UI2Vec：多模态 UI 图像通用嵌入方法

计算机科学与技术

多媒体技术 (101030)

完成时间：2023 年 6 月 20 日

1 概述（选题背景与选题意义）

在互联网和移动设备盛行的时代，我们每天都要和大量的 UI 界面打交道。然而，不同的企业和系统在设计 UI 界面时各有差异，而逐渐臃肿的应用程序功能正在使 UI 界面愈发难以理解。目前，这一趋势已经影响到了不同人群对于手机应用功能的可及性。例如，对于老年人而言，由于年龄的影响，学习各类新功能往往需要在外人的帮助下，通过不断的重复和巩固才能学会。

因此，本任务的出发点为解决**老年人使用手机难**这一问题。虽然看起来这是一个人机交互领域的问题，但其中需要应用多类多媒体技术来为交互层面上的任务提供支持。

我们不妨从系统顶层设计的角度出发，对这一系统需要实现的功能进行建模。我们提出 InstructUI 系统，图1展现了其需要实现的功能。

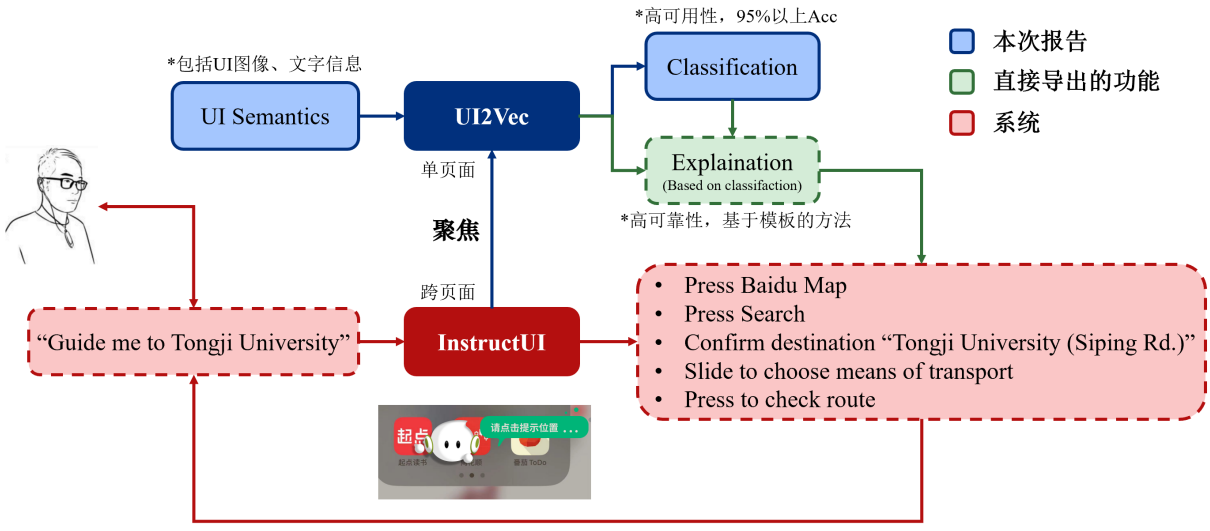

图 1: InstructUI 系统实现路线图

InstructUI 系统是一个语音界面驱动的交互教程系统。用户通过输入手机操作需求（例如：我怎么去同济大学?），InstructUI 系统通过结合语义信息、手机内应用程序元信息，生成一系列操作指令，由悬浮在界面上的助手提供使用引导。总结而言，InstructUI 系统应当具备语音语义理解、应用程序功能理解、跨页面 UI 语义理解能力。

对于本报告而言，我们聚焦于实现 InstructUI 的子模块 UI2Vec。UI2Vec 是对于 InstructUI 系统需求的聚焦和简化。在 UI2Vec 中，我们仅关注单页面的 UI 语义信息。

UI2Vec 模型想解决的核心问题为“如何使机器理解 UI 界面”。为了实现这一目标，在本文中，我们对 UI 语义进行了定义，并使用自然语言处理和计算机视觉的技术来达成这一目标。

实际上，UI 语义往往已经被定义在应用程序之中。然而，绝大多数的应用程序在接口层面均无法提供完整的 UI 元信息。Ross [1] 对于安卓程序的可及性进行了调研，所有 100 个下载次数最多的 Android 应用程序都存在基本的使用可及性问题（即无法获取 UI 组件元信息）。我们使用计算机视觉技术来突破这一障碍。

同时，我们以分类作为我们的训练任务。这一选择同样具有两个原因：首先，当前很多预训练模型均是通过分类问题实现预训练的。例如，计算机视觉模型很多都是 ImageNet [2] 数据集上预训练生成的。经过调研发现，当今对于 UI 界面整体进行分类的研究较少，且分类准确率较低，远未达到可用的级别（当前约 75% 准确率）。同时，类别信息又是描述 UI 界面的一种关键信息，可以作为后续生成 UI 界面功能解释时的良好约束条件。

总结而言，本工作的主要意义有如下 3 点：

- 我们通过扩展 Enrico 数据集的信息，构建了一个包含视觉信息嵌入向量和文字信息嵌入向量的数据集；
- 我们提出 UI2Vec 模型，一个基于嵌入向量的表示学习模型，用于融合视觉信息和文字信息，为诸多 UI 相关的下游任务提供了新的实现方法，并在分类任务上取得了优异的性能；
- 我们总结了在 UI2Vec 模型设计和验证中所总结的经验，并提出了若干针对上述问题的潜在解决方案和未来的研究方向；
- 为 InstructUI 系统实现提供了技术基础。

2 相关工作

2.1 UI 界面数据集

当前，在开源领域和非开源领域之中，已经逐渐涌现了一系列用于分析 UI 界面的数据集。例如，AMP 数据集 [3] 包含来自 4,068 个 iOS 应用程序的 77,000 个截屏，用于训练增强型屏幕阅读器以支持无障碍功能。最大的公开可用数据集为 Rico [4]，包含由来自 9.7K 安卓应用程序的 72K 应用程序 UI 图像，当前仍是研究手机端 UI 界面的主要数据来源。

