

组件识别方法、装置、电子设备及存储介质

相关申请的交叉引用

本发明基于申请号为 202111422206.X、申请日为 2021 年 11 月 26 日的中国专利申请提出，并要求该中国专利申请的优先权，该中国专利申请的全部内容在此引入本发明作为
5 参考。

技术领域

本申请涉及计算机技术领域，涉及一种组件识别方法、装置、电子设备及存储介质。

背景技术

在前端业务开发过程中，为了通过组件代码生成视觉稿中对应的用户界面（UI，User
10 Interface)代码，需要人工识别并标注对应的文档对象模型（DOM，Document Object Model）
中的节点对应的 UI 组件类型，上述做法存在效率不高的问题。

发明内容

有鉴于此，本申请实施例提供一种组件识别方法、装置、电子设备及存储介质，以至少解决相关技术在组件识别时效率不高的问题。

15 本申请实施例的技术方案是这样实现的：

本申请实施例提供了一种组件识别方法，所述方法包括：

将第一图像输入设定识别模型，得到所述设定识别模型输出的所述第一图像中的至少一个 UI 区块；所述第一图像基于第一视觉稿确定；所述设定识别模型用于在输入的图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域；

20 在第一视觉稿对应的文档对象模型 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点；

将每个第一节点对应的图像输入设定分类模型，得到所述设定分类模型输出的第一节点对应的 UI 组件类型；所述设定分类模型用于确定输入的图像对应的 UI 组件类型。

上述方案中，所述在第一视觉稿对应的文档对象模型 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点，包括：
25

遍历所述第一视觉稿对应的文档对象模型 DOM 的节点；

在区域重叠率大于第一设定阈值的情况下，确定所述区域重叠率对应的节点为第一 UI 区块对应的第一节点；其中，

所述区域重叠率表征第二图像与所述第一 UI 区块的重叠率；所述第二图像根据节点的节点信息对所述第一视觉稿裁剪得到。

5 上述方案中，在所述将第一图像输入设定识别模型之前，所述方法还包括：

通过边缘检测，在第一视觉稿中确定出至少两个第一矩形区域；

基于至少一个第一矩形区域裁剪所述第一视觉稿，得到对应的第一图像。

上述方案中，在所述基于至少一个第一矩形区域裁剪所述第一视觉稿，包括：

合并设定方向相邻的第一矩形区域，得到第二矩形区域；其中，所述第二矩形区域

10 在设定方向的长度小于第二设定阈值；

基于所述第二矩形区域裁剪所述第一视觉稿。

上述方案中，所述将每个第一节点对应的图像输入设定分类模型，包括：

将根据第一节点的节点信息裁剪所述第一视觉稿得到的第二图像，输入所述设定分类模型；

15 和/或，

将根据第一节点对应的 UI 区块裁剪所述第一视觉稿得到的第三图像，输入所述设定分类模型。

上述方案中，在所述将第一图像输入设定识别模型之前，所述方法还包括：

基于对应有至少一个第一标注的第二视觉稿，训练得到所述设定识别模型和所述设定

20 分类模型；其中，

所述第一标注包括能够以 UI 组件渲染得到的图像区域的第一位置信息和对应的第一标签；第一位置信息用于描述所述图像区域在第二视觉稿的位置；所述第一标签用于描述所述 UI 组件的类型。

上述方案中，所述基于对应有至少一个第一标注的第二视觉稿，训练得到所述设定识

25 别模型和所述设定分类模型，包括：

基于每个第一标注裁剪所述第二视觉稿，得到对应的第四图像；所述第四图像对应有第一标签；

将所述第二视觉稿的第一标签替换为第二标签，得到第五图像；所述第二标签表征对应的图像区域为 UI 区块；

30 基于第四图像训练得到所述设定分类模型；

基于第五图像训练得到所述设定识别模型。

上述方案中，在所述基于第四图像训练所述设定分类模型之前，所述方法还包括：

在至少一种第一标签对应的第四图像的数量小于第三设定阈值的情况下，生成包含对应 UI 组件类型的页面图像，并基于生成的页面图像得到对应的第四图像。

5 上述方案中，所述生成包含对应 UI 组件类型的页面图像，包括：

在设定界面渲染对应的 UI 组件类型的组件代码，得到对应 UI 组件类型的页面图像；
和/或，

通过视觉稿生成代码工具得到第三视觉稿，得到对应 UI 组件类型的页面图像。

本申请实施例还提供了一种组件识别装置，包括：

10 第一处理模块，被配置为将第一图像输入设定识别模型，得到所述设定识别模型输出的所述第一图像中的至少一个 UI 区块；所述第一图像基于第一视觉稿确定；所述设定识别模型被配置为在输入的图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域；

第二处理模块，被配置为在第一视觉稿对应的 DOM 的节点中，确定与所述至少一个
15 UI 区块中的每个 UI 区块对应的一个第一节点；

第三处理模块，被配置为将每个第一节点对应的图像输入设定分类模型，得到所述设定分类模型输出的第一节点对应的 UI 组件类型；所述设定分类模型被配置为确定输入的图像对应的 UI 组件类型。

本申请实施例还提供了一种电子设备，包括：处理器和用于存储能够在处理器上运行
20 的计算机程序的存储器，

其中，所述处理器用于运行所述计算机程序时，执行上述组件识别方法的步骤。

本申请实施例还提供了一种存储介质，其上存储有计算机程序，所述计算机程序被处理器执行时实现上述组件识别方法的步骤。

本申请实施例提供的一种组件识别方法、装置、电子设备及存储介质，将第一图像输入
25 设定识别模型，得到设定识别模型输出的第一图像中的至少一个 UI 区块；第一图像基于第一视觉稿确定；设定识别模型用于在输入的图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域；在第一视觉稿对应的文档对象模型 DOM 的节点中，确定与至少一个 UI 区块中的每个 UI 区块对应的一个第一节点；将每个第一节点对应的图像输入设定分类模型，得到设定分类模型输出的第一节点对应的 UI 组件类型；设定分类模型用于确定输入的图像对应的 UI 组件类型。上述方案中，通过设定
30

识别模型确定第一视觉稿中能够以 UI 组件渲染得到的图像区域，确定这些图像区域在第一视觉稿对应的文档对象模型 DOM 的第一节点，通过设定分类模型得到第一节点对应的 UI 组件类型，这样，无须人工标注视觉稿对应的 DOM 中的节点的 UI 组件类型，提高了组件识别的效率。

