arXiv:2601.03928v1 [cs.CV] 7 Jan 2026

FOCUSUI：通过保留位置的视觉令牌选择实现高效用户界面定位

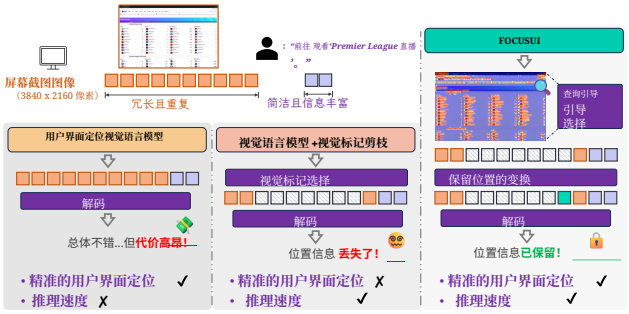
[1]Mingyu Ouyang, [2]Kevin Qinghong Lin, [1]Mike Zheng Shou†, [1]Hwee Tou Ng†新加坡国立大学 [2]牛津大学ouyangmingyu04@u.nus.edu, {kevin.qh.lin, mike.zheng.shou}@gmail.com, dcsnght@nus.edu.sghttps://showlab.github.io/FocusUI

摘要

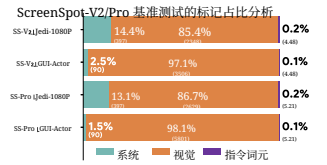
视觉语言模型（*VLMs*）在用户界面（*UI*）定位任务中展现出卓越的性能，这得益于它们处理日益高分辨率屏幕截图的能力。然而，屏幕截图会被切分为数千个视觉令牌（例如，*2K* 分辨率下约为 *4700* 个），带来显著的计算开销并分散注意力。相比之下，人类在与用户界面交互时通常会聚焦于感兴趣区域。在这项工作中，我们首次提出高效用户界面定位任务。通过对任务特征与挑战的实践分析，我们提出了FocusUI，一种高效用户界面定位框架，可选择与指令最相关的图像块，同时保留位置连续性，以实现精确定位。Fo-cusUI 应对两项关键挑战：（***1***）消除视觉编码中的冗余词元。我们通过融合指令条件分数与基于规则的用户界面图分数构建图像块级监督，对较大的同质区域降低权重，从而选出不同且与指令相关的视觉令牌。（***2***）在视觉令牌选择过程中保留位置连续性。我们发现，通用视觉令牌剪枝方法会破坏位置信息，导致其在用户界面定位任务上的准确率严重下降。我们提出了一种新颖的PosPad 策略，将每段连续的被丢弃视觉令牌压缩为一个特殊标记，并将其放置在该序列的最后一个索引处，以保留位置连续性。在四个定位基准测试上进行的全面实验表明，FocusUI优于 *GUI* 专用基线。在 *ScreenSpot-Pro* 基准上，FocusUI-7B 的性能比 *GUI-Actor-7B* 提高了 *3.7%*。此外，即使视觉令牌保留率仅为***30%***，FocusUI-7B的性能也只下降 *3.2%*，同时推理速度最高可提升***1.44***× 倍，峰值 *GPU* 内存降低***17%***。

1. 引言

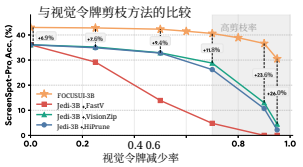
用户界面（UI）视觉定位要求模型根据自然语言在高分辨率屏幕截图中定位目标区域——



(a) **原始用户界面定位视觉语言模型、采用视觉令牌剪枝的视觉语言模型与我们的 FOCUSUI 的比较。**



(b) **研究1：** 用户界面定位任务中，视觉（屏幕截[redacted]元相较于文本（指令）词元所占比例异常高。



(c)**研究2：** 我们提出的位置保持视觉令牌选择与通用视觉令牌剪枝方法的对比。

图1。FOCUSUI 是一种高效用户界面定位框架，可在选择与指令相关的视觉令牌的同时保持位置连续性。**研究1**阐明了处理用户界面定位任务中视觉冗余问题的动机，**研究2**则证明了我们的位置保持选择的有效性。

自然语言指令。现代视觉语言模型（VLMs）在用户界面任务中表现出色，包括导航和定位，这主要得益于其处理高分辨率视觉信息的能力。然而，用户界面截图通常分辨率很高，会被划分为数千个视觉标记，占据了大部分序列预算（图 1b）。视觉标记的这种极端倾斜造成了显著的计算开销。尽管准确率迅速提升，效率问题却鲜少受到关注：na¨为自然图像设计的朴素视觉标记剪枝会破坏多模态序列中的位置连续性，导致精确用户界面定位任务的准确率严重下降。近期关于令牌剪枝策略的研究旨在通过处理视觉标记带来的冗增计算成本来缓解这一问题。通常的做法是

†通讯作者。

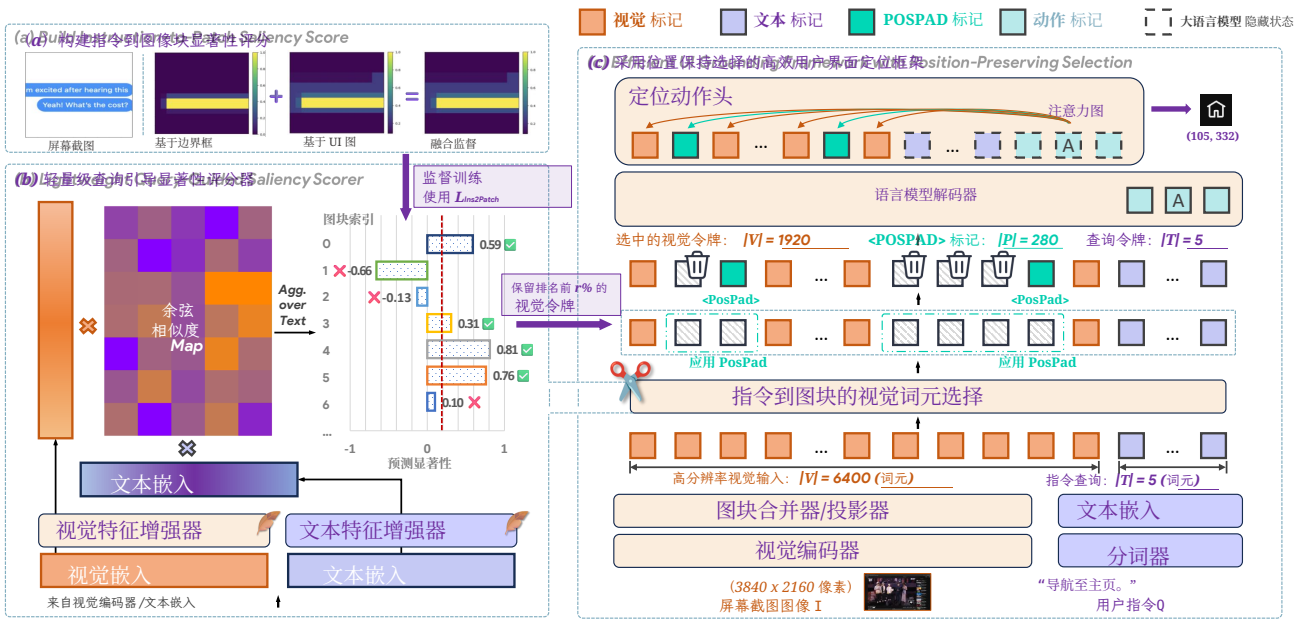


图 2。**我们提出的 FocusUI 概览**。（a）指令到图像块显著性评分构建方式示意图。（b）查询引导显著性评分器与令牌选择。（c）整体用户界面定位框架，展示如何将 POSPAD 应用于丢弃的序列以保持位置连续性。为清晰起见，我们省略了令牌序列中的系统提示。

利用冗余和重要性差异，并在预填充阶段进行筛选，以减少解码过程中的内存和计算成本。然而，直接丢弃视觉标记会造成位置信息丢失，因为这会破坏序列连续性，导致精确用户界面定位的准确率严重下降。

我们提出了 FocusUI，这是一种高效的用户界面定位框架，能够在保持精确定位所需的位置连续性的同时，选择与指令相关的视觉令牌。首先，轻量级的查询引导显著性评分器在密集监督下预测每图像块相关性，该监督融合了指令条件化边界框重叠度信号与基于规则的用户界面图先验，并通过降低大型同质区域的权重来提升效果。其次，我们应用 PosPad，将每个被丢弃的连续序列压缩为一个可学习标记，并将其置于该序列的最后一个索引处，从而保留其位置信息。这种设计缓解了序列碎片化问题，并在激进的保留率下稳定了定位效果。FocusUI 可无缝集成基于 Qwen2.5-VL [3] 和 Qwen3-VL [2] 的不同规模视觉语言模型。在四个基准测试上的实验表明，FOCUSUI 在保持高准确率的同时，显著加快了推理速度并降低了峰值 GPU内存。本文的主要贡献包括：

- **开创高效用户界面定位这一任务**。 我们研究高效用户界面定位的任务特征与挑战，并提出一种专门的方法，在减少视觉令牌的同时保持准确率。
- **通过密集监督实现指令到图像块选择**。 我们将基于规则的*UI* 图结构先验与指令条件化的边界框重叠度相结合，训练轻量级查询引导显著性评分器，以预测每图像块显著性并过滤无关标记。
- **位置保持变换**。 我们提出 POSPAD，在令牌选择过程中保持序列连续性，从而解决通用剪枝方法在精确用户界面定位任务中的失效问题。
- **实际集成与实验结果**。 我们基于不同规模（20亿参数、30亿参数和70亿参数）的 Qwen2.5-VL 和 Qwen3-VL 骨干网络实现了 FOCUSUI。我们的模型优于以往最佳的当前最先进模型，并在四个 UI 定位基准测试上展现出良好的准确率与效率权衡。