Rico 诞生之后，有若干工作着手解决 Rico 数据集中存在的问题。例如，Enrico [5] 从 Rico 中随机抽取了 10,000 个样本，然后对其中的 1460 个样本进行清理并提供额外的注释。VINS 数据集 [6] 是一个用于 UI 元素检测的数据集，它通过收集和手动截取多个来源的屏幕截图而创建的。Clay 数据集（规模为 60K） [7] 是通过自动机器学习模型和人工注释实现了对 Rico 的去噪声，以提供 UI 元素的标签。然而，这些数据集的创建和更新的成本很高，而且随着 UI 设计范式发生改变，对于现实的参考价值也在不断降低。Rico 数据集在 2017 年采集之后，便没有出现更新，目前在开源领域页仍未出现新的 UI 界面数据集。

除了上述在移动设备领域的 UI 数据集，随着互联网在各设备端的融合趋势加速，某些 UI 设计实现了在多类设备上通用。这些 UI 设计往往来自互联网，因此基于网页端的数据集应运而生。

例如，Webzegeist [8] 抓取了 103,744 个网页和相关的网页元素，并实现了网页元素和 HTML 元素的匹配。然而，受限于年代，这一数据集并未考虑其在其它设备端的查看情况。为解决这一问题，WebUI [9] 构建了迄今为止最大的，从网页端爬取的 UI 数据集。WebUI 数据集包含 400,000 个使用多类设备模拟访问的网页，并自动将网页与视觉、语义等信息相匹配。由于包含多种设备的图像信息，这些信息可以用于研究多种设备间在 UI 设计上的差异。

本项目中，受限于时间和计算资源限制，我们采用上述数据集中最小的 Enrico 数据集验证我们的模型性能。我们在 Enrico 数据集上增加了基于 OCR 识别的 UI 图像文字部分，用于增强 Enrico 数据集。这一部分的数据被用来训练我们的多模态表示学习模型 UI2Vec。

2.2 UI 界面理解及应用系统

UI 界面理解是一个跨越多个领域的研究议题，例如多媒体、人机交互及软件工程。研究者往往从视觉角度出发，通过目标识别、图像分类等任务获取对 UI 界面的理解，并结合特定场景多模态的输入输出，完成特定的任务。

例如，VASTA [10] 通过 RetinaNet 进行 UI 目标检测，通过 OCR 标记 UI 元素。基于机器对于 UI 界面的理解，VASTA 通过像素相似度计算判断界面关联性，构建了一个演示编程系统（PBD，Programming By Demonstration），使得手机可以在任意应用程序上通过一定的预设模式自动执行由自然语言输入的指令。Lexi [11] 则完成了一个 UI 说明任务（Image Caption）。Lexi 的 UI 界面基于传统的图像处理方式，无需预训练视觉模型，通过边缘、梯度检测确定 UI 元素位置，并使用 ResNeXt-101 对 UI 元素分类，使用 OCR 标记 UI 元素。继而，Lexi 通过预训练的语言模型 RoBERTA 生成 UI 说明。GUIComp [12] 则从 UI 设计的角度出发，使用编解码器结构的深度学习模型获得 UI 界面知识，并使用模型的瓶颈层用来完成 UI 界面的推荐，用于迭代设计过程。

本项目提出了一种融合视觉信息和文字信息的新的 UI 界面理解方案。和目前对于 UI 元素理解的方法不同，本项目中的关注对象为整个 UI 界面，而非某种 UI 元素。同时，对比通过视觉方法获取整个 UI 界面语义的方法而言，本项目通过之前较少使用的分类问题预训练视觉模型，同时融入了文字语义信息。最终，由编码器生成的 UI 理解向量可以被用于若干下游任务之上。

3 数据集

3.1 数据集介绍

本项目中，使用 Enrico 数据集 [5] 展开实验。

Enrico 数据集脱胎于 Rico 数据集 [4]。Rico 数据集是公开访问的迄今为止最大的手机应用设计数据集。作者从 27 个手机应用类别中挖掘了超过 9700 个免费的 Android 应用程序来创建 Rico 数据集。数据集中的应用程序的平均用户评分为 4.1。Rico 数据集包含超过 72K 个独特的 UI 界面和 3M 个 UI 元素的视觉、文本、结构以及 10.8K 条交互记录。

然而，由于 Rico 数据集是通过网络爬虫和众包等方式构建的，研究表明，Rico 的数据较为嘈杂，有相当含量的标注错误，仅有 10% 左右的有效信息。这一比例也被 Lee 等人 [12] 在创建 GUI 设计系统时证明。因此，在 Rico 的基础上，Enrico 通过手工筛选的方式构建出了一个更高质量的

用户界面数据集。由于是手工完成的数据集，因此作者随机在 Rico 中抽取了 10,000 张 UI 设计，通过人工交叉验证的方式，最终保留了 1460 张 UI 设计。

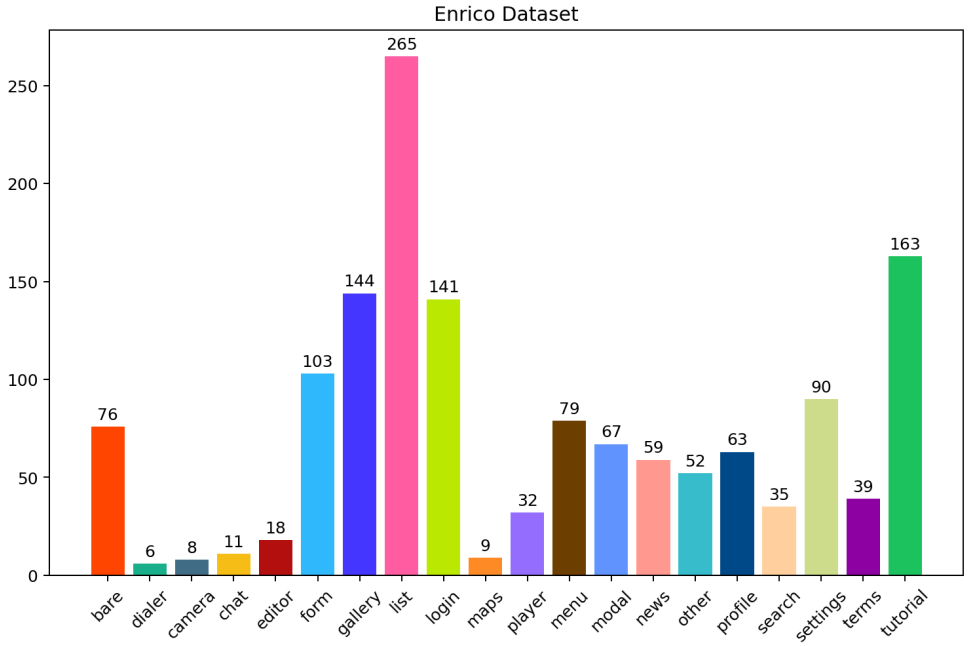

图 2: Enrico 数据集

同时，针对这 1460 张图片，作者还展开了分类标注。图2展现了分类的情况及各类的数量。这是 Enrico 相对 Rico 扩展的信息。本文中，UI2Vec 模型采用分类问题对视觉模型进行预训练，因此这部分信息是不可或缺的。