5 附图说明

图 1 为本申请实施例提供的组件识别方法实现流程示意图；

图 2 为本申请实施例提供的图像中识别 UI 区块的示意图；

图 3 为本申请实施例提供的确定节点的实现流程示意图；

图 4 为本申请实施例提供的 UI 区块识别结果的示意图；

10 图 5 为本申请实施例提供的页面分割的实现流程示意图；

图 6 为本申请应用实施例提供的组件识别的实现流程示意图；

图 7 为本申请应用实施例提供的组件识别的页面示意图；

图 8 为本申请应用实施例提供的模型训练的实现流程图；

图 9 为本申请实施例提供的组件识别装置的结构示意图；

15 图 10 为本申请实施例提供的电子设备的结构示意图。

具体实施方式

随着低代码平台、无代码平台的出现，少量编码甚至无须编码，即可实现从视觉稿到应用程序前端代码的转换。视觉稿生成 UI 代码的过程，包括输入结构化的视觉稿数据，经过图层信息解析、布局、组件替换、语义化、生成代码等步骤，得到生成 UI 代码。其中，组件替换是视觉稿生成 UI 代码中的关键一环，通过将常用的视觉模块开发为组件，并在视觉稿生成 UI 代码过程中确定视觉稿中与设定视觉模块一致的部分，使用对应组件的代码进行替换。其中，组件替换的前提是明确视觉稿中能替换为组件的部分。

20

目前，在前端业务开发过程中，为了通过组件代码生成视觉稿中对应的 UI 代码，需要人工识别并标注对应的 DOM 中的节点对应的 UI 组件类型，上述做法存在效率不高的问题。

25

基于此，在本申请的各种实施例中，将第一图像输入设定识别模型，得到设定识别模型输出的第一图像中的至少一个 UI 区块；第一图像基于第一视觉稿确定；设定识别模型用于在输入的图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染

得到的图像区域；在第一视觉稿对应的 DOM 的节点中，确定与至少一个 UI 区块中的每个 UI 区块对应的一个第一节点；将每个第一节点对应的图像输入设定分类模型，得到设定分类模型输出的第一节点对应的 UI 组件类型；设定分类模型用于确定输入的图像对应的 UI 组件类型。上述方案中，通过设定识别模型确定第一视觉稿中能够以 UI 组件渲染得到的图像区域，确定这些图像区域在第一视觉稿对应的 DOM 的第一节点，通过设定分类模型得到第一节点对应的 UI 组件类型，这样，无须人工标注视觉稿对应的 DOM 中的节点的 UI 组件类型，提高了组件识别的效率。

为了使本申请的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本申请进行进一步详细说明。应当理解，此处描述的具体实施例仅仅用以解释本申请，并不用于限定本申请。

图 1 为本申请实施例提供的组件识别方法的实现流程示意图，本申请实施例提供了一种组件识别方法，应用于电子设备，其中，电子设备包括但不限于服务器、终端等电子设备。如图 1 示出的，组件识别方法，包括：

步骤 101：将第一图像输入设定识别模型，得到所述设定识别模型输出的所述第一图像中的至少一个 UI 区块。

其中，所述第一图像基于第一视觉稿确定；所述设定识别模型用于在输入的图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域。

组件识别装置将第一图像输入设定识别模型，设定识别模型对输入的第一图像中的至少一个 UI 区块进行识别，每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域。其中，设定识别模型通过标注有 UI 区块的视觉稿训练得到。第一图像基于第一视觉稿确定，可以是部分或全部的第一视觉稿的图像。每个 UI 区块为图像区域，可以认为是图像。

步骤 102：在第一视觉稿对应的 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点。

组件识别装置基于设定识别模型确定出的至少一个 UI 区块中的每个 UI 区块，在第一视觉稿对应的节点中确定出与每个 UI 区块对应的一个第一节点。其中，在节点与某个 UI 区块满足设定条件的情况下，将这个节点确定为这个 UI 区块的第一节点。

其中，DOM 的节点为视觉稿解析得到的结构化数据（即 Schema 数据）的节点，Schema 数据为视觉稿中所有元素构成的树状结构，可以存储为（JavaScript Object Notation）JSON 格式，每个节点包括对应图像的宽度、高度、位置等节点信息。

其中，第一视觉稿的图像可以通过视觉稿对应的 DOM 中的根节点的节点信息得到。

根节点的节点信息包括完整页面预览图的统一资源定位器(Uniform Resource Locator, URL)地址。

步骤 103: 将每个第一节点对应的图像输入设定分类模型, 得到所述设定分类模型输出的第一节点对应的 UI 组件类型。

5 其中, 所述设定分类模型用于确定输入的图像对应的 UI 组件类型。

组件识别装置将每个第一节点对应的图像输入设定分类模型, 设定分类模型对输入的图像的 UI 组件类型进行分类, 确定第一节点对应的图像所对应的 UI 组件类型, 从而得到第一节点对应的 UI 组件类型。其中, 设定分类模型通过标注有 UI 组件类型的图像训练得到。

10 其中, 可以将确定出的分类结果挂载在 JSON 文件中。

本申请实施例提供的方案中, 确定第一视觉稿中能够以 UI 组件渲染得到的图像区域对应的第一节点, 通过设定分类模型得到第一节点对应的 UI 组件类型, 这样, 无须人工标注视觉稿对应的 DOM 中的节点的 UI 组件类型, 提高了组件识别的效率。

同时, 在组件识别装置确定出第一节点对应的 UI 组件类型之后, 在将视觉稿转换为 UI 代码时, 基于对应类型的 UI 组件的代码生成 UI 代码, 简化了视觉稿到 UI 代码的编程工作, 使用对应组件的代码能够实现自动化的组件替换, 降低了 UI 代码开发的使用门槛。

并且, 组件识别装置通过设定识别模型确定第一视觉稿中能够以 UI 组件渲染得到的图像区域, 确定对应的第一节点, 将这些第一节点的图像输入设定分类模型, 这样, 设定分类模型无须处理第一视觉稿所有节点的图像, 降低了模型的计算量, 节约了组件识别所需的计算资源。

此外, 在本申请实施例中, 设定识别模型在识别 UI 区块时, 不要求识别出的 UI 区块的区域与通过 UI 组件渲染得到的图像区域完全相同, 这样, 设定分类模型识别的精准度要求放宽, 提升了设定识别模型的强健性。如图 2 示出的, 第一图像中包括多个 UI 区块 (21-23), UI 区块 23 与 UI 组件渲染得到的图像区域 231 不完全相同, 但不影响设定分类模型确定节点对应的 UI 组件类型。