2. 高效用户界面定位：任务特征与挑战

我们指出，用户界面定位面临两项关键挑战：**(1)**高分辨率屏幕截图带来的极端标记偏斜与冗余，以及**(2)**由于位置连续性被破坏，朴¨素视觉令牌剪枝导致准确率骤降。本节将对这些挑战进行全面的实证分析，进一步阐明我们提出高效用户界面定位框架的动机。

2.1. 高分辨率视觉理解

UI 落地任务与自然视觉理解的主要区别在于输入特征：**用户界面截图通常具有很高的分辨率**（例如 2560×1440 处于 2K 或 4K 分辨率

3840×2160），**具有结构化的组成**，并且**以大面积同质面板为主，其间穿插着小型控件**。为量化这种偏斜，**研究1**在图1b 中显示，视觉（屏幕截图）令牌占据了两个基准测试和两个落地模型中所有词元的 $\geq 85.4\%$，证实视觉令牌存在严重失衡，并造成了显著的计算开销。

这启发我们采用指令感知选择，优先选择与指令相关的图像块，并降低视觉上重复区域的重要性。我们通过指令到图像块显著性评分（§3.2）来实现这一点，该评分融合了：(i) 与真实边界框的边界框重叠度，以及 (ii) 基于规则的用户界面图先验，该先验会降低大型连通分量的权重，以引导选择。

2.2. 用户界面定位中的位置敏感性

视觉语言模型将多模态输入处理为由视觉图像块令牌和文本令牌交错组成的序列 [26]。具体而言，多模态旋转位置嵌入（M-RoPE） [28] 旨在建模空间和时间关系。实际上，Qwen2-VL 的 M-RoPE 将旋转维度分解为时间、高度和宽度分量，以编码$(t,\ h,\ w)$结构 [14]。然而，我们发现**精确的用户界面定位对视觉嵌入的位置信息十分敏感**，这使得令牌缩减更具挑战性。直接剪枝会造成位置跳跃，出现在 M-RoPE 序列的$(h,\ w)$维度中，导致细粒度目标出现显著的定位偏移。为研究这种敏感性，我们在图**1c 的研究2**中评估了采用先进视觉令牌剪枝方法的用户界面定位模型。准确率大幅下降表明，尽管这些剪枝方法适用于通用视觉理解场景，但在精确定位任务上的性能会显著退化。

我们通过一种 PosPad (§3.3) 策略来解决这一问题：对于每个连续的 被丢弃视觉词元序列，我们用一个可学习标记替换该序列，并将其放置在序列的最后一个索引处，使其继承该索引的$(h,\ w)$ 位置信息。这个特殊标记保留了位置连续性，并减轻了对模型空间理解的干扰。综合来看，**研究1** 阐明了应该移除哪些内容（与指令无关或属于同质区域的部分），而**研究2**则决定了如何进行选择（保留位置，而非天真地丢弃）。这些发现共同构成了我们高效用户界面定位框架的设计动机。

3. FOCUSUI

我们提出 FOCUSUI，这是一种由查询引导的高效用户界面定位框架，可在保持位置连续性的同时，选择与指令相关的视觉令牌。如图2所示，FOCUSUI 由以下几个旨在实现高效用户界面定位的关键组件构成：**(i)** 融合监督每图像块显著性评分，以识别与指令相关的视觉令牌，**(ii)** 用于视觉令牌选择的轻量级查询引导显著性评分器，以及**(iii)** 一种新颖的位置保持型 POSPAD 策略，可在令牌选择过程中保留位置信息。在接下来的章节中，我们将详细介绍各个组件。

3.1. 指令到图像块显著性评分

受 §2 中观察结果的启发，我们首先构建每图像块显著性评分的密集监督，以选出相关视觉标记。我们融合了两个互补部分：(i) 指令条件化边界框重叠度，以及 (ii) 通过并查集实现的用户界面图先验，该先验会降低较大同质区域的权重。

边界框显著性评分。 如算法 1所述，我们将图像划分为一个 $G_h \times G_w$ 图像块网格，图像块大小为 p，并将图像块单元记为 $R_{i,\ j} = [jp, ip, (j+1)p,\ (i+1)p]$。给定一个元素边界框 b_{gt}，每个图像块单元都会根据其与 b_{gt} 的重叠程度获得相应评分。我们将 $S_{\text{bbox}} \in [0,1]$ 设置为归一化重叠面积$(R_{i,\ j} \cap b_{gt})/p^2$，使得完全覆盖的图像块得分为 1

算法1： 构建边界框显著性评分

输入： $I \in [0,1]^{H \times W \times 3}$，图像块大小 p，真实边界框 $b_{gt} = (x_1, y_1, x_2, y_2)$
输出： $S_{\text{bbox}} \in [0,1]^{G_h \times G_w}$
$G_h \leftarrow \lfloor H/p \rfloor$，$G_w \leftarrow \lfloor W/p \rfloor$
对于 $i \leftarrow 0$ **到** $G_h - 1$，**执行**
  对于 $j \leftarrow 0$ **到** $G_w - 1$，**执行**
    $R_{i,\ j} \leftarrow [jp, ip, (j+1)p,\ (i+1)p]$;
    $S_{\text{bbox}}[i,j] \leftarrow$ 区域$(R_{i,\ j} \cap b_{gt})/p^2$
返回 S_{bbox}

算法2： 构建 UI 图显著性分数

输入： $I \in [0,1]^{H \times W \times 3}$，阈值 τ，图像块大小 p
输出： $S_{\text{uig}} \in [0,1]^{G_h \times G_w}$
$G_h \leftarrow \lfloor H/p \rfloor$，$G_w \leftarrow \lfloor W/p \rfloor$
构建图像块像素 $PP_{i,\ j} \in \mathbb{R}^{3 \times p \times p}$用于 $0 \leq i < G_h,\ \ 0 \leq j < G_w$
并查集 对节点进行 $(i,\ j)$ **对于** $i \leftarrow 0$ **到** $G_h - 1$ **执行**
  对于 $j \leftarrow 0$ **到** $G_w - 1$ **执行**
    如果 $j + 1 < G_w$ 且 $\|\text{vec}(PP_{i,\ j}) - \text{vec}(PP_{i,\ j+1})\|_2 < \tau$ **则**
      并查集合并$((i,\ j),\ (i,\ j+1))$
    如果 $i + 1 < G_h$ 且 $\|\text{vec}(PP_{i,\ j}) - \text{vec}(PP_{i+1,\ j})\|_2 < \tau$ **则**
      并查集合并$((i,\ j),\ (i+1,\ j))$
获取连通分量编号 $r_{i,j} \leftarrow$ 并查集查找(i,j)
计数 $n_u \leftarrow \big|\{(i,j):\ r_{i,j} = u\}\big|$ 对于每个唯一根节点 u**分配权重：** $w_u \leftarrow \big(\max\{1, \ln(n_u + 1)\}\big)^{-1}$
设置 $S_{\text{uig}}[i,j] \leftarrow w_{r_{i,j}}$ 对于所有 i,j
返回 S_{uig}

而不相交的图像块得分为 0，从而沿边界框边界呈现从中心向边缘衰减的分布。

UI 图显著性分数。 为进一步抑制背景区域，并加强对未标注区域的监督，我们提出一种基于视觉图块连通分量并查集的 UI 图显著性分数，其灵感源自 *UI* 图 ShowUI 中的先验 [17]。具体而言，我们将 $R_{i,j}$ 中的每个图像块 (i, j) 视为一个节点，并连接 RGB 空间中的距离小于阈值 τ 的四邻域对 ℓ_2。这种并查集会将连通分量归为一组，其大小 n_u 反映了一个区域在视觉上的重复程度。

随后，我们为每个图像块 $w_u = (\max\{1, \ln(n_u+1)\})^{-1}$ 赋予权重，使较大的同质区域（例如，空白背景）获得较低权重。UI 图分数S_{uig} 将每个图像块的分数设为其所属连通分量的权重 w_u。这种设计自然能够抑制背景区域，并增强显著元素的显著性。该分数与指令无关、无需标注，并且可与 S_{bbox} 互为补充，每个图像块对应一个分数。完整流程见算法 2 。

融合监督。 最后，我们融合这两个分数，得到联合监督 $S_{\text{Ins2Patch}}$，即指令到图像块显著性评分：

$$S_{\text{Ins2Patch}} = \lambda\, S_{\text{bbox}} + (1-\lambda)\, S_{\text{uig}} \tag{1}$$

其中 $\lambda \in [0,1]$ 是一个可控权重，在实验中经验性地设为 0。8，适用于所有实验。图 3 展示了这两个组成部分以及最终融合监督的示意图。

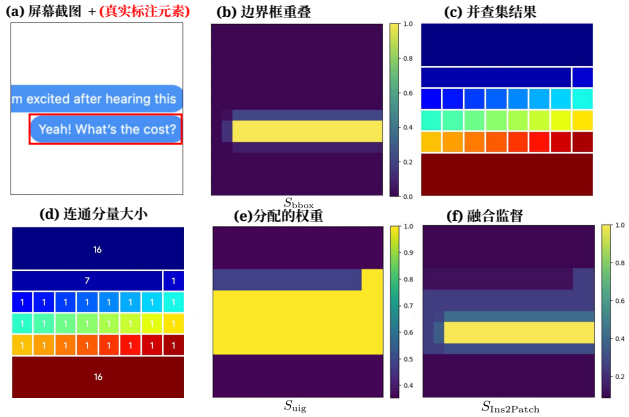


图 3. 构建指令到图像块显著性评分的示例。**(a)** 带有真实标注边界框 b_{gt} 的屏幕截图 I。**(b)** 边界框显著性评分 S_{bbox}。**(c)** 并查集结果。**(d)** 每个连通分量的大小 n_u。**(e)** UI 图显著性分数 S_{uig}。**(f)** 结合**(d)** 和**(e)**得到的融合监督 $S_{\text{Ins2Patch}}$。较亮区域表示正图块，较暗区域表示负图块。