3.2 数据组织

在 UI2Vec 训练过程中，我们并不会使用到所有 Enrico 中包含的信息，仅使用原始图像和 UI 类别信息。

根据 Multibench 数据集 [13] 中对于 Enrico 数据集的处理方法，我们剔除了 50105，50109 两张图片。这是由于两张图片的线框图（Enrico 中的另一类信息）无法正确加载。为了避免和后续可能的应用冲突，本文中同样不考虑这两张图片。

同时，为了保证 OCR 在不同的图像上识别到的具体位置绝对坐标具备可比性，我们还对所有图像进行了尺寸的统一（400×800）。对于计算机视觉的任务，在输入时规定两种固定尺寸，即 224×224 和 256×256。

经过以上处理，本文使用的 Enrico 数据集共 1458 个条目，所有图像统一规格为 400×800。

4 方法

4.1 UI 语义信息表示

用户界面（UI）是机器和人之间信息交流的媒介，通过多种信息信道进行信息交换。在 UI 界面中，不同的信息信道承担了不同的交流任务。例如，人们可以通过颜色、形状、大小、位置等

视觉信息来判断界面中的功能分区、按钮等要素。同时，通过阅读文字的方式可以获知具体的功能或展示的内容。

虽然现代 UI 界面往往还会通过声音、震动等其它信息通道和人产生交互，但相比于上述两个视觉能够直接感知的维度，在影响人的感知中仅占据很小的一部分。

总结而言，人们感知 UI 界面时，最重要的元素是视觉要素，而视觉要素可以划分为文字要素和线框要素。

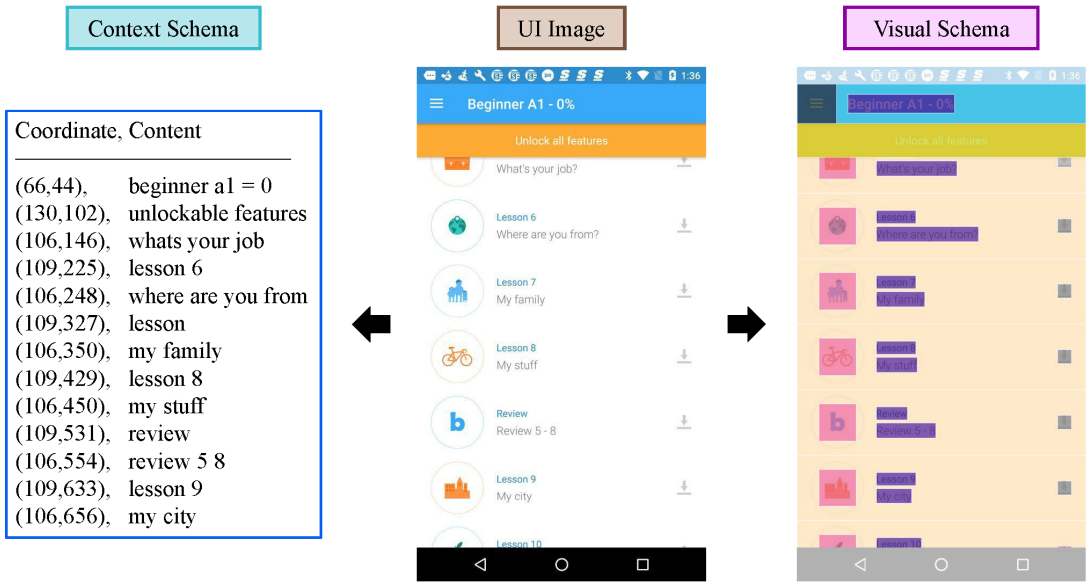

图 3: UI 语义信息定义

图3展现了 Enrico 数据集编号为 2941 的 UI 图像。图3左侧即 UI 界面的文字要素，在本文中我们称为文字语义部分（Context Schema）。图3右侧即 UI 界面的线框要素，往往以颜色、图标等形式呈现，在本文中我们称之为视觉语义部分（Visual Schema）。两种语义相结合，才能完整的表现一张 UI 图像给用户的所有信息。例如，如果没有文字语义，用户便仅能猜测这是一张菜单界面，而无法猜测处这是一个关于初学者课程的选择菜单。反之同理。

由此，我们正规的给出 UI 界面语义信息定义（公式1）：

$$semantics^{(i)} = \left\{visual^{(i)}, context^{(i)}\right\} \tag{1}$$

其中，使用 i 表示样本，$semantics$ 表示 UI 界面语义，$visual$ 表示 UI 视觉语义，$context$ 表示 UI 文字语义。

接下来的挑战，就是如何让计算机读懂 UI 视觉语义和 UI 界面含义。

4.2 UI2Vec 模型

针对上文的挑战，我们提出 UI2Vec 模型视图让机器更好的理解 UI 界面。以往的研究往往仅仅聚焦于语义中的视觉语义部分，而忽略了文字语义。因此，UI2Vec 是一个具备多模态信息输入