在一实施例中, 所述将每个第一节点对应的图像输入设定分类模型, 包括:

将根据第一节点的节点信息裁剪所述第一视觉稿得到的第二图像, 输入所述设定分类模型;

和/或,

30 将根据第一节点对应的 UI 区块裁剪所述第一视觉稿得到的第三图像, 输入所述设定

分类模型。

其中，组件识别装置输入设定分类模型的每个第一节点对应的图像，可以是根据第一节点的节点信息裁剪第一视觉稿得到的第二图像，可以是基于第一节点对应的 UI 区块裁剪第一视觉稿得到的第三图像，也可以是第一节点对应的第二图像和第三节点。其中，由于识别出的 UI 区块的边缘不一定是精准的，使用节点信息裁剪得到的第二图像进行 UI 组件类型分类，能够提升分类结果的精确度。

其中，在节点与某个 UI 区块满足设定条件的情况下，将这个节点确定为这个 UI 区块的第一节点。在一实施例中，设定条件为节点对应的图像与 UI 区块的区域重叠率大于设定阈值。

10 所述在第一视觉稿对应的文档对象模型 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点，包括：

遍历所述第一视觉稿对应的文档对象模型 DOM 的节点；

在区域重叠率大于第一设定阈值的情况下，确定对应的节点为第一 UI 区块对应的第一节点；其中，

15 所述区域重叠率表征第二图像与所述第一 UI 区块的重叠率；所述第二图像根据节点的节点信息对所述第一视觉稿裁剪得到。

组件识别装置对于每个 UI 区块，在第一视觉稿对应的 DOM 的节点中确定对应的一個第一节点时，遍历第一视觉稿对应的 DOM 中的节点，计算节点对应的第二图像与这个 UI 区块的区域重叠率，在区域重叠率大于第一设定阈值的情况下，确定对应的节点为第一 UI 区块对应的第一节点。在确定出第一 UI 区块对应的第一节点之后，可以停止对第一视觉稿剩余的 DOM 中节点的遍历，重新开始确定下一个第一 UI 区块对应的第一节点。其中，区域覆盖率表征这个 UI 区块和节点对应的第二图像的图像重合情况，区域覆盖率越高说明两个图像的重合比例越高。

其中，组件识别装置对第一视觉稿对应的 DOM 中的节点的遍历，可以是前序遍历、中序遍历或后序遍历。结合图 3 示出的确定节点的实现流程示意图，以前序遍历，取第一设定阈值为 0.7 为例进行说明。对于每个第一 UI 区块，对视觉稿的结构化数据（即 Schema 数据）进行前序遍历。

首先，组件识别装置取根节点为当前节点，计算当前节点对应的第二图像与第一 UI 区块的区域覆盖率。

30 若区域覆盖率不大于 0，表示两者并不相交，则这个节点及节点的子节点不可能对应

到这个第一 UI 区块,判断这个节点是否有未遍历的兄弟节点,若存在未遍历的兄弟节点,则取兄弟节点为当前节点,计算区域覆盖率。

若区域覆盖率大于 0 而小于等于 0.7,判断这个节点是否有未遍历的子节点。若存在未遍历的子节点,则取第一个未遍历子节点为当前节点,计算区域覆盖率;若不存在未遍历的子节点,判断这个节点是否有未遍历的兄弟节点,若存在未遍历的兄弟节点,则取兄弟节点为当前节点,计算区域覆盖率。

若区域覆盖率大于 0.7,则认为该节点与第一 UI 区块对应,确定并标记这个节点为第一 UI 区块对应的第一节点。在节点的 JSON 信息中增加{'smart_ui': 'ui'}字段进行标记。

如前所述,设定识别模型在识别 UI 区块时,不要求识别出的 UI 区块的区域与通过 UI 组件渲染得到的图像区域完全相同,也就是说,如图 4 示出的 UI 区块识别结果的示意图,识别出的 UI 区块的边缘并非是完全精准的。在本实施例中,通过确定 UI 区块对应的第一节点,可以进一步获得第一节点对应的图像,提升了设定分类模型分类结果的精确度。

在一实施例中,在所述将第一图像输入设定识别模型之前,所述方法还包括:

通过边缘检测,在第一视觉稿中确定出至少两个第一矩形区域;

基于至少一个第一矩形区域裁剪所述第一视觉稿,得到对应的第一图像。

组件识别装置通过边缘检测等图像算法确定图像边缘特征,基于边缘检测识别第一视觉稿的至少两个第一矩形区域,对至少两个第一矩形区域进行划分,每次基于至少一个第一矩形区域裁剪第一视觉稿,得到一个对应的第一图像。

组件识别装置基于边缘检测识别图像中的成块元素(即第一矩形区域)。其中,可以使用视觉稿生成代码开源算法 UI2CODE 的元素检测模块,不会破坏节点具有完整轮廓的图像。

这样,在不破坏节点具有完整轮廓的图像的情况下,将第一视觉稿分割成至少两个图像,分割后的图像的长宽比在设定识别模型的最佳识别效果范围内,从而提高了识别长视觉稿的 UI 区块的精确度,提升了设定识别模型对图像的识别效果。

在一实施例中,在所述基于至少一个第一矩形区域裁剪所述第一视觉稿时,所述方法包括:

合并设定方向相邻的第一矩形区域,得到第二矩形区域;其中,所述第二矩形区域在设定方向的长度小于第二设定阈值;

基于所述第二矩形区域裁剪所述第一视觉稿。

其中,设定方向可以为竖直方向,也就是 y 轴方向。

组件识别装置将设定方向的长度之和小于第二设定阈值的至少两个第一矩形区域进行合并，得到第二矩形区域，并基于第二矩形区域裁剪第一视觉稿。

其中，组件识别装置可以在设定方向上依次判断相邻的两个第一矩形区域的长度之和是否小于第二设定阈值，在长度之和小于第二设定阈值的情况下，将这两个第一矩形区域合并，并继续判断合并得到的矩形区域和在设定方向上的下一个相邻的第一矩形区域的长度之和，直到两个矩形区域在设定方向的长度之和大于或等于第二设定阈值，基于合并得到的矩形区域裁剪第一视觉稿。