3.2. 轻量级查询引导显著性评分器

利用公式（ $S_{\text{Ins2Patch}}$ ）得到的每图像块监督 1，我们训练了一个轻量级模块——查询引导显著性评分器。该模块根据视觉语言模型骨干网络中图像块与查询文本嵌入之间的相似度，预测每图像块显著性，如图 2 （b）所示。

具体而言，令 $\{v_i\}_{i=1}^{M}$ 表示来自视觉编码器的图块嵌入，$\{e_j\}_{j=1}^{N}$ 表示语言模型（LM）空间中的查询文本嵌入（仅包含与指令对应的部分）。我们使用一个自注意力层增强每种模态的特征，在保留原始嵌入语义的同时强化跨模态交互。对每个特征应用双曲正切约束，随后进行 ℓ_2 归一化，以限制相似度范围。然后，我们通过对图像块和文本嵌入进行矩阵乘法，计算逐词元相似度 $\mathcal{P} \in \mathbb{R}^{M\times N}$。最后，通过均值池化对文本查询维度上的相似度进行聚合，得到每图像块显著性评分 s_i:

$$\mathcal{P} = \tilde{V}\tilde{E}^{\top} \in \mathbb{R}^{M\times N}, \quad s_i = \frac{1}{N}\sum_{j=1}^{N} \mathcal{P}_{i,j}\,. \tag{2}$$

为训练查询引导显著性评分器，我们将分数转换为概率，并优化 KL散度目标。给定式（1）中的融合监督，我们最小化:

$$\mathcal{L}_{\text{Ins2Patch}} = \text{KL}(\text{softmax}(S_{\text{Ins2Patch}}) \,\|\, \text{softmax}(s))\,. \tag{3}$$

3.3. PosPad：位置连续性保持

词元选择策略。 我们首先对式（K 中的预测每图像块显著性评分 $\{s_i\}_{i\in\mathcal{I}}$ 应用 top-2选择。给定保留率 $r \in (0,1]$, , 保留图像块的数量设为 $K = \lfloor rM \rfloor$。令 γ 为排序列表 K中的第 $\{s_i\}_{i\in\mathcal{I}}$个元素。我们构建保留索引集 $\mathcal{K} = \{i \in \mathcal{I} \mid s_i \geq \gamma\}$，并丢弃其余索引 $\mathcal{D} = \{i \in \mathcal{I} \mid s_i < \gamma\}$。

序列变换。 在选出与指令相关的视觉词元后，我们进一步优化序列，以减轻模型空间理解中的位置信息丢失。我们提出 PosPad，这是一种保留位置的序列变换，它将每个连续序列被丢弃的视觉词元替换为一个可学习的特殊词元 PosPad，并将其放置在该序列的最后一个索引处。PosPad 的示意图见图 4。

具体而言，给定原始视觉标记序列$x_{1:M}$、保留索引集 $\mathcal{K}$ 以及上述定义的丢弃索引集 $\mathcal{D}$，我们按照一维展平序列顺序，将 $\mathcal{D}$ 划分为连续序列（即，最长连续序列） $\mathcal{R}_1, \ldots, \mathcal{R}_U$。对于每个序列 $\mathcal{R}_u$，我们只保留其最后一个索引 $r_u^{\text{end}} = \max\ \mathcal{R}_u$，并移除其余索引。令 $\mathcal{E}_{\text{seq-end}} = \{r_u^{\text{end}}\}_{u=1}^{U}$ 表示序列末端索引集，并定义保留索引集 $\mathcal{S} = \mathcal{K} \cup \mathcal{E}_{\text{seq-end}}$。

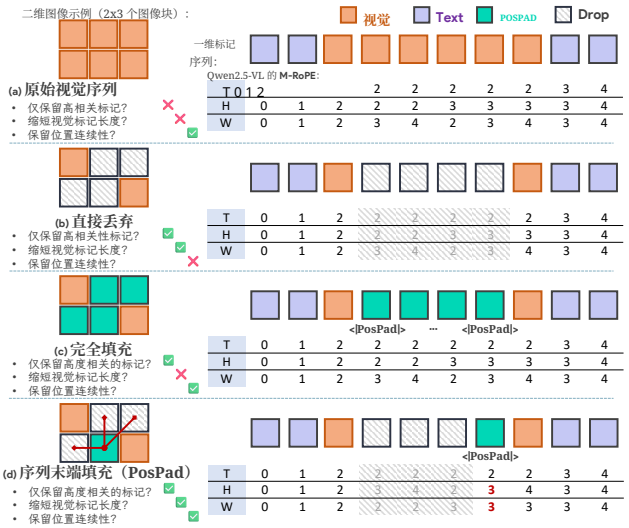


图 4. POSPAD 序列变换示意图：通过一个二维图像（2×3 图像块）及其一维序列示例，展示如何保持位置连续性。正如策略 (d) 所示，可学习的 <位置填充> 标记被放置在每个被丢弃的视觉词元连续序列的末尾索引处。

接着，我们用单个标记 <pos pad>替换每个连续序列，并保持所有其他词元不变：

$$x'_j = \begin{cases} \texttt{<pos_pad>} & \text{if } j \in \mathcal{E}_{\text{seq-end}}, \\ x_j & \text{if } j \in \mathcal{K}, \end{cases} \tag{4}$$
$$\textsc{PosPad}(x_{1:M}) = \{x'_j\}_{j\in\mathcal{S}}.$$

因此，视觉词元的最终输出长度为 $M' = M - (|\mathcal{D}| - U)$，其中 <位置填充> 词元的总数为 U。每个被丢弃的序列 $\mathcal{R}_u$ 都会使序列缩短 $|\mathcal{R}_u| - 1$，同时保持序列末尾的位置连续性。表 M、M' 和 U 研究了不同保留率下的具体示例。7c。

与直接丢弃相比，POSPAD 保持了位置连续性，并且在实证中稳定了模型的空间理解能力。§4.2.5 还研究了其他策略。由于 POSPAD 只改变序列稀疏度，不改变词元索引或旋转基，因此它兼容常见的 M-RoPE 实现，且无需修改下游语言模型架构。

3.4. 高效用户界面定位框架

与视觉语言模型的集成。 我们将视觉令牌选择策略集成到现有视觉语言模型中，在视觉图像块嵌入输入语言模型解码器之前进行处理。具体而言，查询引导显著性评分器接收图像块特征 $\{v_i\}_{i=1}^{M}$ 和指令词元嵌入 e_j，通过公式（2）计算得分 s_i，并根据给定的保留率 r选择排名前-K 的索引 $\mathcal{K}$。随后，我们使用 POSPAD 优化序列，得到长度为 $M' \ll M$ 的紧凑视觉序列，同时保持位置连续性。语言模型解码器处理此序列，且不改变其原始架构。我们将该框架应用于 Qwen2.5-VL 和 Qwen3-VL 模型。对于采用 DeepStack [20] 视觉编码器的 Qwen3-VL 模型，仅为保留的图像标记 $\mathcal{K}$收集深层视觉嵌入。

使用选定图像块进行无坐标用户界面定位。 我们发现，GUI-Actor [30] 中的无坐标用户界面定位方案与我们的选择方式最为兼容：模型通过在语言模型解码器之上额外添加一个动作头，直接在图块嵌入上定位元素；而我们的视觉令牌选择则通过丢弃与指令无关的区域来减少候选项。具体而言，解码器语言模型会输出一个动作词元序列：

$$\begin{aligned}\text{LM}(I, q) = \{&x_{1:i-1}, \texttt{<ACTOR_START>}, \\ &\texttt{<ACTOR>}, \texttt{<ACTOR_END>}, x_{i+3:N}\}.\end{aligned} \tag{5}$$

随后，动作头将 $h_{\texttt{ACTOR}}$ 与视觉图块对齐，从而生成图块上的注意力图。我们首先使用一个自注意力层对选定的图像块特征 $\{\tilde{v}_i\}_{i=1}^{M'}$ 进行上下文细化：$\tilde{v}_1, \ldots, \tilde{v}_{M'} =$ 自注意力$(v_1, \ldots, v_{M'})$。然后，我们分别使用 MLP_T 和 MLP_V 对 $h_{\texttt{ACTOR}}$ 和每个 $\tilde{v}_i$ 进行投影，并计算注意力分数：

$$\begin{aligned} &z = \text{MLP}_T(h_{\texttt{ACTOR}}), \quad z_i = \text{MLP}_V(\tilde{v}_i), \\ &\alpha_i = \tfrac{z^\top z_i}{\sqrt{d}}, \quad a_i = \text{softmax}(\alpha)_i. \end{aligned} \tag{6}$$

分布 a_i 标识出执行该动作最相关的区域。借助选定的视觉词元，这类动作头受益于更少的视觉候选项，以及与指令更相关的保留图像块。

训练目标。 查询引导的显著性评分器通过下游大语言模型目标——下一词元预测损失 $\mathcal{L}_{\text{NTP}}$ 和用于定位的动作注意力损失$\mathcal{L}_{\text{Attn}}$ ——进行端到端训练：

$$\mathcal{L}_{\text{Attn}} = \sum_{i=1}^{M'} p_i \log \frac{p_i}{a_i}, \; p_i = \frac{y_i}{\sum_{j=1}^{M'} y_j + \epsilon}, \; i = 1, \ldots, M' \tag{7}$$