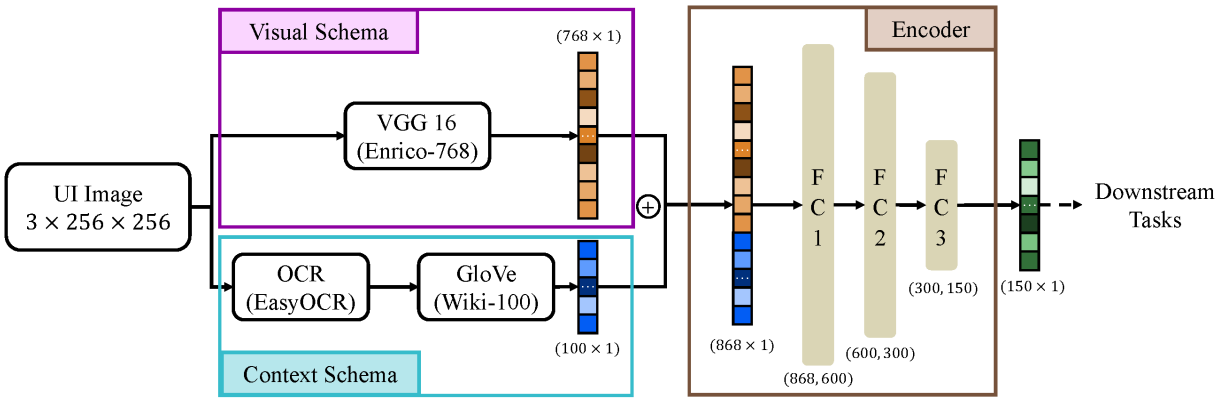

图 4: UI2Vec 模型架构

能力的模型。这一章，我们首先介绍 UI2Vec 的整体架构，继而对各子模块的实现和细节进行阐述。

4.2.1 整体架构

图4展现了 UI2Vec 模型的整体架构。UI2Vec 模型主要分为三个部分：视觉语义模块、文字语义模块和编码器模块。

视觉模块通过计算机视觉技术，使用一种类似于 VGG16 [14] 的模型，通过抽取全连接层中的 768 维嵌入向量作为视觉信息表征。这一部分的模型设计将在第4.2.2节中阐述。

语义模块则通过开源的 EasyOCR [15] 模型提取文字信息，通过 gensim [16] 中的 GloVe [17] 模型（'wiki-gigaword-100'）生成 100 维词嵌入向量作为语义信息表征。

将 768 维的视觉嵌入向量和 100 维的词嵌入向量经过归一化后直接拼接，生成 868 维的原始嵌入向量，输入编码器部分。编码器部分是一个三层全连接神经网络，对原始嵌入变量进行降维和非线性变换，输出 150 维的嵌入向量。这一 150 维的嵌入向量即本模型的输出，用以表征 UI 语义信息。

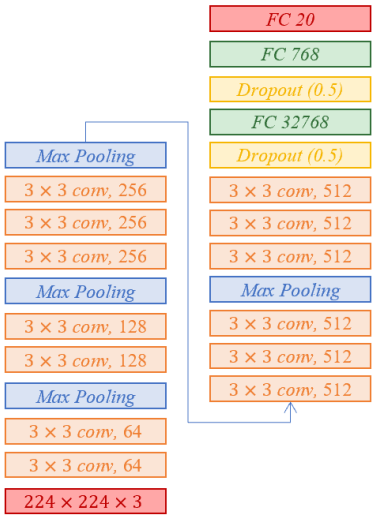

图 5: 本文中 VGG 模型架构

4.2.2 视觉语义嵌入模块

模型选择：目前，迁移学习方法已被广泛应用。然而，这些模型往往是在自然图像数据集上训练而成的，并不适用于我们的任务。因此，我们需要自己训练一个新的模型。我们考虑了现代计算机视觉方法中的两类代表：卷积神经网络模型（VGG [14]）以及 ViT 模型 [18]。

两类模型在数据处理方法上有较大不同，且参数量也有一定差异。VGG 通过多个卷积层叠加的方式，训练多个卷积核提取图像有效信息。ViT 则是通过将图像展开为 16×16 的输入串，将图像类比为自然语言，以图片串的方式输入，对图像进行训练。在参数数量上，由于卷积层的引入，大幅减少了参数数量。因此，本项目使用的 VGG 的参数量大约是 ViT 的一半。

为了统一嵌入输出层的参数，我们将两个模型全连接层用于输出嵌入向量的向量维度统一为 768。这一参数的设计使得我们无需对 PyTorch 中的 ViT 模型进行过多修改。然而，对于 VGG 模型，由于希望获得 768 维的嵌入向量，需要对原始的 VGG16 模型进行一定改造。本文中，使用的基于 VGG 的模型架构如图5。具体而言，相比于原始的 VGG 模型，我们增加了卷积层和 dropout 层。

为了选择出最适合 Enrico 数据集的模型，我们进行了模型选择测试。按照 Enrico 论文 [5] 中的做法，我们将训练数据、验证数据和测试数据按照 $65:15:20$ 的比例进行划分。两个模型均采用 Adam 优化器，采用交叉熵损失函数，使用批数量为 64。在每一轮训练集训练完成后，在验证集上验证分类准确率。当验证集准确率达到相对稳定、且损失函数值不再有显著下降时，认为模型收敛。继而，在测试集上对两个模型的性能进行比对。上述模型的实现基于 PyTorch 2.0，并使用 RTX 3090(24G, cuda 11.8) 显卡加速训练过程。

训练结果在第5.1节呈现。总结而言，ViT 在 Enrico 这个较小的数据集上的表现并不理想，在使用相同划分的情况下不如 VGG 的性能，且参数数量更大。因此，我们选用基于 VGG 的视觉模型作为视觉嵌入模型。

预训练模型：使用 Enrico 数据集中的**所有数据**对模型进行预训练。**这是一个非常规的做法**。然而，由于 Enrico 数据集的类别不平等性较大，按照传统的划分方法可能会丢失某些类别的特征。这一问题应当在后续研究中通过扩充 Enrico 数据，使数据量增大且分布较为均匀后再训练，以生成合适的模型。为了避免模型因为训练方式过拟合，我们在采用验证集进行监督约束的同时，并在其性能略超过 Enrico 论文 [5] 中的结果时中止训练。我们将此次训练的结果生成的模型称为 'Enrico-768'，表示是在 Enrico 训练的，能够输出 768 维度视觉嵌入向量的基于 VGG 方法的模型。

图6展现了 VGG 的预训练模型的训练过程。可以看出，验证集的验证准确率收敛，同时损失基本不再下降。

在此之后，我们将统一为 224×224 的 UI 图像输入至 'Enrico-768' 模型中，输出 768 维嵌入向量。我们使用此向量作为视觉语义的嵌入向量。我们认为，这一向量应当能较好的表征 UI 图像的视觉特征。