结合图 5 示出的页面分割的流程示意图进行说明，视觉稿的分割包括以下步骤：

a) 基于边缘检测识别图像中的成块元素（即第一矩形区域）。其中，可以使用视觉稿生成代码开源算法 UI2CODE 的元素检测模块，将元素检测模块的参数阈值设置为合适大小，检测后得到多个第一矩形区域。由于元素检测模块基于图像边缘信息进行分割，不会破坏节点具有完整轮廓的图像。

b) 对识别出的第一矩形区域根据 y 轴坐标进行排序，即根据距离视觉稿顶部的距离进行排序。

其中，为避免未检测出的区域信息丢失，可以在第一矩形区域之间插入矩形区域进行填充。例如，图 5 中的元素 61、62 未被识别检测出，通过插入矩形区域 63、64，以保证对应的信息不丢失。

c) 对第一矩形区域进行成组。自上向下合并第一矩形区域，当合并后高度 <400 则继续往下合并，若合并后高度 >600 ，则当前合并的第二矩形区域为一个新组。再取下一个第一矩形区域，重新开始成组。其中的高度，也就是设定的竖直方向上的长度。

d) 基于第二矩形区域裁剪第一视觉稿，得到合适高度的第一图像。

由于设定识别模型通过标注有 UI 区块的视觉稿训练得到，考虑到训练成本等原因，训练样本的数量不可能是无限的。对于一些图像，例如长宽比超过训练样本的长宽比的图像，通过设定识别模型的识别效果不佳。在本实施例中，在不破坏能以 UI 组件渲染得到图像区域的情况下，将第一视觉稿分割成至少两个图像，且分割后的图像的长宽比在设定识别模型的最佳识别效果范围内，从而提高了识别长视觉稿的 UI 区块的精确度，提升了设定识别模型对图像的识别效果。

每次在处理一个新的任务场景时，都需要针对任务场景进行模型训练。具体到组件识别场景，在使用设定识别模型和设定分类模型之前，需要对模型进行训练。在一实施例中，在所述将第一图像输入设定识别模型之前，所述方法还包括：

基于对应有至少一个第一标注的第二视觉稿，训练得到所述设定识别模型和所述设定分类模型；其中，

所述第一标注包括能够以 UI 组件渲染得到的图像区域的第一位置信息和对应的第一标签；第一位置信息用于描述所述图像区域在第二视觉稿的位置；所述第一标签用于描述
5 所述 UI 组件的类型。

组件识别装置基于第二视觉稿对应的每个第一标注的第一位置信息，确定一个能够以 UI 组件渲染得到的图像区域在第二视觉稿的位置。基于第二视觉稿对应的每个第一标注的第一标签，确定这个图像区域基于何种类型的 UI 组件渲染得到。基于对应有至少一个第一标注的第二视觉稿的图像，确定对应的识别样本数据集和分类样本数据集，基于对应的
10 数据集训练对应的模型。其中，第二视觉稿为视觉稿样本，可以是真实的视觉稿，也可以是根据需要生成的视觉稿。

在一实施例中，所述基于对应有至少一个第一标注的第二视觉稿，训练得到所述设定识别模型和所述设定分类模型，包括：

基于每个第一标注裁剪所述第二视觉稿，得到对应的第四图像；所述第四图像对应有
15 第一标签；

将所述第二视觉稿的第一标签替换为第二标签，得到第五图像；所述第二标签表征对应的图像区域为 UI 区块；

基于第四图像训练得到所述设定分类模型；

基于第五图像训练得到所述设定识别模型。

20 组件识别装置基于每个第一标注的第一位置信息，裁剪第二视觉稿得到对应的第四图像，第四图像对应有这个第一标注的第一标签，基于第一标签能够确定对应的第四图像的 UI 组件的类型，基于第四图像训练设定分类模型。

组件识别装置将第二视觉稿的第一标签替换为第二标签得到第五图像，基于第五图像的第二标签能够确定对应的图像区域为 UI 区块，基于第五图像训练设定识别模型。

25 在本实施例中，组件识别装置通过复用第二视觉稿的图像，能够得到两种训练样本，分别用于训练对应类型的模型，这样，降低了获取模型训练样本的成本。

在一实施例中，在所述基于第四图像训练所述设定分类模型之前，所述方法还包括：

在至少一种第一标签对应的第四图像的数量小于第三设定阈值的情况下，组件识别装置生成包含对应 UI 组件类型的页面图像，并基于生成的页面图像得到对应的第四图像。

30 在存在一个 UI 组件类型的第四图像的数量少于第三设定阈值的情况下，组件识别装

置生成对应 UI 组件类型的页面图像,再通过 Puppeteer 等工具对页面中的组件节点属性进行抖动,包括文本更改、元素位置偏移等,对处理后的页面进行截图,得到第四图像。

在一实施例中,所述生成包含对应 UI 组件类型的页面图像,包括:

在设定界面渲染对应的 UI 组件类型的组件代码,得到对应 UI 组件类型的页面图像;

5 和/或,

通过视觉稿生成代码工具得到第三视觉稿,得到对应 UI 组件类型的页面图像。

其中,组件识别装置可以采用以下至少一种方式生成对应 UI 组件类型的网页图像:

在页面中渲染对应的 UI 组件类型的组件代码,得到对应的网页图像;通过视觉稿生成代码工具,得到还原视觉稿的网页图像。

10 其中,组件识别装置可以通过调用代码库中的组件代码生成网页图像。

组件识别装置通过生成模型的训练样本,克服了数据集中不同组件类型的样本分布不均衡的问题,这样,基于数据集训练分类模型,提高了分类模型分类图像的精确度。

下面结合应用实施例对本申请再作进一步详细的描述。

图 6 示出了本申请应用实施例提供的组件识别的实现流程示意图。

15 组件识别的实现流程包括以下步骤:

步骤 701: 输入结构化的视觉稿数据 Schema。

其中,常见的视觉稿为 sketch 或者 psd 格式,本应用实施例的输入为视觉稿解析得到的结构化数据描述,即 schema 数据。Schema 数据为视觉稿中所有元素构成的树状结构,存储为 JSON 格式,其中每个节点包括宽度、高度、位置等节点信息。