其中，y_i 表示第 i 个图像块的注意力分数标签（若其与真实标注边界框重叠，则为 1，否则为 0），ϵ 是用于数值稳定性的小常数。整体训练目标为：

$$\mathcal{L} = \mathcal{L}_{\text{Ins2Patch}} + \mathcal{L}_{\text{NTP}} + \mathcal{L}_{\text{Attn}}. \tag{8}$$

4. 实验

4.1. 实验设置

实现细节 我们采用当前最先进的视觉语言模型 Qwen2.5-VL [3] 和 Qwen3-VL [2] 作为基础模型，并使用不同规模的模型来展示我们方法的通用性。我们进行监督微调，以

模型	ScreenSpot-V2							ScreenSpot-Pro								
	Mob.-T	Mob.-I	Des.-T	Des.-I	网页-T	网页-I	平均	开发	创意	CAD	科学	办公	操作系统	平均值-T	平均值-I	平
Operator [22]	47.3	41.5	90.2	80.3	92.8	84.3	70.5	35.1	39.6	16.1	43.7	53.0	32.7	45.0	23.0	3
OS-Atlas-7B [31]	95.2	75.8	90.7	63.6	90.6	77.3	84.1	17.7	17.9	10.3	24.4	27.4	16.8	28.1	4.0	18.9
Aguvis-7B [35]	95.5	77.3	95.4	77.9	91.0	72.4	86.0	16.1	21.4	13.8	34.6	34.3	19.4	-	-	22.9
Tong-UI-7B [38]	93.1	81.5	96.4	82.9	90.2	84.7	88.7	22.7	21.1	15.3	34.3	38.3	18.4	35.1	8.0	25.7
UGround-V1-7B [13]	95.0	83.3	95.0	77.8	92.1	77.2	87.6	28.1	31.7	14.6	39.0	49.6	24.5	-	-	31.1
UI-TARS-7B [23]	96.9	89.1	95.4	85.0	93.6	85.2	91.6	36.1	32.8	18.0	50.0	53.5	24.5	47.8	16.2	3
UI-TARS-72B [23]	94.8	86.3	91.2	87.9	91.5	87.7	90.3	40.8	39.6	17.2	45.7	54.8	30.1	50.9	17.5	3
UI-TARS-1.5-7B [23]	-	-	-	-	-	-	90.0	31.8	40.2	31.8	47.2	65.6	33.2	-	-	42.6
Qwen2.5-VL-3B [3]	93.4	73.5	88.1	58.6	88.0	71.4	80.9	21.4	25.8	18.4	29.5	40.9	20.4	37.8	6.6	25.9
Qwen2.5-VL-7B [3]	97.6	87.2	90.2	74.2	93.2	81.3	88.8	29.1	24.9	13.8	31.1	45.7	22.4	39.9	7.6	27.6
Qwen2.5-VL-32B [3]	97.9	88.2	98.5	79.3	91.2	86.2	91.3	48.5	41.1	32.6	57.1	67.4	42.3	63.2	22.5	4
GUI-Actor-3B [30]	97.6	83.4	96.9	83.6	94.0	85.7	91.0	39.8	36.7	34.1	49.6	61.3	35.2	-	-	42.2
GUI-Actor-7B [30]	97.6	88.2	96.9	85.7	93.2	86.7	92.1	38.1	41.4	38.3	50.8	63.0	38.8	-	-	44.6
Jedi-3B [33]	96.6	81.5	96.9	78.6	88.5	83.7	88.6	38.1	34.6	23	38.6	57.0	25.0	49.8	13.7	3
Jedi-7B [33]	96.9	87.2	95.9	87.9	94.4	84.2	91.7	27.4	34	32.2	52.4	68.7	26.0	52.6	18.2	3
FOCUSUI-3B（$r=100\%$）	99.2	85.9	96.1	87.3	95.4	81.9	91.5	43.1						59.3	18.9	4
FOCUSUI-3B（$r=50\%$）	98.8	86.9	95.0	87.3	95.4	81.9	91.4	42.1							19.0	4
FOCUSUI-30亿参数（$r=30\%$）	98.5	85.3	96.1	87.3	94.3	81.9	91.0	38.1							17.4	4
FOCUSUI-70亿参数（$r=100\%$）	98.8	91.6	95.6	92.1	95.0	84.4	93.1	44.5						64.7	21.9	4
FOCUSUI-7B（$r=50\%$）	98.8	92.2	93.9	87.3	95.0	85.2	9	42.8							21.9	4
FOCUSUI-7B（$r=30\%$）	98.8	90.1	93.3	85.7	93.9	85.2	91.8	38.8							20.4	4

表 1. 在 *ScreenSpot-V2* [31] 和*ScreenSpot-Pro* [15]上的性能比较。

模型	文本	元素	布局	操作	拒答	平均值
Gemini-2.5-Pro [12]	59.8	45.5	49.0	33.6	38.9	45.2
Operator [22]	51.3	42.4	46.6	31.5	0.0	40.6
UGround-V1-7B [13]	51.3	40.3	43.5	24.8	0.0	36.4
Aguvis-7B [35]	55.9	41.2	43.9	28.2	0.0	38.7
UI-TARS-7B [23]	60.2	51.8	54.9	35.6	0.0	47.5
UI-TARS-1.5-7B [23]	70.1	57.9	59.7	51.7	0.0	56.0
Qwen2.5-VL-3B [3]	41.4	28.8	34.8	13.4	0.0	27.3
Qwen2.5-VL-7B [3]	45.6	32.7	41.9	18.1	0.0	31.4
GUI-Actor-3B [30]	60.5	56.1	58.5	32.2	0.0	50.5
GUI-Actor-7B [30]	60.2	54.2	58.1	30.9	0.0	49.5
Jedi-3B [33]	67.4	53.0	53.8	44.3	7.4	50.9
Jedi-7B [33]	65.9	55.5	57.7	46.9	7.4	54.1
FOCUSUI-3B ($r=100\%$) (			59.7	37.6	0.0	53.4
FOCUSUI-3B ($r=50\%$) 6			63.6	37.6	0.0	54.6
FOCUSUI-3B ($r=30\%$) 6			62.9	33.6	0.0	51.8
FOCUSUI-7B ($r=100\%$) (			63.6	34.9	0.0	54.4
FOCUSUI-7B ($r=50\%$) 6			63.6	31.5	0.0	54.1
FOCUSUI-7B ($r=30\%$) 6			64.4	31.5	0.0	53.9

表 2. OSWorld-G [33] 上的性能比较。

模型	基础	功能空间		Avg
Claude-3.7-Sonnet [1]	9.48	7.73	7.60	8.27
ShowUI-2B [17]	8.07	7.67	2.07	5.94
OSAtlas-7B [31]	12.2	11.2	3.67	9.02
UGround-7B [13]	11.5	12.2	2.79	8.83
UGround-V1-7B [13]	15.4	17.1	6.25	12.9
Aguvis-7B [35]	17.8	18.3	5.06	13.7
UI-TARS-7B [23]	20.1	24.3	8.37	17.6
UI-TARS-72B [23]	31.4	30.5	14.7	25.5
GUI-Actor-3B [30]	27.4	24.6	7.0	19.3
GUI-Actor-7B [30]	30.1	28.1	7.8	21.6
Jedi-3B [33]	22.3	25.2	9.35	18.7
Jedi-7B [33]	32.3	30.5	12.8	24.8
FOCUSUI-3B（$r=100\%$）	30.0	26.9	8.7	21.5
FOCUSUI-3B（$r=50\%$）	29.7	26.0	8.2	20.9
FOCUSUI-3B（$r=30\%$）	29.1	26.4	7.6	20.6
FOCUSUI-7B（$r=100\%$）	33.6	31.2	11.2	24.9
FOCUSUI-7B（$r=50\%$）	32.5	31.0	11.3	24.5
FOCUSUI-7B（$r=30\%$）	32.3	29.2	11.0	23.8

表 3. UI-Vision [21] 上的性能比较。

获得以下变体：基于 Qwen2.5-VL 的 FOCUSUI-3B 和 FOCUSUI- 7B，以及基于 Qwen3-VL 的 FOCUSUI-QWEN3-VL-2B。

为确保公平比较，我们将训练预算与基线方法 GUI-Actor [30], 对齐，使用从多个公开用户界面数据集中收集的约 1M 张屏幕截图。为确保标注质量，我们遵循 V2P [6]，使用 O mni- Parser [19] 筛选真实标注与检测框之间交并比低于 0.3 的样本

。训练期间，视觉令牌保留率 r 从 (0.1,1.0) 中均匀采样。所有模型均使用 DeepSpeed [25]Zero-2，在8× NVIDIA H200 GPU 上训练 1 个 epoch。更多训练细节见附录。

评测基准我们在四个用户界面定位基准测试上进行了实验，包括 ScreenSpot-V2 [31],

模型	ScreenSpot-V2			ScreenSpot-Pro		
	平均值-T	平均值-I		平均值-T	平均值-I	
Qwen3-VL-2B† [3]	94.7	78.9	87.8	52.8	16.7	39.0
FOCUSUI-QWEN3-VL-2B（$r = 100\%$）	95.8	85.6	9	51.5	20.9	3
FOCUSUI-QWEN3-VL-2B（$r = 50\%$）	95.7	85.0	9	52.5	20.9	4
FOCUSUI-QWEN3-VL-2B（$r = 30\%$）	93.5	84.3	8	49.7	20.2	3

表 4. 基于**Qwen3- VL** 骨干网络的模型性能比较。 † 表示我们在 HuggingFace 上对官方模型进行自行评估所得的结果 [29]。

ScreenSpot-Pro [15], OS-World-G [33], 和 UI-Vision [21]。其中，ScreenSpot-Pro 提供了更高分辨率的界面，可模拟多源真实世界应用，是评估效率和精确用户界面定位特性的实用测试平台。