4.2.3 文字语义嵌入模块

虽然在文字语义嵌入模块中使用的模型均为预训练模型，但其中仍存在若干处理细节和挑战。

剔除多余的噪声信息：由于 UI 图像截图中，往往会保留智能手机的工具栏，因此在顶层的信

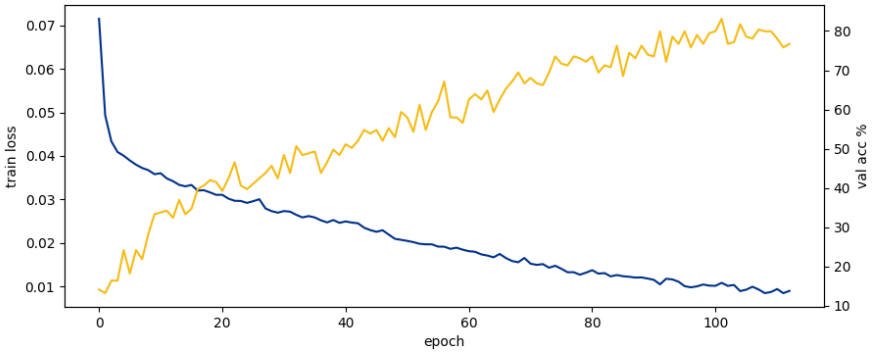

图 6: VGG 预训练模型训练过程

息，例如时间、电量等信息并不能有效的反映 UI 界面的实际文字语义。为解决这一问题，我们首先将所有图像统一为 800×400（高、宽），用于确定文字的具体位置。继而，通过启发式方法，将高度在 20 以内的识别文字进行剔除。这一操作一定程度会导致部分有效信息丧失，但我们认为这一操作可以显著提升语义信噪比。

识别错误：实验中，我们发现 EasyOCR [15] 的识别性能并不十分稳定。对于 UI 图像，常常会出现单词拼写错误的情况。由于 GloVe 模型 [17] 仅能支持在词典中的词向量生成，因此，大量拼写错误将使得 UI 界面的语义大量丧失，影响嵌入向量训练效果。本实验使用的预训练模型时 gensim [16] 提供的 'glove-wiki-gigaword-100' 模型，模型词典中包含 400,000 词，均来自维基百科。因此，我们对于 UI 界面中可能出现的词语进行一个强假设：假设不在词典中的词语均出现了拼写错误，且原始的正确拼写词语应当能在词典中找到。根据这一假设，我们通过 Python 内置的 difflib 模糊匹配算法，对不在 'glove-wiki-gigaword-100' 词典中的词语进行替换。由此，我们可以尽量确保在不破坏语义的情况下，保留尽可能多的语义信息。

解决无文字的情形：UI 图像不一定是一个富文字的场景，因此首先需要考虑的问题是如何处理无文字或无法识别文字的情形。鉴于 Enrico [5] 数据集中的图片主要语言为英文，因此我们设置 EasyOCR [15] 的语言模式为 'en'。因此，无文字的情况也可能出现在 UI 界面语言不是英文的 UI 图片中。在 UI2Vec 模型中，我们通过初始化一个 100 维的全零向量来规避这一问题。如此，当词嵌入向量被用于训练编码器时，对应的神经元的输入为 0，可以理解为此时不激活此神经元。

词向量生成：对于每一个 UI 界面，其中都会包含多条文字，文字之间可能存在语义关系或不存在语义关系。GloVe [17] 指出，生成的词向量在空间中具备一定的聚类关系，即相似的词语分布在相近的空间位置，而不同的词语分布在较远的位置。同时，我们可以通过向量的数值运算比较两个由若干单词组成的句子的语义相似性。在 UI2Vec 模型中，我们在完成词语分词后，将所有词语的连接视为一个句子，直接输入 'glove-wiki-gigaword-100' 获得各单词的词向量，并对输出求平均，获得 UI 界面的平均词向量。我们认为，这一向量将有效的表征 UI 功能语义信息。

4.2.4 编码器

在 UI2Vec 模型中，编码器的作用是融合视觉语义和文字语义，并通过特征选择和降维，将 UI 语义压缩至一个 150 维的向量之中。具体而言，在编码器中执行了两个操作：

特征拼接：编码器接收视觉嵌入向量和文字嵌入向量。这两个向量分别来自两个单独的模型，

由于均为中间层输出，两者的数值大小可能差异很大，不能直接用来拼接。因此，在拼接向量前，我们对两个向量分别做了归一化处理，使得两个向量的数值大小近似。此后，前 768 维划分给视觉嵌入向量，后 100 维划分给文字嵌入向量。这里，我们没有使用按位置的嵌入方法。这是因为，虽然我们获得的词语具备位置信息，但在词向量生成过程中，已经丢失了位置信息。除此之外，在视觉嵌入模型由 2D 矩阵转化为 1D 向量的过程中，位置信息也出现了丢失。因此，在此处进行位置嵌入并不存在可行性。同时，由于全连接神经网络的特征，所有输入神经元实际具有相同的地位，因此进行位置嵌入可能不会对实验结果造成影响。

特征融合：特征融合的过程体现在数据维度下降的过程中。在三层神经网络之中，维度不断下降至 150 维。根据全连接神经网络的特征，在运算过程中，来自不同模态的特征实现了融合，并且通过权重的设置，可以实现特征选择，保留和 UI 语义最相关的特征。

4.2.5 损失函数

与自然图像不同，从视觉上可以发现，UI 图像的特征空间较小，冗余出现特征的混合。为了解决这一问题，我们在编码器训练时并没有采用传统的分类问题训练。

Triplet 损失函数（公式2）[19] 可能是对于我们的研究问题较为合适的损失函数。

$$J(\theta) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \max\left\{\left[d\left(v_a^{(i)}, v_p^{(i)}\right) - d\left(v_a^{(i)}, v_n^{(i)}\right) + m\right], 0\right\} \tag{2}$$

其中，θ 代表参数，N 代表样本数，i 代表样本，v 代表嵌入向量，下标 a、p、n 分别代表基准向量（anchor）、正例向量（positive）、反例向量（negative）。参数 m 为偏置量，通常设置为 0.5，用于将损失函数保持在 0 以上。同时，采用 *max* 函数保证损失函数不会降到小于 0。

对于 UI2Vec 训练任务，正例向量即 UI 属于某类型的向量，而反例向量则为不属于某类型的向量。损失函数的训练要求类内间距尽可能小、类间间距尽可能大。依据此方法进行分类，数据表现出来的特征应类似于聚类的效果。