20 步骤 702: 取根节点的图像作为视觉稿图像。

Schema 数据的根节点中包含了完整页面预览图的 URL 地址,下载完整页面预览图作为视觉稿原图进行后续处理。

步骤 703: 页面分割。

组件识别装置通过边缘检测等页面分割算法,将长视觉稿分割为合适高度的多个图像。

25 对于长宽比较大的视觉稿,无法使用通用目标检测模型得到较好的识别效果,所以需要将长视觉稿切分为合适高度的图像。页面分割的流程如图 5 所示。

其中,页面分割将长视觉稿分割为合适高度的图像,并且采用边缘检测等图像算法,基于图像边缘特征进行分割,不会破坏能以 UI 组件渲染得到的图像区域,解决了长视觉稿的识别问题。

30 目标检测是计算机视觉领域的基础任务之一,包含物体定位与分类两个子任务。输入

一张图像，可以找到图像中目标物体的类别与位置。

步骤 704: UI 区块识别。

使用 UI 区块识别模型（即设定识别模型），识别图像中的至少一个 UI 区块。

UI 区块识别通过 UI 区块识别模型识别图像中可能为组件的区域（即 UI 区块）。其中，
5 UI 区块识别模型是一种目标检测网络模型，通过带有 UI 区块标签的目标检测数据集，基于深度学习目标检测模型的 Mask-RCNN 预训练模型进行训练。

其中，数据集的制作使用 Labelme 工具，对收集到的视觉稿进行标注，标注视觉稿中各个组件的位置以及分类。得到的标注结果为 JSON 格式，其中记录了组件的类型与坐标。对标注结果进行备份。通过脚本将类型统一替换为“ui”。再导出为 cocos 格式的目标检测
10 数据集。其中，Labelme 是一种数据标注工具，可用来标注分类、检测、分割等常见的视觉任务，支持 VOC 格式和 COCO 等的格式导出。

组件识别装置基于目标检测数据集进行模型训练与优化，将训练得到的 UI 区块识别模型用于 UI 区块识别。对于步骤 703 中分割的图像，通过 UI 区块识别模型进行识别，得到 UI 区块的坐标与类别“ui”，如图 7 所示出的组件识别的页面示意图。其中，UI 区块识
15 别模型还会输出识别出的每个 UI 区块对应的置信度，例如“ui 0.95”即该图像区域为 UI 区块的置信度为 0.95。

步骤 705: 节点映射。

组件识别装置通过计算图像的 UI 区块与视觉稿对应的 DOM 的节点的区域覆盖率，将识别出的区块映射到视觉稿中的节点，标记节点为 UI 区块。根据区域覆盖率，确定视觉
20 稿中与 UI 区块对应的更小节点。其中，视觉稿数据为步骤 701 输入的视觉稿。

如图 4 示出的，UI 区块识别模型识别到的区块的边缘并非精准的，需要将区块映射到视觉稿中的节点。

节点映射的实现流程参见图 3，在此不再赘述。在区域覆盖率大于第一设定阈值的情况下，则认为该节点与第一 UI 区块对应，确定并标记这个节点为第一 UI 区块对应的第一
25 节点。在节点的 JSON 信息中增加{‘smart_ui’: ‘ui’}字段进行标记。

步骤 706: 组件分类。

组件识别装置根据 UI 区块对应的节点的坐标、宽高信息，从视觉稿原图中裁剪 UI 区块图像，送入 UI 类型分类模型（即设定分类模型）进行分类，得到组件的类型。

其中，图像分类是计算机视觉领域的基础任务之一，输入一张图像，输出图像的分类
30 结果。

- 步骤 707: 输出带组件标记的视觉稿结构数据。
- 组件标记为视觉稿对应的 Dom 节点的组件类型标记。
- 在另一应用实施例中, 还可以对视觉稿中每个节点进行 UI 组件分类。
- 组件识别装置通过以上步骤得到了可能是区块的节点, 还需要确定节点对应的 UI 组件类型。通过训练 MobileNet 分类模型来分类。分类模型数据集的样本来源, 包括真实的视觉稿和/或生成的视觉稿。其中, 真实的视觉稿使用已标注好的目标检测数据集, 根据标注信息中的坐标对视觉稿页面进行裁剪, 得到分类数据集, 分类数据集中的每个样本带有对应的 UI 组件类型标签。对于数据量过少的 UI 组件类型, 通过生成样本来扩充数据集。
- 组件识别装置训练得到 UI 类型分类模型后, 根据标记的 UI 区块节点, 根据坐标、宽高等节点信息, 从视觉稿原图中裁剪图像, 送入 UI 类型分类模型进行推理, 得到 UI 组件的类型。
- 其中, 可以将确定出的分类结果挂载在 JSON 文件中。
- 另外, 在一应用实施例中, 在步骤 701 之后, 可以跳过步骤 702、步骤 703, 直接执行步骤 704。还可以针对不同长度的视觉稿制作对应的目标检测数据集, 训练多个 UI 区块识别模型, 每个模型对应识别不同长度的输入图像。也就是说, 在步骤 701 输入结构化的视觉稿数据之后, 无需对页面进行分割, 而是直接执行步骤 704, 使用识别对应长度的 UI 区块识别模型进行图像识别。
- 此处, 结合图 8 示出的模型训练的实现流程图, 进一步说明如何生成模型训练的数据集和如何训练模型。
- 组件识别装置对真实的视觉稿进行标注, 标注信息包括组件位置与 UI 组件类型, 得到带有 UI 组件类型标签 (第一标签) 的目标检测数据集。通过修改 UI 组件类型标签为统一的 UI 区块标签 (第二标签) ‘ui’, 得到带有 UI 区块标签的目标检测数据集。基于目标检测数据集训练目标检测模型, 得到 UI 区块识别模型 (即设定识别模型)。
- 组件识别装置基于得到的带有 UI 组件类型标签的目标检测数据集, 通过对目标检测数据集进行裁剪转换, 得到带有 UI 组件类型标签的分类数据集, 并判断分类数据集中数据量较少的 UI 组件类型的样本数量是否满足设定条件, 通过生成数据量较少的 UI 组件类型的样本。基于裁剪得到和生成的分类数据集的样本, 训练分类模型, 得到 UI 组件类型分类模型 (即设定分类模型)。
- 其中, 生成样本的流程如下:
- 1) 渲染带有组件的页面。

可以采用两种方式，一种是在页面中渲染组件代码，另一种是使用通过视觉稿生成代码工具得到的还原视觉稿的网页；

2) 使用 Puppeteer 工具对页面中的组件节点属性进行抖动，包括文本更改、元素位置偏移等，并截图得到样本。

5 其中, Puppeteer 是一个 Node 库, 通过 DevTools Protocol 来控制 Chrome 或 Chromium, 基于提供的一系列 API, 可以模拟人工打开网页、点击等操作, 可以用于屏幕截图、自动化测试等。

