较好的图像，第一张图是直接从数据集里挑选的素材，可以发现该模型具有一定的稳定性，不过也有可能是训练数据较为单一的原因。

在此基础上，我又尝试了 FlexDM 模型的一些额外功能，加入了想要的主题文字，并让它为其选择排版位置和字体型号，成功样例如图 31 所示：

输入图像　　裁剪图像　　输出设计

图 31 加入主题的多模态设计生成结果（输入主题从上至下分别为 detergent，orange 以及 fashion）

图 31 中的主题文字分别为 detergent，orange 以及 fashion 均为手动输入，由于输入的图像较为抽象化，可以看到模型将生成的背景作为文字的字体参照物，形成了统一。

在尝试成功后我又试着用该模型对平面设计图像以外的图片进行生成，同时也应用文字模型，其中的成功案例如图 32 所示：

图 32 非平面设计图像生成图例

可见该图像驱动的平面设计生成模型在输入平面设计图像数据集进行训练后，对于输入的非平面设计图像的处理也有一定的鲁棒性。

由于训练用数据集是手动挑选的原因，成功的样例较少，希望在未来的研究中能够获得更大的数据集进行实验。

第六章 结论

本文在前三个章节中将图像驱动的平面设计生成这一项目进行了拆解分析，从图像的选择：显著性检测；到选择后的裁剪：图像裁剪技术；以及最后对裁剪出的显著对象进行创意元素扩充的灵活的多模态文档模型。

首先，在显著性检测章节中，本文探讨了如何通过计算机视觉技术识别图像中的显著区域。这些显著区域往往是图像中最引人注目的部分，通常也是设计师在构图时希望突出的部分。对此本文采用了先进的无监督博弈论，通过在大规模数据集上的训练，使模型能够精确识别并标记图像的显著区域。实验结果表明，基于显著性检测的图像预处理显著提高了后续裁剪步骤的效率和准确性。

其次，在图像裁剪章节中，本文详细介绍了如何利用显著性检测的结果进行自动裁剪。针对传统图像裁剪方法的不足，本文采用了一种列表排序算法，使裁剪过程能够更好地保留图像的核心内容，减少不必要的背景部分。通过实验验证，本文的方法在多种图像类型和应用场景下均表现出色，显著提高了图像裁剪的质量和效果。

最后，在多模态文档模型章节中，本文采用了 FlexDM 模型，该模型通过多任务学习和域内预训练，实现了对多种设计任务的统一处理。FlexDM 利用多模态遮盖字段预测技术，能够在处理复杂设计任务时表现出强大的灵活性和高效性。实验结果表明，FlexDM 模型在多任务处理上展现出较高的准确性和稳定性，并且生成结果十分接近真实值。

综上所述，本论文通过显著性检测、图像裁剪以及多模态文档模型三个方面的深入研究，为图像驱动的平面设计生成提供了一套高效、准确的解决方案。未来的研究可以进一步优化这些方法，提高其在更多复杂场景中的适应性和性能。

参考文献

[1] C. FANG, Z. LIN, R. MECH, and X. H. SHEN. Automatic Image Cropping using Visual Composition, Boundary Simplicity and Content Preservation Models [J]. Association for Computing Machinery, 2014,1105–1108.

[2] A. BORJI, M.M. CHENG, H. JIANG, and J. LI. Salient object detection: A benchmark [J]. IEEE Transactions on Image Processing, 24(12): 2015,5706– 5722.

[3] Y. ZENG, M. FENG, H. LU, G. YANG and A. BORJI, An Unsupervised Game-Theoretic Approach to Saliency Detection [J]. IEEE Transactions on Image Processing, 27(9): 2018, 4545-4554.

[4] B. JIANG, L. ZHANG, H. LU, C. YANG, and M.H. YANG. Saliency detection via absorbing markov chain [C]// Proceedings of the IEEE International Conference on Computer Vision, 2013, 1665–1672.

[5] C. YANG, L. ZHANG, H. LU, X. RUAN, and M.H. YANG. Saliency detection via graph-based manifold ranking [C]// Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2013, 3166–3173.