4.2. 主要结果

我们围绕以下五个研究问题（RQ）组织主要结果：

- **RQ1（§4.2.1 性能）**： FocusUI 能否在保持准确率的同时有效减少视觉令牌？
- **RQ2（§4.2.2 与通用剪枝方法的比较）**： FocusUI 与通用视觉标记剪枝方法相比表现如何？
- **RQ3（§4.2.3效率分析）**： FocusUI 在不同设置下能实现哪些效率提升？
- **RQ4（§4.2.4 定性结果）**： FocusUI如何选择与指令相关的视觉令牌？
- **RQ5（§4.2.5消融研究）**： 各组件和保留设置会如何影响性能？

4.2.1. RQ1：性能

表1、2 和3分别报告了 ScreenSpot-V2 [31] & ScreenSpot-Pro [15], OS-World-G [33], 和 UI-Vision [21], 上的定位性能。我们测试了一系列保留率 $r \in \{100\%、50\%、30\%\}$，以刻画性能退化曲线，并与使用全部视觉令牌的密集基线进行比较。在所有四个基准测试中，即使在 $30-50\%$ 令牌保留率下，FOCUSUI 的表现仍优于同等规模的 GUI 专用基线，实现了最先进的定位性能。此外，我们还在表 OCUSUI-QWEN3-VL-2B 基于更新的最先进 Qwen3-VL [2]骨干网络的性能。 4。更详细的结果细分和保留率结果见附录。

4.2.2. RQ2：与通用剪枝方法的比较

表5 展示了与其他通用视觉语言模型视觉令牌剪枝方法的比较。具体而言，我们将 FOCUSUI 与 Fast-V [7],HiPrune [18],和 Vision-Zip [36]进行比较。在词元保留率为 30% 时，我们的 FOCUSUI 仍能保持接近基线的准确率（在 ScreenSpot-V2/Pro/OS-World-G 上分别仅低 0.5/3.2/2.5 个百分点），而通用剪枝方法会严重降低性能。值得注意的是，我们的方法原生兼容

模型 + 剪枝方法（发表场所）	%Ret. 比例	SS-V2 Avg	SS-Pro Avg	OSWorld-G Avg
Qwen2.5-VL-3B	100%	81.5	26.1	27.3
+ Fast-V（ECCV'24） [7]	30%	38.6 (-52.7%)	4.8 (-81.6%)	14.4 (-47.4%)
+ HiPrune（arXiv'25） [18]	30%	72.0 (-11.7%)	18.0 (-30.8%)	20.4 (-25.3%)
+ Vision-Zip（CVPR'25） [36]	30%	75.4 (-7.5%)	18.9 (-27.4%)	23.0 (-15.6%)
Jedi-3B	100%	88.9	36.1	48.8
+ Fast-V（ECCV'24） [7]	30%	51.0（-42.6%）	14.1（-60.9%）	23.9 (-51.0%)
+ HiPrune (arXiv'25) [18]	30%	80.9（-9.0%）	26.2 (-27.3%)	40.4（-17.1%）
+ Vision-Zip（CVPR'25） [36]	30%	82.8（-6.9%）	28.8 (-20.3%)	41.5（-14.9%）
FOCUSUI-3B	100%	91.5	43.8	53.4
+ 带 POSPAD 的显著性评分器	30%	91.0（-0.5%）	40.6 (-7.3%)	51.8（-3.0%）

表 5. 与通用视觉令牌剪枝方法的比较。

模型	%保留率 比例	每个样本的视觉标记数 标记时间（秒）	最大 GPU 内存（MB）	SS-Pro Acc
基础模型：*Qwen2.5-VL*，$max_pixel = 6400 * 28 * 28 = 4816000$				
FOCUSUI-7B	100%	5319 1.75 (1.00×)	20994 (1.00×)	48.3
FOCUSUI-7B	70%	3989 1.67 (1.05×)	18334 (0.87×)	47.7
FOCUSUI-7B	50%	2659 1.49 (1.18×)	17944 (0.85×)	46.5
FOCUSUI-7B	30%	1329 1.22 (1.44×)	17392 (0.83×)	45.1
基础模型：*Qwen3-VL*，$max_pixel = 6000 * 32 * 32 = 6144000$				
FOCUSUI-QWEN3-VL-2B	100%	4627 0.97 (1.00×)	6278 (1.00×)	39.8
FOCUSUI-QWEN3-VL-2B	70%	3470 0.90 (1.08×)	6142 (0.98×)	40.1
FOCUSUI-QWEN3-VL-2B	50%	2313 0.85 (1.14×)	5680 (0.91×)	40.4
FOCUSUI-QWEN3-VL-2B	30%	1156 0.71 (1.37×)	5170 (0.82×)	38.5

表6. 在不同保留率和 FOCUSUI 模型骨干网络下，ScreenSpot-Pro 基准上的效率分析。 * 不包含 <pos pad> 词元数量。

FlashAttention [10]，因为它不需要任何中间注意力或激活信息。

4.2.3. RQ3：效率分析

在表6中，我们使用不同的 Qwen2.5-VL 和 Qwen3-VL 骨干网络评估 FOCUSUI，以研究其效率提升及准确率与效率权衡。结果表明，将保留率从 100% 降至 30%，推理速度最高可提升 1.44× 倍，峰值内存降低约 17–18%，而准确率仅损失 3.2 个百分点。

4.2.4. RQ4：定性结果

图5 展示了 FOCUSUI 的定性示例。预测的热力图表明，该模型能够有效地为指令选择相关视觉令牌，同时抑制背景区域。

4.2.5. RQ5：消融研究

我们在表中展示了所提出组件的有效性。 7表 7a 和 7b中的实验评估模型均以 Qwen2.5-VL-3B 为基础模型，并使用30% 的完整数据进行训练。

视觉令牌选择。 我们与图中所示的变体进行比较。4：(a) 原始视觉序列。(b) 直接丢弃。(c) 完全填充 通过在每个被丢弃的位置插入 <位置填充> 来保持连续性。我们还测试了将零样本CLIP [24] 用作评分策略。这些变体的性能见表7a。

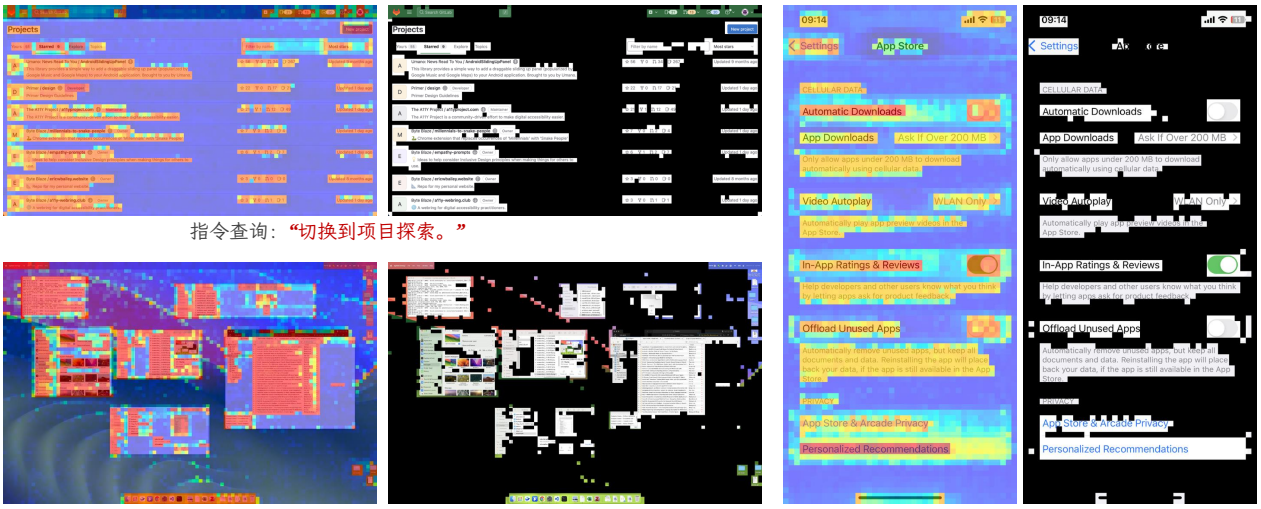


图5. 预测显著性热图和保留图像块的定性可视化，保留**比例** $r = 30\%$。黑色区域表示在解码过程中未被大语言模型使用的被丢弃视觉词元。示例取自 ScreenSpot-V2 和 ScreenSpot-Pro 基准测试，涵盖网页、桌面和移动端界面。

评分变体	保留策略	%Ret. 比例	减少标记长度?	保留位置连续性?	SS-Pro Acc
基线	(a) 不适用	100%	✗	✓	40.9
CLIP 分数 [24]	(b) 直接丢弃	50%	✓	✗	28.5
	(c) 完全填充	50%	✗	✓	38.7
	(d) POSPAD	50%	✓	✓	38.2
Ins2Patch 分数	(b) 直接丢弃	50%	✓	✗	29.2
	(c) 完全填充	50%	✗	✓	42.1
	(d) POSPAD	50%	✓	✓	**42.3**

(a) 不同的视觉令牌选择方法和位置连续性保持策略。

变体	SS-Pro准确率
仅使用 UI-Graph标注	41.1
仅使用基于边界框的标注	39.8
完整 FOCUSUI	**42.3**

(b) 减少率为 50% 时的 Ins2Patch 分数消融。

%Ret. 比例	#Vis. 词元	#POSPAD 词元	#总计	SS-Pro Acc
100%	6019	0	6140	43.8
75%	4514	435	5070	43.3
50%	3009	433	3563	42.3
25%	1504	315	1941	40.6
10%	601	193	915	36.6