相较于通过深度学习网络目标检测模型识别图像中 UI 组件的位置与类型, 为训练模型需要准备大量各类型的样本, 并对大量的样本进行标注, 由于真实视觉稿中组件的出现
10 概率不一, 导致不同类别样本的数量非常不均衡, 准备训练样本的难度较大。同时, 训练得到的模型只能识别长宽比在一定范围内的视觉稿, 模型的识别效果不佳。

此外, 在相关技术中也可以通过对视觉稿的每个节点进行截图, 使用训练的分类模型进行分类, 由于需要遍历每个节点进行分类, 模型计算量巨大, 计算成本高。

在本应用实施例中, 组件识别装置通过将组件识别分成两个阶段: 先通过目标检测识
15 别出 UI 区块, 即可能是组件的区域; 再对 UI 区块进行分类得到对应的 UI 组件类型, 极大减少了计算量。例如, 一个视觉稿中有 200 个节点, 有 10 个组件需要识别, 如果对视觉稿中每个节点进行遍历得到分类结果, 需要处理 200 个节点图像, 通过本申请的方案, 只需要在目标检测后, 对 10 张识别有区块节点的图像进行分类。

深度学习的训练需要大量的样本, 样本的质量极大程度决定了模型效果的上限。
20 由于不同类型的组件的使用频率不同, 造成数据集中不同组件类型的样本分布不均衡, 样本数量少的组件类型的识别、分类效果较差。并且, 模拟生成类似真实视觉稿的完整视觉稿图像样本, 所需的成本较高, 而生成单个 UI 组件的样本的扩充较为方便, 所需的成本更低。并且, 由于一些 UI 组件渲染得到的图像具有相似的轮廓特性, 且视觉上相似度较高, 基于这些 UI 组件对应的图像训练得到的 UI 区块识别模型, 也可以检测到出现频率较
25 少的 UI 组件的 UI 区块, 降低了目标检测数据集所需的样本数量。然后, 通过分类模型具体判断 UI 组件类型。在本申请应用实施例中, 通过两个模型处理图像, 通过复用训练样本, 只需要较少样本训练模型, 便可完成组件识别, 降低了获取训练样本的成本。同时, 通过生成训练样本, 克服了数据集中不同组件类型的样本分布不均衡的问题, 这样, 基于数据集训练分类模型, 提高了分类模型分类图像的精确度, 提升了视觉稿的 UI 组件类型
30 识别的效果。

为实现本申请实施例的方法,本申请实施例还提供了一种组件识别装置,如图9所示,该装置包括:

第一处理模块1001,被配置为将第一图像输入设定识别模型,得到所述设定识别模型输出的所述第一图像中的至少一个UI区块;所述第一图像基于第一视觉稿确定;所述设定识别模型被配置为在输入的图像中识别出至少一个UI区块;每个UI区块至少包括能够以UI组件渲染得到的图像区域;

第二处理模块1002,被配置为在第一视觉稿对应的DOM的节点中,确定与所述至少一个UI区块中的每个UI区块对应的一个第一节点;

第三处理模块1003,被配置为将每个第一节点对应的图像输入设定分类模型,得到所述设定分类模型输出的第一节点对应的UI组件类型;所述设定分类模型被配置为确定输入的图像对应的UI组件类型。

其中,在一个实施例中,所述第二处理模块1002,还被配置为:

遍历所述第一视觉稿对应的文档对象模型DOM的节点;

在区域重叠率大于第一设定阈值的情况下,确定所述区域重叠率对应的节点为第一UI区块对应的第一节点;其中,

所述区域重叠率表征第二图像与所述第一UI区块的重叠率;所述第二图像根据节点的节点信息对所述第一视觉稿裁剪得到。

在一个实施例中,所述装置还包括裁剪模块,被配置为:

通过边缘检测,在第一视觉稿中确定出至少两个第一矩形区域;

基于至少一个第一矩形区域裁剪所述第一视觉稿,得到对应的第一图像。

在一个实施例中,所述裁剪模块,还被配置为:

合并设定方向相邻的第一矩形区域,得到第二矩形区域;其中,所述第二矩形区域在设定方向的长度小于第二设定阈值;

基于所述第二矩形区域裁剪所述第一视觉稿。

在一个实施例中,所述将每个第一节点对应的图像输入设定分类模型,包括:

将根据第一节点的节点信息裁剪所述第一视觉稿得到的第二图像,输入所述设定分类模型;

和/或,

将根据第一节点对应的UI区块裁剪所述第一视觉稿得到的第三图像,输入所述设定分类模型。

在一个实施例中，所述装置还包括训练模块，被配置为：

基于对应至少有至少一个第一标注的第二视觉稿，训练得到所述设定识别模型和所述设定分类模型；其中，

5 所述第一标注包括能够以 UI 组件渲染得到的图像区域的第一位置信息和对应的第一标签；第一位置信息被配置为描述所述图像区域在第二视觉稿的位置；所述第一标签被配置为描述所述 UI 组件的类型。

在一个实施例中，所述训练模块，还被配置为：

基于每个第一标注裁剪所述第二视觉稿，得到对应的第四图像；所述第四图像对应第一标签；

10 将所述第二视觉稿的第一标签替换为第二标签，得到第五图像；所述第二标签表征对应的图像区域为 UI 区块；

基于第四图像训练得到所述设定分类模型；

基于第五图像训练得到所述设定识别模型。

在一个实施例中，所述装置还包括生成模块，被配置为：

15 在至少一种第一标签对应的第四图像的数量小于第三设定阈值的情况下，生成包含对应 UI 组件类型的页面图像，并基于生成的页面图像得到对应的第四图像。

在一个实施例中，所述生成包含对应 UI 组件类型的页面图像，包括：

在设定界面渲染对应的 UI 组件类型的组件代码，得到对应 UI 组件类型的页面图像；和/或，

20 通过视觉稿生成代码工具得到第三视觉稿，得到对应 UI 组件类型的页面图像。

实际应用时，所述第一处理模块 1001、所述第二处理模块 1002、所述第三处理模块 1003、所述裁剪模块、所述训练模块、所述生成模块可由基于组件识别装置中的处理器，比如中央处理器（CPU，Central Processing Unit）、数字信号处理器（DSP，Digital Signal Processor）、微控制模块（MCU，Microcontroller Unit）或可编程门阵列（FPGA，Field -
25 Programmable Gate Array）等实现。