[6] J. LONG, E. SHELHAMER, and T. DARRELL, Fully convolutional networks for semantic segmentation [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, 3431-3440.

[7] Q. HOU , M.M. CHENG, X. HU, A. BORJI, Z. TU, and P. H. TORR. Deeply supervised salient object detection with short connections [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(4): 2019, 815-828.

[8] Y.L. CHEN, T.W. HUANG, KAI, H. CHANG, Y.C. TSAI, H.T. CHEN, and B.Y. CHEN. Quantitative analysis of automatic image cropping algorithms: A dataset and comparative study. [C]// Proceedings of IEEE Computer Vision and Pattern Recognition, 2017, 226–234.

[9] W. LU, X. XING, B. CAI and X. XU, Listwise View Ranking for Image Cropping [J]. IEEE Transactions on Image Processing, 25(7), 2019, 91904-91911.

[10] C. HONG, S. DU, K. XIAN, H. LU, Z. CAO and W. ZHONG, Composing Photos Like a Photographer [C]// Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2021, 7053-7062.

[11] N. INOUE, K. KIKUCHI, E. SIMO-SERRA, M. OTANI and K. YAMAGUCHI, Towards Flexible Multi-modal Document Models [C]// Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2023, 14287-14296.

[12] D.M. ARROYO, J. POSTELS, and F. TOMBARI, Variational transformer networks for layout generation [C]// Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2021,10324-10336.

[13] N.X. ZHAO, Y. CAO, and R.W.H. LAU. Modeling fonts in context: Font prediction on web designs [J]. Computer Graphics Forum, 37(7): 2018, 385-395.

[14] A. JAEGLE, F. GIMENO, A. BROCK, O. VINYALS, A. ZISSERMAN, and J. CARREIRA. Perceiver: General perception with iterative attention [C]// Proceedings of International Conference on Machine Learning, 2021, 1065-1078.

[15] J. DEVLIN, M.W. CHANG, K. LEE, and K. TOUTANOVA. BERT: Pre-training of deep bidirectional transformers for language understanding [C]// Proceedings of The North American Chapter of the Association for Computational Linguistics, 2019, 783-795.
[16] A. JAEGLE, S. BORGEAUD, J.B. ALAYRAC, C. DOERSCH, C. IONESCU, D. DING, S. KOPPULA, D. ZORAN, A. BROCK, E. SHELHAMER et al. Perceiver IO: A general architecture for structured inputs & outputs [C]// Proceedings of International Conference on Learning Representations, 2022, 1045-1069.
[17] K. YAMAGUCHI. CanvasVAE: Learning to generate vector graphics documents [C]// Proceedings of International Conference on Computer Vision, 2021, 825-837.
[18] T.J. FU, W.Y. WANG, D. MCDUFF, and Y. SONG. Doc2ppt: Automatic presentation slides generation from scientific documents [C]// Proceedings of AAAI Conference on Artificial Intelligence, 2022, 36-49.

致谢

首先，我想感谢指导我完成该课题的老师，曹迎老师。当时在选取论文题目时便与曹老师在关于设计类和计算机系的交叉应用之间能碰撞出什么火花进行了讨论。之后在论文的修改期间，曹老师提供了大量精确的修改意见，真的让我学到了很多在课堂上不会接触到的东西。虽然这个课题跟我预想中与设计方面的关联性并没有特别强，但是在研究的过程中还是收获颇多，对于这两个学科在未来能如何进一步融合并产生新的命题也有一定的展望。

其次，我想感谢我的父母给予我的耐心和希冀。在上科大学习的四年里，有不少苦恼和烦闷的时候，不过好在逐渐探索到了计算机的一些交叉学科的乐趣，也找到了未来希望能继续钻研的方向。

最后，我想感谢陪伴了我四年时光的上海科技大学。在这里虽有苦有乐，但大多数时光还是乐大于苦，苦中作乐的。上科大的严格教育和自由学风让每一个上科大学子都能向着自己所希望的方向发展。