(c) 不同的保留率和词元数量。

表 7. FOCUSUI 关键组件的消融实验。

指令到图块显著性分数监督。 表中的结果 7b 表明，移除用户界面图先验或边界框重叠分数，都会使准确率低于 FOCUSUI 的融合监督。

保留率。 表7c 表明，我们的 FOCUSUI 在准确率与保留率之间实现了平滑的权衡：100% 与密集基线相当，50% 仍能保留大部分性能，而进一步采取更激进的设置则会导致准确率大幅下降。

5. 相关工作

视觉语言模型驱动的图形用户界面智能体 视觉语言模型的最新进展加速了图形用户界面智能体的发展，使其能够感知、规划并在图形界面中采取行动 [1, 3, 12, 22, 28]。它从文本依赖型图形用户界面智能体 [11, 42], 逐步过渡到纯视觉解决方案，以进行任务规划、元素定位和界面控制 [9, 17, 23, 33]，充分发挥视觉语言模型的能力。

用户界面视觉定位 给定一张屏幕截图和一条自然语言指令，用户界面定位会在屏幕上定位用于交互的目标区域。随着模型设计的改进 [6, 9, 17, 23, 27, 30, 31, 37, 38] 和数据扩展 [8, 13, 33, 35], ，近年来用户界面定位的性能迅速提升。

视觉令牌削减 与信息密集的文本相比，视觉令牌通常存在大量冗余，因此需要削减令牌以降低计算成本 [4, 7,34, 40]。近期研究进一步探索了基于词元重要性和冗余的免训练剪枝 [18, 41]，或采用编码器侧压缩 [36, 39]。

6. 结论

本文介绍了 FOCUSUI，这是一种由查询引导的高效用户界面定位框架，可在保持位置连续性的同时选择与指令相关的 视觉令牌。将 FOCUSUI 与最先进的视觉语言模型集成后，它在四个用户界面定位基准测试中实现了出色的准确率与效率权衡。

局限性与未来工作。 FOCUSUI 主要通过空间视觉令牌缩减提升效率。未来工作可以考虑时间维度，因为用户界面交互通常涉及多轮顺序动作。

A. 实现细节

A.1. 训练数据

原始数据集 我们的训练集汇集了若干高质量的公开图形用户界面数据集，并遵循 GUI-Actor。为确保评估公平，我们排除了 Wave-UI 中与下游任务测试集重叠的样本。

优化标注质量 我们使用 OmniParser V2 [19] 过滤真实标注与 OmniParser 检测框之间交并比低于 0.3 的样本。这使元素数量减少了 22.9%。最终的训练统计数据见表 8。

数据集	#屏幕截图	#元素	平台
UGround [13]	775K	8M	Web
GUI-Env [30]	70K	262K	Web
GUI-Act [30]	13K	42K	Web
AndroidControl [16]	47K	47K	Android
AMEX [5]	100K	1.2M	Android
Wave-UI	7K	50K	混合
总计（原始数据集）	1012K	9.6M	–
总计（过滤后）	**976K**	**7.4M**	–

表8. FOCUSUI 使用的训练数据集统计信息。

A.2. 训练细节

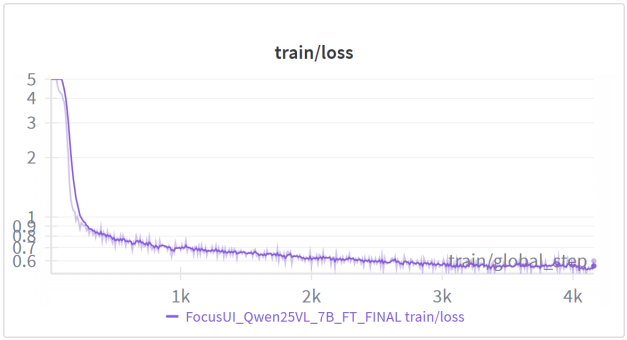
我们在 8× NVIDIA H200 GPU 上使用 bfloat16 精度、DeepSpeed ZeRO-2 [25]，和 FlashAttention-2 [10] 训练 FOCUSUI。每个 GPU 的有效批量大小设为 32（梯度累积为 4），最大像素数设为 5720064，与 GUI-Actor 保持一致。训练分为两个阶段：

阶段 1：显著性评分器预训练。 我们使用学习率 $1e-4$ 对随机初始化的查询引导显著性评分器进行 1 个 epoch 的预训练。对于 30亿参数和 70亿参数模型，这一过程均耗时约 12 小时。

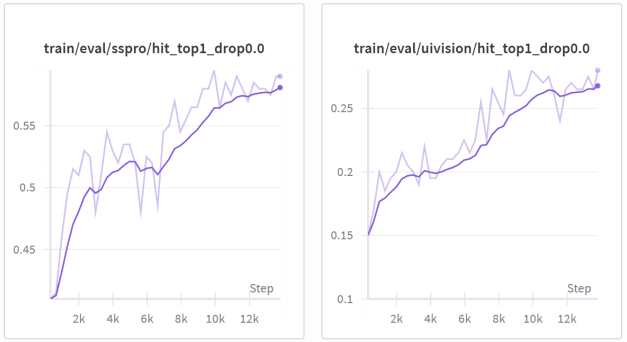
阶段 2：全模型微调。 我们使用学习率 $5e-6$ 对所有参数进行 1 个 epoch 的微调。30亿参数模型约需 36 小时，70亿参数模型约需 48 小时。

超参数详情。 训练期间，我们使用 $\tau = 2$ 和 $\lambda = 0.8$构建逐图像块显著性分数监督。对于基于 Qwen2.5-VL 和 Qwen3-VL 的模型，图像块大小 p 分别设为 14 和 16。每个训练样本的视觉令牌保留率 r 均匀采样自 (0.1, 1.0)。

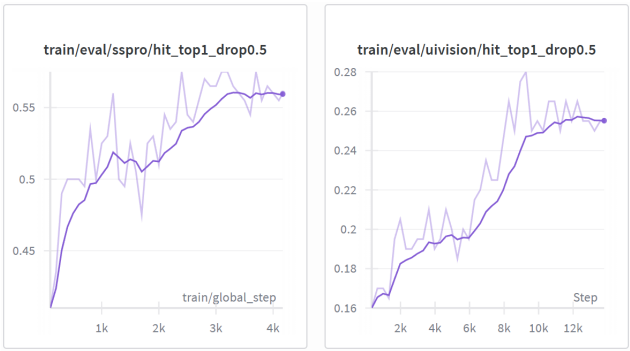
为确保可复现性，我们对所有获得的 FocusUI 模型均使用最终检查点。我们还提供了所有训练模型的完整 Weights & Biases (WandB) 日志。FocusUI-7B 训练过程中的训练损失和评估曲线如图所示。6。



(a)**训练期间的总损失** 曲线。



(b)**评估：** ScreenSpot-Pro 和 UI-Vision 在**保留率** = **100** % 下的表现。



(c)**评估：** ScreenSpot-Pro 和 UI-Vision，保留率为%。

图 6。FOCUSUI-7B 的 WandB 损失和评估结果。

A.3. UI 定位基准测试

我们在四个公开基准测试上进行评估，这些基准测试包含与指令配对的屏幕截图：**ScreenSpot-V2** [31]，**ScreenSpot-Pro** [15]，**OS-World-G** [33]，和 **UI-Vision** [21]。这些基准测试的统计数据见表 9。

ScreenSpot-V2 [31]。 ScreenSpot 的改进版本 [9]，包含来自移动端、桌面和网页环境的 1,272 个样本。

ScreenSpot-Pro [15]。 该基准包含来自 23 款专业应用程序的 1,581 个样本，面向高分辨率界面和复杂布局，用于测试泛化能力。

OS-World-G [33]。从 OSWorld 抽样 [32], 该基准包含 564 个样本，并按任务类型分类（文本匹配、元素识别、布局理解、精细操作和拒绝）。

UI-Vision [21]。这是一个以桌面为中心的基准测试，包含来自 83 个应用程序的 5,790 个样本，用于评估元素定位、布局定位和动作预测。

基准	#样本数	平均分辨率	最大分辨率	平台
ScreenSpot-V2	1272	1725×1657	2880×1800	混合
ScreenSpot-Pro	1581	3267×1727	6016×3384	桌面
OS-World-G	564	1696×955	1920×1080	桌面
UI-Vision	5790	1851×1034	3360×2036	桌面

表9 。本文所用评测基准概览 工作。

B. 讨论

B.1. 视觉冗余分析

表。10给出了**研究1**的词元统计（如主论文中的图1(b)所示）。在 ScreenSpot-Pro 评估中采用默认模型设置后，我们发现，在所研究的基准测试中，视觉令牌至少占序列的 84.3%，证实了用户界面定位任务中存在显著的视觉冗余。

基准	模型	#系统。词元	#视觉。词元	#指令 词元	视觉词元 %
ScreenSpot-V2	Jedi-1080p	397	2348	4.5	85.4%
	GUI-Actor	90	3506	4.5	97.1%
ScreenSpot-Pro	Jedi-1080p	397	2629	5.2	86.7%
	GUI-Actor	90	5801	5.2	98.1%
OS-World-G	Jedi-1080p	397	2244	21.3	84.3%
	GUI-Actor	90	2244	21.3	95.3%
UI-Vision	Jedi-1080p	397	2249	9.9	84.7%
	GUI-Actor	90	2566	9.9	96.3%