需要说明的是：上述实施例提供的组件识别装置在进行组件识别时，仅以上述各程序模块的划分进行举例说明，实际应用中，可以根据需要而将上述处理分配由不同的程序模块完成，即将装置的内部结构划分成不同的程序模块，以完成以上描述的全部或者部分处理。另外，上述实施例提供的组件识别装置与组件识别方法实施例属于同一构思，其具体
30 实现过程详见方法实施例，其中不再赘述。

基于上述程序模块的硬件实现，且为了实现本申请实施例组件识别方法，本申请实施例还提供了一种电子设备。图 10 为本申请实施例电子设备的硬件组成结构示意图，如图 10 所示，电子设备包括：

通信接口 1，能够与其它设备比如网络设备等进行信息交互；

5 处理器 2，与通信接口 1 连接，以实现与其它设备进行信息交互，用于运行计算机程序时，执行上述一个或多个技术方案提供的方法。而所述计算机程序存储在存储器 3 上。

当然，实际应用时，电子设备中的各个组件通过总线系统 4 耦合在一起。可理解，总线系统 4 用于实现这些组件之间的连接通信。总线系统 4 除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，在图 10 中将各种总线都标为总线系统 4。

本发明实施例中的存储器 3 用于存储各种类型的数据以支持电子设备的操作。这些数据的示例包括：用于在电子设备上操作的任何计算机程序。

可以理解，存储器 3 可以是易失性存储器或非易失性存储器，也可包括易失性和非易失性存储器两者。其中，非易失性存储器可以是只读存储器（ROM，Read Only Memory）、
15 可编程只读存储器（PROM，Programmable Read-Only Memory）、可擦除可编程只读存储器（EPROM，Erasable Programmable Read-Only Memory）、电可擦除可编程只读存储器（EEPROM，Electrically Erasable Programmable Read-Only Memory）、磁性随机存取存储器（FRAM，ferromagnetic random access memory）、快闪存储器（Flash Memory）、磁表面存储器、光盘、或只读光盘（CD-ROM，Compact Disc Read-Only Memory）；磁表面存储器
20 可以是磁盘存储器或磁带存储器。易失性存储器可以是随机存取存储器（RAM，Random Access Memory），其用作外部高速缓存。通过示例性但不是限制性说明，许多形式的 RAM 可用，例如静态随机存取存储器（SRAM，Static Random Access Memory）、同步静态随机存取存储器（SSRAM，Synchronous Static Random Access Memory）、动态随机存取存储器（DRAM，Dynamic Random Access Memory）、同步动态随机存取存储器（SDRAM，
25 Synchronous Dynamic Random Access Memory）、双倍数据速率同步动态随机存取存储器（DDRSDRAM，Double Data Rate Synchronous Dynamic Random Access Memory）、增强型同步动态随机存取存储器（ESDRAM，Enhanced Synchronous Dynamic Random Access Memory）、同步连接动态随机存取存储器（SLDRAM，SyncLink Dynamic Random Access Memory）、直接内存总线随机存取存储器（DRRAM，Direct Rambus Random Access
30 Memory）。本发明实施例描述的存储器 2 旨在包括但不限于这些和任意其它适合类型的存

存储器。

上述本发明实施例揭示的方法可以应用于处理器 2 中，或者由处理器 2 实现。处理器 2 可能是一种集成电路芯片，具有信号的处理能力。在实现过程中，上述方法的各步骤可以通过处理器 2 中的硬件的集成逻辑电路或者软件形式的指令完成。上述的处理器 2 可以是通用处理器、DSP，或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等。处理器 2 可以实现或者执行本发明实施例中的公开的各方法、步骤及逻辑框图。通用处理器可以是微处理器或者任何常规的处理器等。结合本发明实施例所公开的方法的步骤，可以直接体现为硬件译码处理器执行完成，或者用译码处理器中的硬件及软件模块组合执行完成。软件模块可以位于存储介质中，该存储介质位于存储器 3，处理器 2 读取存储器 3 中的程序，结合其硬件完成前述方法的步骤。

处理器 2 执行所述程序时实现本发明实施例的各个方法中的相应流程，为了简洁，在此不再赘述。

在示例性实施例中，本发明实施例还提供了一种存储介质，即计算机存储介质，具体为计算机可读存储介质，例如包括存储计算机程序的存储器 3，上述计算机程序可由处理器 2 执行，以完成前述方法所述步骤。计算机可读存储介质可以是 FRAM、ROM、PROM、EPROM、EEPROM、Flash Memory、磁表面存储器、光盘、或 CD-ROM 等存储器。

在本申请所提供的几个实施例中，应该理解到，所揭露的装置、电子设备和方法，可以通过其它的方式实现。以上所描述的设备实施例仅仅是示意性的，例如，所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，如：多个单元或组件可以结合，或可以集成到另一个系统，或一些特征可以忽略，或不执行。另外，所显示或讨论的各组成部分相互之间的耦合、或直接耦合、或通信连接可以是通过一些接口，设备或单元的间接耦合或通信连接，可以是电性的、机械的或其它形式的。

上述作为分离部件说明的单元可以是、或也可以不是物理上分开的，作为单元显示的部件可以是、或也可以不是物理单元，即可以位于一个地方，也可以分布到多个网络单元上；可以根据实际的需要选择其中的部分或全部单元来实现本实施例方案的目的。

另外，在本申请各实施例中的各功能单元可以全部集成在一个处理单元中，也可以是各单元分别单独作为一个单元，也可以两个或两个以上单元集成在一个单元中；上述集成的单元既可以采用硬件的形式实现，也可以采用硬件加软件功能单元的形式实现。

本领域普通技术人员可以理解：实现上述方法实施例的全部或部分步骤可以通过程序指令相关的硬件来完成，前述的程序可以存储于一计算机可读取存储介质中，该程序在执

行时，执行包括上述方法实施例的步骤；而前述的存储介质包括：移动存储设备、ROM、RAM、磁碟或者光盘等各种可以存储程序代码的介质。

或者，本申请上述集成的单元如果以软件功能模块的形式实现并作为独立的产品销售或使用时，也可以存储在一个计算机可读取存储介质中。基于这样的理解，本申请实施例的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备（可以是个人计算机、服务器、或者网络设备等）执行本申请各个实施例所述方法的全部或部分。而前述的存储介质包括：移动存储设备、ROM、RAM、磁碟或者光盘等各种可以存储程序代码的介质。