4.2.6 编码器训练

我们首先对训练编码器的数据进行描述。

数据增广：训练编码器前，我们需要对生成的嵌入向量进行数据改造，以适应损失函数的输入需求。同时，由于数据集样本数较小，且类内不均衡，使用此损失函数为我们提供了一个数据增广的机会，用于平衡类间数量差距。我们规定，将每一个类别的样例数量均扩增到 1000 条。具体的方法为：对每一类的样例，随机寻找一个与其不同的样例作为正例，再在剩余 19 个类中，任意选择一类中的某一个向量作为反例。按顺序迭代每一个类，如果类内样本已经遍历完全，则返回开头重新进行采样，直到数量达到 1000 条。如此操作，虽然有一定几率出现重复的训练样例，但从整体的概率上而言较低。同时，这有效的解决了样例不平衡的问题。

训练参数：基于嵌入向量的数据维数和网络参数，我们选择使用 RTX2050 Laptop(4G, cuda 11.7) 在 PyTorch 2.0 框架训练三层全连接神经网络。训练参数中，使用 Adam 优化器，批数量设置为 64。我们总共训练了 111 轮，111 轮后损失值收敛，不再出现显著下降的趋势（图7）。

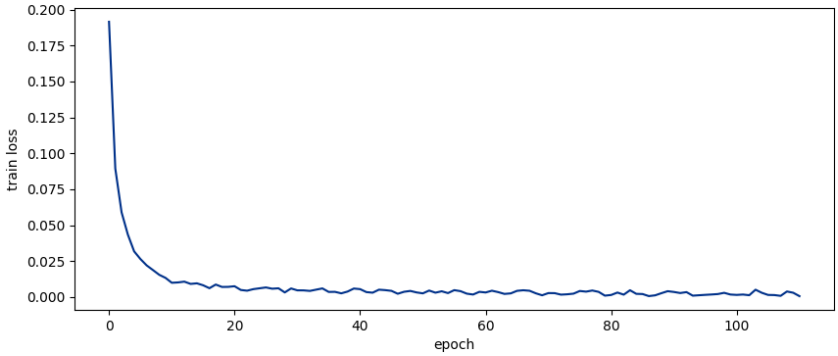

图 7: 编码器训练过程

5 评估

在评估环节，我们可以分为两个部分。首先，第一部分展示我们为何确定使用 VGG 作为视觉信息的嵌入模型。其次，在第二部分，我们以嵌入向量的分类任务为例展开消融实验，用于评估使用 UI 语义嵌入分类向量的有效性。

5.1 CV 预训练模型

以下，对 CV 预训练模型的评估数据、评估标准和结果进行说明。

评估数据：对于 CV 预训练模型，我们使用 Enrico 的图像数据直接输入，并生成了一个随机打散的种子文件，用于确保在训练各模型时样本的稳定。按照 Enrico 的要求，使用 65 : 15 : 20 的比例划分训练、验证和测试集。

评估标准：我们使用 top-k 准确率来评估两个模型的性能。实验中，我们取 $k = 1, 3, 5$ 的情况。在这一指标上，我们主要关注 top-1 准确率，因为这是算法是否具备可用性的关键标准。同时，我们还关注固化模型的大小。如果在相似结果下，模型可以做得更小，则我们认为小模型更符合我们的需求。

结果：表1展现了 VGG 和 ViT 在 Enrico 数据集上的表现。可见，VGG 不仅具备更高的 top-k 准确率，且模型大小更小。鉴于此，我们选择 VGG 模型作为本实验的预训练模型。同时，我们列出 Enrico 论文中的训练结果 Enrico-paper 以及我们的预训练模型性能 Enrico-768。

表 1: 预训练模型性能评估

模型	参数 (MB)	Top-1 acc(%)	Top-3 acc(%)	Top-5 acc(%)
模型选择				
VGG	154	38.4	61.6	73.3
ViT	327	20.5	44.5	62.0
模型预训练				
Enrico-paper	-	75.8	88.4	92.9
Enrico-768	154	79.1	95.9	98.6

5.2 消融实验

我们通过构建 UI 分类作为嵌入向量的下游任务，来评估 UI2Vec 模型的性能。消融实验共 2 组 5 个实验。第一组实验用于对比在 UI2Vec 模型架构的前提下，去掉视觉语义或文字语义的性能表现。第二组实验则直接将编码器模块去掉，用于验证编码器模块的有效性。

5.2.1 数据集

针对第一组实验，我们通过 UI2Vec 模型，分别将包含视觉和文字语义的 768 维向量、仅包含视觉语义的 768 维向量和仅包含文字语义的 768 维向量分别输入在对应数据上进行训练的模型，生成三组 150 维度嵌入模型。需要说明的是，为了保证编码器的结构不变，对于不存在的语义部分，使用全零向量进行填充。

针对第二组实验，我们直接使用视觉语义和文字语义嵌入模块所生成的数据。和第一组实验的数据处理方式相同，对于不存在的语义部分，使用全零向量进行填充。因此，第二组实验使用的嵌入向量是未经编码的 768 维向量。

对两组实验，使用指定随机种子的方式，使用相同的随机过程对数据进行重排，并将数据集的训练集和测试集划分定为 $8:2$。这样可以确保所有实验的训练集和测试机使用的样例相同。

5.2.2 评估指标

我们以 UI 分类任务作为评估 UI2Vec 模型性能的任务。针对这一任务，我们选择 top-k 准确率来评估模型的性能。由于在真实场景之下，我们最为关心的是 k 值较小时的准确率，因此与 CV 预训练模型的评估方法不同，选择 $k = 1, 2, 3$ 作为评估指标。

5.2.3 实验与结果

为了确保实验环境稳定，我们采用了传统机器学习模型 SVM 作为分类器。SVM 可以通过非梯度下降的数值解法对数据超平面进行划分，因此可以有效避免梯度下降的随机性，以提升实验结果的稳定程度。