表10. 正文图1(b)中所示研究1的令牌统计。

B.2. 位置敏感性分析

表11 展示了**研究2** 的详细结果（见论文正文中的图1(c)），比较了 FOCUSUI 与集成了先进视觉令牌剪枝方法的用户界面定位模型。

%Ret. 比例	模型 + 剪枝方法	SS-V2 Avg	SS-Pro Avg	OSW-G Avg
100%	Qwen2.5-VL-3B	81.5	26.1	27.3
50%	+ Fast-V	43.5	13.9	14.3
	+ HiPrune	80.4	20.3	26.2
	+ Vision-Zip	81.0	21.0	27.1
30%	+ Fast-V	38.6	4.8	14.4
	+ HiPrune	72.0	18.0	20.4
	+ Vision-Zip	75.4	18.9	23.0
100%	Jedi-3B	88.9	36.1	48.8
50%	+ Fast-V	50.3	20.4	25.3
	+ HiPrune	88.3	32.8	46.4
	+ Vision-Zip	88.1	32.9	46.6
30%	+ Fast-V	51.0	14.1	23.9
	+ HiPrune	80.9	26.2	40.4
	+ Vision-Zip	82.8	28.8	41.5
100%	FOCUSUI-3B	91.5	43.8	53.4
50%	+ 完整设置	91.4	42.3	54.6
30%	+ 完整设置	91.0	40.6	51.8

表11. 与研究2的通用视觉令牌剪枝方法进行详细比较，见正文中的图1(c)。

C. 更多实验结果

C.1. 有效视觉选择：Patch Recall@K%

我们使用 **Patch Recall@K%** 验证视觉令牌选择的有效性，该指标定义为：在按显著性排序的前 K% 图像块中捕获的真实标注（GT）区域所占的比例

图像块（即前 K 个视觉令牌）：

$$\text{Patch Recall@}K\% = \frac{|\,\text{GT positive regions in top } K\%\,|}{|\,\text{Total GT positive regions}\,|}$$

其中，真实标注的正区域对应于与给定指令配对的用户界面元素区域。我们评估 $K \in \{5\%、10\%、25\%、50\%\}$。此外，我们报告**完全覆盖预算**，即完全覆盖真实标注元素所需的视觉标记百分比。结果见表12。

模型	Patch Recall@K% ↑ @5%	@10%	@25%	@50%	平均值	完整覆盖预算 ↓
零样本基线						
随机	0.05	0.11	0.26	0.51	0.23	0.85
CLIP	0.12	0.21	0.41	0.65	0.35	0.61
我们的查询引导显著性评分器						
FOCUSUI-3B	0.39	0.56	0.83	0.96	0.69	0.25
FOCUSUI-7B	0.43	0.60	0.84	0.97	0.71	0.24

表12. **Patch Recall@K%** 和**完全覆盖预算**在ScreenSpot-Pro基准上的性能比较。

C.2. 对 POSPAD 的分析

为更好地理解保留原始空间布局的影响，我们进一步分析了 POSPAD 在被丢弃视觉词元的连续序列中的放置位置，并比较了三种变体：(i) 序列首位、(ii)序列中间和 (iii)序列末尾 （我们提出的 POSPAD）。

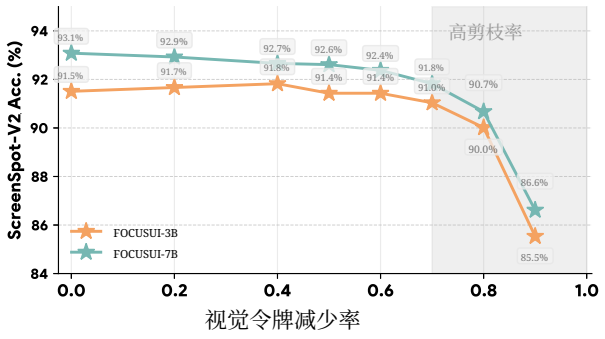
序列类型	SS-Pro 平均值			
	$r = 100\%$	$r = 75\%$	$r = 50\%$	$r = 25\%$
序列首位	42.0	41.8	39.8	36.8
序列中间	41.2	41.1	38.7	33.5
序列末尾 (POSPAD)	42.3	42.1	40.4	37.7

表 13 . POSPAD 令牌的不同位置 n.

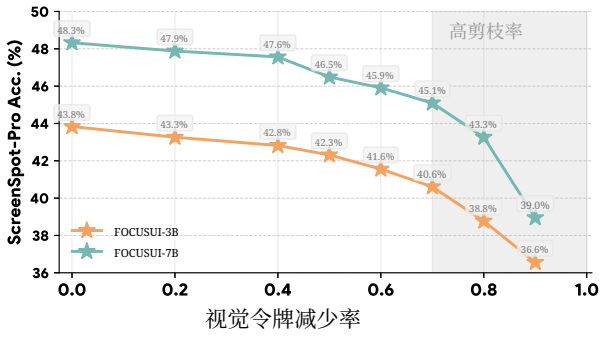
表 13 报告了一项研究，该研究改变了视觉标记保留率 r 以及 POSPAD 标记的位置。在所有保留率下，将 POSPAD 放在每个被丢弃序列的末尾都能取得最佳性能，尤其是在保留率较低时。实证结果证实了我们的直觉：将 POSPAD 放在序列末尾，更符合视觉编码器和 M-RoPE 使用的光栅扫描顺序。相比之下，将 POSPAD 放在序列开头或中间，会使整个区域偏向更早的位置，从而增加语言模型解码器与原始空间布局对齐的难度。

C.3. 不同比例下的详细性能

图7展示了 FOCUSUI 在 ScreenSpot-V2 和 ScreenSpot-Pro 上不同缩减比例（1 –保留率 r）下的详细性能。结果表明，即使大幅削减视觉令牌，FOCUSUI 仍能保持较高的用户界面定位准确率。



（a）ScreenSpot-V2 上性能与缩减比例的关系。



（b）ScreenSpot-Pro 上性能与缩减比例的关系。

图 7. 不同令牌缩减比例下的用户界面定位准确率。

C.4. 定性示例

图8 展示了 ScreenSpot-V2 和 ScreenSpot-Pro 上的显著性图，表明我们的查询引导显著性评分器能够有效突出与指令相关的区域，同时抑制背景。我们发现，对于简单任务（图 8(a, d)），显著性分数在真实位置处显著达到峰值。在更复杂的场景（图8 (b, c)）中，尽管分数可能不那么集中，模型仍能成功区分潜在目标与无关的背景元素。

D. 提示模板

FocusUI（基于 Qwen2.5-VL 基础模型）以下是 FOCUSUI-3B 和 FOCUSUI-7B 的系统提示。

```
你是一个 GUI 智能体。给定当前 GUI 的屏幕截图和一条人工指令，
```

FocusUI（基于 Qwen3-VL 基础模型）以下是 FOCUSUI-QWEN3-VL-2B 的系统提示。

```
你是一个 GUI 智能体。你的任务是定位与指令对应的屏幕元素。
```

Qwen2.5-VL 以下是用于评估 Qwen2.5-VL 模型的系统提示。

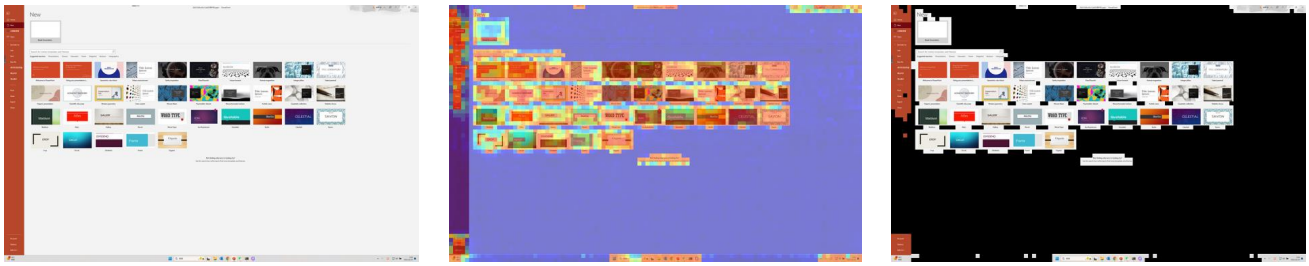
```
你是一个 GUI 智能体。你的任务是找到与指令对应的屏幕元素。
```

Jediand Qwen3-VL 以下是用于评估 Jedi-3B、Jedi-7B 和 Qwen3-VL 模型的系统提示。

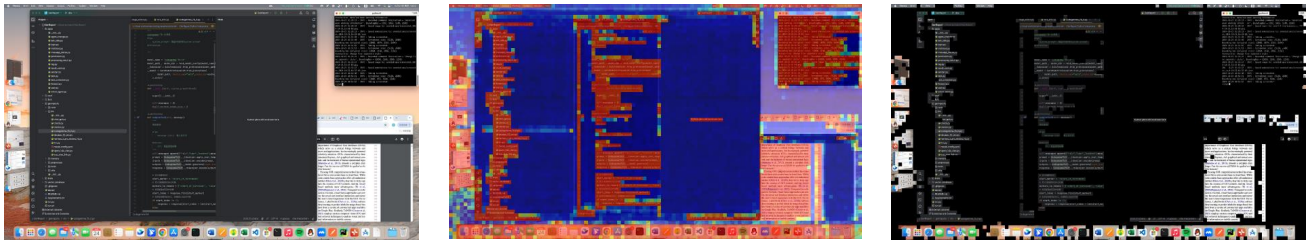
```
# 工具
你可以调用一个或多个函数来协助处理用户查询。

我们在<tools></tools> XML 标签中提供了函数签名：<tools>

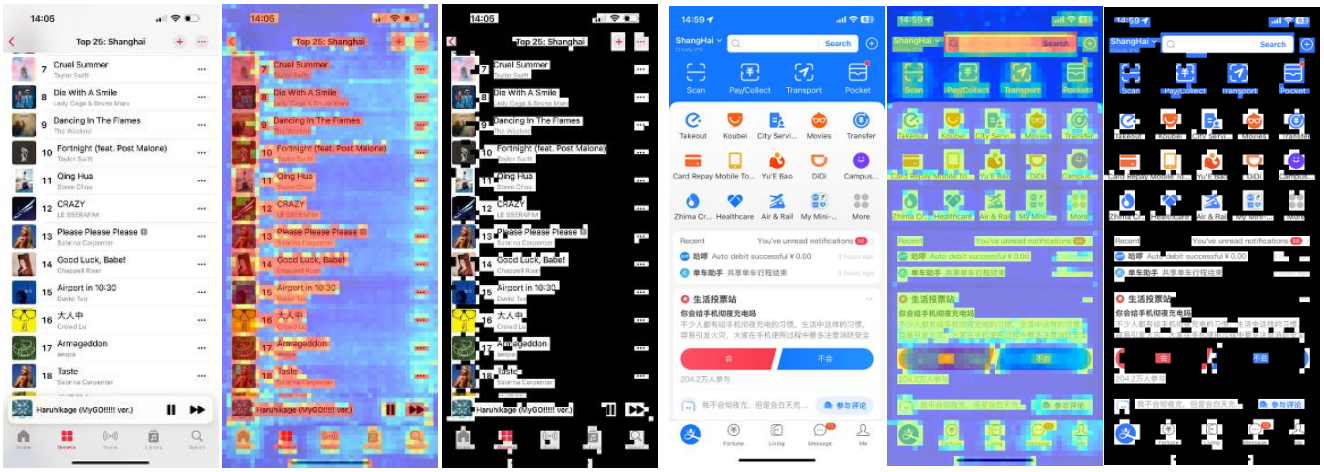
    {"type": "function", "function": {"name": " 计算机_use", "
```