需要说明的是，本申请实施例所记载的技术方案之间，在不冲突的情况下，可以任意组合。除非另有说明和限定，术语“连接”应做广义理解，例如，可以是电连接，也可以是两个元件内部的连通，可以是直接相连，也可以通过中间媒介间接相连，对于本领域的普通技术人员而言，可以根据具体情况理解上述术语的具体含义。

另外，在本申请实例中，“第一”、“第二”等是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解“第一\第二\第三”区分的对象在适当情况下可以互换，以使其中描述的本申请的实施例可以除了在其中图示或描述的那些以外的顺序实施。

本文中术语“和/或”，仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。另外，本文中术语“至少一个”表示多个中的任意一个或多个中的至少两个的任意组合，例如，包括A、B、C中的至少一个，可以表示包括从A、B和C构成的集合中选择的任意一个或多个元素。

以上所述，仅为本申请的具体实施方式，但本申请的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本申请揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应以所述权利要求的保护范围为准。

在具体实施方式中所描述的各个实施例中的各个具体技术特征，在不矛盾的情况下，可以进行各种组合，例如通过不同的具体技术特征的组合可以形成不同的实施方式，为了避免不必要的重复，本申请中各个具体技术特征的各种可能的组合方式不再另行说明。

权利要求书

1、一种组件识别方法，所述方法包括：

将第一图像输入设定识别模型，得到所述设定识别模型输出的所述第一图像中的至少一个用户界面 UI 区块；所述第一图像基于第一视觉稿确定；所述设定识别模型用于在输入的
5 图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域；

在第一视觉稿对应的文档对象模型 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点；

将每个第一节点对应的图像输入设定分类模型，得到所述设定分类模型输出的第一节点对应的 UI 组件类型；所述设定分类模型用于确定输入的图像对应的 UI 组件类型。

10 2、根据权利要求 1 所述的方法，其中，所述在第一视觉稿对应的文档对象模型 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点，包括：

遍历所述第一视觉稿对应的文档对象模型 DOM 的节点；

在区域重叠率大于第一设定阈值的情况下，确定所述区域重叠率对应的节点为第一 UI 区块对应的第一节点；其中，

15 所述区域重叠率表征第二图像与所述第一 UI 区块的重叠率；所述第二图像根据节点的节点信息对所述第一视觉稿裁剪得到。

3、根据权利要求 1 所述的方法，其中，在所述将第一图像输入设定识别模型之前，所述方法还包括：

通过边缘检测，在第一视觉稿中确定出至少两个第一矩形区域；

20 基于至少一个第一矩形区域裁剪所述第一视觉稿，得到对应的第一图像。

4、根据权利要求 3 所述的方法，其中，在所述基于至少一个第一矩形区域裁剪所述第一视觉稿，包括：

合并设定方向相邻的第一矩形区域，得到第二矩形区域；其中，所述第二矩形区域在设定方向的长度小于第二设定阈值；

25 基于所述第二矩形区域裁剪所述第一视觉稿。

5、根据权利要求 1 所述的方法，其中，所述将每个第一节点对应的图像输入设定分类模型，包括：

将根据第一节点的节点信息裁剪所述第一视觉稿得到的第二图像，输入所述设定分类模型；

30 和/或，

将根据第一节点对应的 UI 区块裁剪所述第一视觉稿得到的第三图像，输入所述设定分类模型。

6、根据权利要求 1 所述的方法，其中，在所述将第一图像输入设定识别模型之前，所述方法还包括：

- 5 基于对应至少有至少一个第一标注的第二视觉稿，训练得到所述设定识别模型和所述设定分类模型；其中，

所述第一标注包括能够以 UI 组件渲染得到的图像区域的第一位置信息和对应的第一标签；第一位置信息用于描述所述图像区域在第二视觉稿的位置；所述第一标签用于描述所述 UI 组件的类型。

- 10 7、根据权利要求 6 所述的方法，其中，所述基于对应至少有至少一个第一标注的第二视觉稿，训练得到所述设定识别模型和所述设定分类模型，包括：

基于每个第一标注裁剪所述第二视觉稿，得到对应的第四图像；所述第四图像对应有一标签；

- 15 将所述第二视觉稿的第一标签替换为第二标签，得到第五图像；所述第二标签表征对应的图像区域为 UI 区块；

基于第四图像训练得到所述设定分类模型；

基于第五图像训练得到所述设定识别模型。

8、根据权利要求 7 所述的方法，其中，在所述基于第四图像训练所述设定分类模型之前，所述方法还包括：

- 20 在至少一种第一标签对应的第四图像的数量小于第三设定阈值的情况下，生成包含对应 UI 组件类型的页面图像，并基于生成的页面图像得到对应的第四图像。

9、根据权利要求 8 所述的方法，其中，所述生成包含对应 UI 组件类型的页面图像，包括：

- 25 在设定界面渲染对应的 UI 组件类型的组件代码，得到对应 UI 组件类型的页面图像；
和/或，

通过视觉稿生成代码工具得到第三视觉稿，得到对应 UI 组件类型的页面图像。

10、一种组件识别装置，包括：

- 30 第一处理模块，被配置为将第一图像输入设定识别模型，得到所述设定识别模型输出的所述第一图像中的至少一个 UI 区块；所述第一图像基于第一视觉稿确定；所述设定识别模型被配置为在输入的图像中识别出至少一个 UI 区块；每个 UI 区块至少包括能够以 UI 组件渲染得到的图像区域；

第二处理模块，被配置为在第一视觉稿对应的 DOM 的节点中，确定与所述至少一个 UI 区块中的每个 UI 区块对应的一个第一节点；

第三处理模块，被配置为将每个第一节点对应的图像输入设定分类模型，得到所述设定分类模型输出的第一节点对应的 UI 组件类型；所述设定分类模型被配置为确定输入的图像
5 对应的 UI 组件类型。

11、一种电子设备，包括：处理器和用于存储能够在处理器上运行的计算机程序的存储器，

其中，所述处理器用于运行所述计算机程序时，执行权利要求 1 至 9 任一项所述方法的步骤。

10 12、一种存储介质，其上存储有计算机程序，所述计算机程序被处理器执行时实现如权利要求 1 至 9 任一项所述方法的步骤。