表2展现了消融实验的结果。

从第一组实验中可以看出，视觉语义仍旧是在 UI 分类任务中的关键信息，将视觉语义去除将导致性能大幅衰减。同时，对于文字语义的融入在 Top-1 准确率上反而导致了准确率的下降，但却在 Top-2、Top-3 准确率上反超了仅包含视觉语义的组别。我们认为，这种现象恰好反映了 UI2Vec 融合视觉语义和文字语义后表现出了更丰富的语义含义，因此虽然损失了少量 Top-1 准确率，却可以使正确的分类标签排名更为靠前。这一点可以从 Top-3 准确率中得到反映。我们认为，这种特质将使得 UI2Vec 模型在其它下游任务（例如 UI 视觉检索）中获得相比仅包含视觉语义的模型更好的性能。

从第二组实验中可以看出，无论保留何种语义信息，在去除编码器后，均会导致性能有不同程度的削弱。这证明了使用 Triplet 损失函数训练的编码器有效的拉开了不同类型之间的间隔，并有效的实现了模态的融合。其中，仅保留视觉语义的组别的性能衰减相对最小。这是由于仅保留

视觉语义不存在跨模态融合的问题，本质上与直接使用预训练视觉模型分类无异。同时，直接拼接多模态信息的操作可能会分类性能弱于仅使用某较强的模态信息的分类结果。

表 2: 消融实验结果

模型	Top-1 acc(%)	Top-2 acc(%)	Top-3 acc(%)
UI2Vec	95.55	**98.97**	**100.00**
使用的嵌入向量			
Visual Only	**96.92**	**98.97**	99.32
Context Only	25.68	41.10	54.79
不使用编码器			
Visual+Context	85.27	94.86	96.58
Visual Only	87.67	94.18	97.26
Context Only	16.44	32.19	44.18

同时，将本实验结果和纯 CV 方法的性能比较如表3。

表 3: 预训练模型性能评估

模型	总参数 (MB)	Top-1 acc(%)
UI2Vec	157	95.6
VGG	154	38.4
ViT	327	20.5

6 讨论

在这一章节，我们将讨论在训练过程中获得的经验、UI2Vec 模型的限制及模型潜在的应用场景。

6.1 训练过程中获得的经验

在 UI2Vec 模型中，我们认为可以总结出两点经验：

对模态融合编码器训练的重要性：在 UI2Vec 模型中，我们在获得了视觉语义和文字语义信息后，进一步对其进行了数据变换，使得嵌入向量在空间中具备更为良好的分布。由实验结果可以表明，这一举措是高度有效的，尤其是对于分类问题的 Top-1 准确率，这一操作能够获得极大的性能提升。

引入文字语义的意义：在我们的实验中，我们发现，虽然引入了文字语义可以一定程度上提升在 Top-3 分类上的准确率，但在某些场景下，其性能甚至不如仅使用单一模态的嵌入向量。我们认为，需要从两个方面来进一步分析这一问题的成因：首先，不同功能的 UI 界面本身在视觉上就具备了足够的差异，引入语义信息可能反而增加了一定的噪声。但这一观点是基于仅做分类问题的视角上出发的。对于同一类 UI 界面，不同的文字信息可以反映其它的信息，例如在设置界面

上的不同层级、不同设置种类等。因此，仍旧有必要在编码器生成嵌入向量时考虑文字语义。同时，这里还存在信息融合比例的问题。UI2Vec 模型中，视觉和文字信息的比例为 768 : 100，这一比例是一种启发式的结论，具体如何设定这一比例，仍是一个开放的话题。

6.2 模型的限制

UI2Vec 模型最大的限制在于训练数据。UI2Vec 模型在全流程中均仅在 Enrico 数据集上开展训练，且评估阶段也使用了来自 Enrico 数据集的数据。这样的操作可以反映出 UI2Vec 的语义嵌入具备有效性，但无法验证对于域外数据的泛化性能。

后续的工作，应当聚焦于引入经过标注的域外数据集对 UI2Vec 模型进行测试。例如，由于 Enrico [5] 是 Rico [4] 数据集的一个很小的子集，我们可以尝试通过把模型应用在 Rico 数据集上验证其有效性。同时，由于 UI 界面设计的风格在不断变迁，也需要考虑引入近年的 UI 设计数据对模型进行验证。

6.3 潜在应用场景

UI2Vec 通过将 UI 图像转化为高维向量，这使得很多仅基于图像难以计算或具有较高计算复杂度的任务变得简单。

例如，我们可以通过向量的相似性计算，以实现高效的 UI 图像检索。UI 设计师可以从海量 UI 设计数据中，找到和当前设计最为相近的 UI 设计进行参考，在其基础上进行改进或者增加自己的创新。VASTA [10] 就构建了类似的系统，我们的模型和他们的不同之处在于我们更加关注 UI 界面本身的文字信息，而 VASTA 仅关注了图像和图标的结构信息。

除此以外，UI2Vec 生成的高度可用的分类结果也使得其可以在人机交互中可及性研究领域或多媒体教程自动生成场景下具备应用前景。例如，根据类别信息，我们可以生成更为精确的图像说明（Image Caption）。我们可以在现有的图像说明生成中，加入类别和内容的约束信息，使得生成的说明更详细、更准确。这样的说明可以被用来为老年人或有视觉障碍的人群生成更为精确的 UI 界面描述，帮助他们能够更好的使用手机。例如，在类似 Synapse [20] 的系统中，我们可以加入使用自然语言生成的 UI 界面描述，进一步增强教程的效果。

7 总结

在本文中，我们提出了 UI2Vec 模型，一种融合 UI 设计文字信息的通用的嵌入模型，能够实现在较低算力的需求下的 UI 图像高精度的分类和搜索。为了更好的描述 UI 的语义信息，我们提出了 UI 语义的定义，融合了视觉信息和文字信息。我们在模型中使用预训练模型生成原始嵌入，使用编码器进行跨模态融合，以此提高生成的嵌入向量的质量。为了验证模型的效果，我们通过设计消融实验的方式，验证了 UI 语义定义的合理性，并说明了基于 Triplet 损失函数的编码器设计的有效性。同时，我们讨论了 UI2Vec 模型搭建过程中总结的经验以及其潜在的应用方向。未来，我们将通过扩大训练数据集的方式，使得模型具备更好的泛化能力。同时，我们会尝试融合自然语言生成技术（NLG），以生成高度精确的 UI 自然语言描述。最终，我们应当基于上述技术

实现，进行系统集成，完成 InstructUI 系统，解决视觉障碍人群或老年人群对于手机界面理解困难的问题。

参考文献

[1] A. S. Ross, "An epidemiology-inspired, large-scale analysis of mobile app accessibility," en, *ACM SIGACCESS Accessibility and Computing*, no. 123, pp. 1–1, Mar. 2020, ISSN: 1558-2337, 1558-1187. DOI: 10.1145/3386402.3386408. [Online]. Available: https://dl.acm.org/doi/10.1145/3386402.3386408 (visited on 05/27/2023).

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[3] X. Zhang, L. de Greef, A. Swearngin, *et al.*, "Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21, New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–15, ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411764.3445186. [Online]. Available: https://dl.acm.org/doi/10.1145/3411764.3445186 (visited on 04/22/2023).

[4] B. Deka, Z. Huang, C. Franzen, *et al.*, "Rico: A Mobile App Dataset for Building Data-Driven Design Applications," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '17, New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 845–854, ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126651. [Online]. Available: https://dl.acm.org/doi/10.1145/3126594.3126651 (visited on 04/22/2023).

[5] L. A. Leiva, A. Hota, and A. Oulasvirta, "Enrico: A Dataset for Topic Modeling of Mobile UI Designs," in *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI '20, New York, NY, USA: Association for Computing Machinery, Feb. 2021, pp. 1–4, ISBN: 978-1-4503-8052-2. DOI: 10.1145/3406324.3410710. [Online]. Available: https://dl.acm.org/doi/10.1145/3406324.3410710 (visited on 04/22/2023).

[6] S. Bunian, K. Li, C. Jemmali, C. Harteveld, Y. Fu, and M. S. Seif El-Nasr, "VINS: Visual Search for Mobile User Interface Design," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21, New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–14, ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411764.3445762. [Online]. Available: https://dl.acm.org/doi/10.1145/3411764.3445762 (visited on 05/20/2023).

[7] G. Li, G. Baechler, M. Tragut, and Y. Li, "Learning to Denoise Raw Mobile UI Layouts for Improving Datasets at Scale," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22, New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 1–13, ISBN: 978-1-4503-9157-3. DOI: 10.1145/3491102.3502042. [Online]. Available: https://dl.acm.org/doi/10.1145/3491102.3502042 (visited on 06/05/2023).

[8] R. Kumar, A. Satyanarayan, C. Torres, *et al.*, "Webzeitgeist: Design mining the web," en, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Paris France: ACM, Apr. 2013, pp. 3083–3092, ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2466420. [Online]. Available: https://dl.acm.org/doi/10.1145/2470654.2466420 (visited on 05/27/2023).

[9] J. Wu, S. Wang, S. Shen, Y.-H. Peng, J. Nichols, and J. P. Bigham, "WebUI: A Dataset for Enhancing Visual UI Understanding with Web Semantics," en, in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Hamburg Germany: ACM, Apr. 2023, pp. 1–14, ISBN: 978-1-4503-9421-5. DOI: 10.1145/3544548.3581158. [Online]. Available: https://dl.acm.org/doi/10.1145/3544548.3581158 (visited on 04/23/2023).

[10] A. R. Sereshkeh, G. Leung, K. Perumal, *et al.*, "VASTA: A vision and language-assisted smartphone task automation system," in *Proceedings of the 25th International Conference on Intelligent User Interfaces*, ser. IUI '20, New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 22–32, ISBN: 978-1-4503-7118-6. DOI: 10.1145/3377325.3377515. [Online]. Available: https://dl.acm.org/doi/10.1145/3377325.3377515 (visited on 05/10/2023).

[11] P. Banerjee, S. Mahajan, K. Arora, C. Baral, and O. Riva, *Lexi: Self-Supervised Learning of the UI Language*, arXiv:2301.10165 [cs], Jan. 2023. DOI: 10.48550/arXiv.2301.10165. [Online]. Available: http://arxiv.org/abs/2301.10165 (visited on 04/23/2023).

[12] C. Lee, S. Kim, D. Han, *et al.*, "GUIComp: A GUI Design Assistant with Real-Time, Multi-Faceted Feedback," en, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–13, ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376327. [Online]. Available: https://dl.acm.org/doi/10.1145/3313831.3376327 (visited on 05/27/2023).

[13] P. P. Liang, Y. Lyu, X. Fan, *et al.*, "MultiBench: Multiscale Benchmarks for Multimodal Representation Learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[14] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv:1409.1556 [cs], Apr. 2015. DOI: 10.48550/arXiv.1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556 (visited on 05/26/2023).

[15] *EasyOCR*, original-date: 2020-03-14T11:46:39Z, May 2023. [Online]. Available: https://github.com/JaidedAI/EasyOCR (visited on 05/26/2023).

[16] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," ser. Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks, Valetta, MT: University of Malta, May 2010, pp. 45–50. [Online]. Available: http://is.muni.cz/publication/884893/en.

[17] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://aclanthology.org/D14-1162 (visited on 05/26/2023).

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929 [cs], Jun. 2021. [Online]. Available: http://arxiv.org/abs/2010.11929 (visited on 05/26/2023).

[19] E. Hoffer and N. Ailon, *Deep metric learning using Triplet network*, arXiv:1412.6622 [cs, stat], Dec. 2018. [Online]. Available: http://arxiv.org/abs/1412.6622 (visited on 05/26/2023).

[20] X. Jin, X. Hu, X. Wei, and M. Fan, "Synapse: Interactive Guidance by Demonstration with Trial-and-Error Support for Older Adults to Use Smartphone Apps," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, no. 3, 121:1–121:24, Sep. 2022. DOI: 10.1145/3550321. [Online]. Available: https://dl.acm.org/doi/10.1145/3550321 (visited on 05/10/2023).

附录

A Enrico 数据集分类情况详情

表 4: Enrico 数据集分类情况及数量统计

类别	数量	描述
空白	76	大量留白区域
电话	6	号码输入界面
相机	8	相机功能
聊天	11	聊天功能
编辑器	18	文本或图像编辑
表单	103	表单填写功能
图库	144	图像网格布局
列表	265	按列组织的内容
登录	141	登录界面
地图	9	地理显示
媒体播放器	32	音乐或视频播放器
菜单	79	项目列表
弹出页面	67	弹出的窗口
新闻	59	新闻片段
其它	52	其它所有内容
个人信息	63	应用“我的”界面
搜索	35	搜索引擎功能
设置	90	手机应用设置
条款	39	服务条款
教程	163	界面上的界面
总计	1460	