*(a)*指令查询：“创建一份迷幻风格、色彩鲜艳的演示文稿。”



(b) 指令查询：“查看层级。”



*(c)*指令查询：“暂停正在播放的音乐。”

(d) 指令查询：“点击搜索栏。”

图8. 预测的每图像块显著性定性示例。**左侧：**原始屏幕截图；**中间：**预测的显著性图；以及**右侧：**视觉令牌选择结果，比例为 $r = 30\%$。

时，光标尖端应位于元素中心。除非另有要求，否则不要点击

顶部边缘）坐标，用于将鼠标移动到该位置。仅 ‘action=

对于每次函数调用，请在 <tool_call></ tool_call> XML 标签内

参考文献

[1] Anthropic。《Claude 3.7 sonnet 系统卡》，2025年。6, 8[2] Shuai Bai、Yuxuan Cai、Ruizhe Chen、Keqin Chen、Xionghui Chen、Zesen Cheng、Lianghao Deng、Wei Ding、Chang Gao、Chunjiang Ge、Wenbin Ge、Zhifang Guo、Qidong Huang、Jie Huang、Fei Huang、Binyuan Hui、Shutong Jiang、Zhao- hai Li、Mingsheng Li、Mei Li、Kaixin Li、Zicheng Lin、Jun- yang Lin、Xuejing Liu、Jiawei Liu、Chenglong Liu、Yang Liu、Dayiheng Liu、Shixuan Liu、Dunjie Lu、Ruilin Luo、Chenxu Lv、Rui Men、Lingchen Meng、Xuancheng Ren、Xingzhang Ren、Sibo Song、Yuchong Sun、Jun Tang、Jianhong Tu、Jian- qiang Wan、Peng Wang、Pengfei Wang、Qiuyue Wang、Yuxuan Wang、Tianbao Xie、Yiheng Xu、Haiyang Xu、Jin Xu、Zhibo Yang、Mingkun Yang、Jianxin Yang、An Yang、Bowen Yu、Fei Zhang、Hang Zhang、Xi Zhang、Bo Zheng、Humen Zhong、Jin- gren Zhou、Fan Zhou、Jing Zhou、Yuanzhi Zhu 和 Ke Zhu。《Qwen3 21631, 2025 2511 2, 5, 7-vl 技术报告》。arXiv 预印本 arXiv: . . [3] Shuai Bai、Keqin Chen、Xuejing Liu、Jialin Wang、Wenbin Ge、Sibo Song、Kai Dang、Peng Wang、Shijie Wang、Jun Tang 等。《Qwen2.5-VL 技术报告》。arXiv 预印本 arXiv:2502.13923，2025年。2, 5, 6, 7, 8[4] Daniel Bolya、Cheng-Yang Fu、Xiaoliang Dai、Peizhao Zhang、Christoph Feichtenhofer 和 Judy Hoffman。《令牌合并：让你的视觉变换器更快》。发表于国际学习表征会议，2023年。8[5] Yuxiang Chai、Siyuan Huang、Yazhe Niu、Han Xiao、Liang Liu、Dingyu Zhang、Peng Gao、Shuai Ren 和 Hongsheng Li。《AMEX：面向移动端 GUI 智能体的 Android 多重标注演示数据集》。arXiv 预印本 arXiv:2407.17490，2024年。9[6] Jikai Chen、Long Chen、Dong Wang、Leilei Gan、Chenyi Z huang 和 Jinjie Gu。《V2P：从背景抑制到中心峰化，实现稳健的 GUI 定位任务》，2025 年。6, 8[7] Liang Chen 、 Haozhe Zhao 、Tianyu Liu、Shuai Bai、Junyang Lin、Chang Zhou 和 Baobao Chang。《一张图像在第 2 层之后仅值 1/2 个词元：大型视觉语言模型的即插即用推理加速》。发表于欧洲计算机视觉会议，第 19–35 页。施普林格，2024年。7, 8[8] Wentong Chen、Junbo Cui、Jinyi Hu、Yujia Qin、Junjie Fang、Yue Zhao、Chongyi Wang、Jun Liu、Guirong Chen、Yupeng Huo、Yuan Yao、Yankai Lin、Zhiyuan Liu 和 Maosong Sun。《GUICourse：从通用视觉语言模型到多功能 GUI 智能体》。发表于第 63 届计算语言学协会年会论文集（第 1 卷：长篇论文），第 21936–21959 页，2025年。8[9] Kanzhi Cheng、Qiushi Sun、Yougang Chu、Fangzhi Xu、Li YanTao、Jianbing Zhang 和 Zhiyong Wu。《Seeclick：利用 GUI 定位赋能高级视觉 GUI 智能体》。发表于第 62 届计算语言学协会年会论文集（第 1 卷：长篇论文），第 9313–9332 页，2024年。8, 9[10] Tri Dao、Daniel Fu、Stefano Ermon、Atri Rudra 和 Christo- pher R´e。《FlashAttention：具备 I/O 感知能力的快速、节省内存的精确注意力》。发表于神经信息处理系统进展，2022年。7, 9[11] Difei Gao、Lei Ji、Zechen Bai、Mingyu Ouyang、Peiran Li、Dongxing Mao、Qinchen Wu、Weichen Zhang、Peiyi Wang、Xiangwu Guo、Hengxu Wang、Luowei Zhou 和 Mike Zheng Shou。AssistGUI：面向任务的桌面图形用户界面自动化。arXiv 预印本 arXiv:2312.13108，2023。8[12] Google。Gemini 2.0 发布介绍。网址：https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024，2024。6、8[13] Boyu Gou、Ruohan Wang、Boyuan Zheng、Yanan Xie、Cheng Chang、Yiheng Shu、Huan Sun 和 Yu Su。像人类一样导航数字世界：面向 GUI 智能体的通用视觉定位。载于《第十三届国际学习表征会议》，2025。6、8、9[14] Jie Huang、Xuejing Liu、Sibo Song、Ruibing Hou、Hong Chang、Junyang Lin 和 Shuai Bai。重新审视视觉语言模型中的多模态位置编码，2025。3[15]Kaixin Li、Ziyang Meng、Hongzhan Lin、Ziyang Luo、Yuchen Tian、Jing Ma、Zhiyong Huang 和 Tat-Seng Chua。ScreenSpot-Pro：面向专业高分辨率计算机使用的 GUI 定位。载于第33届 ACM国际多媒体会议论文集，第8778–8786页，2025。6、7、9 – 。[16] Wei Li、William E Bishop、Alice Li、Christopher Rawles、Folawiyo Campbell-Ajala、Divya Tyamagundlu 和 Oriana Riva。数据规模对 UI 控制智能体的影响。《神经信息处理系统进展》，37：92130–92154，2024。9[17] Kevin Qinghong Lin、Linjie Li、Difei Gao、Zhengyuan Yang、Shiwei Wu、Zechen Bai、Stan Weixian Lei、Lijuan Wang 和 Mike Zheng Shou。ShowUI：用于 GUI 视觉智能体的视觉语言动作模型。载于《计算机视觉与模式识别会议》论文集，第19498–19508页，2025。4、6、8[18] Jizhihui Liu、Feiyi Du、Guangdao Zhu、Niu Lian、Jun Li 和 Bin Chen。HiPrune：通过视觉语言模型中的层次化注意力实现无需训练的视觉令牌剪枝，2025。7、8[19] Yadong Lu、Jianwei Yang、Yelong Shen 和 Ahmed Awadallah。用于纯视觉 GUI 智能体的 OmniParser。arXiv 预印本 arXiv:2408.00203，2024。6、9[20] Lingchen Meng、Jianwei Yang、Rui Tian、Xiyang Dai、Zuxuan Wu、Jianfeng Gao 和 Yu-Gang Jiang。DeepStack：深度堆叠视觉令牌对大语言模型而言出奇地简单且有效。载于《神经信息处理系统进展》，2024。5[21] Shravan Nayak、Xiangru Jian、Kevin Qinghong Lin、Juan A¨ Rodriguez、Montek Kalsi、Nicolas Chapados、M Tamer Ozsu、Aishwarya Agrawal、David Vazquez、Christopher Pal 等。UI-Vision：以桌面为中心的视觉感知与交互 GUI 基准。载于第四十二届国际机器学习会议，2025。6、7、9、10[22]OpenAI。计算机使用智能体。网址：https://openai.com/index/computer-using-agent，2025。6、8[23] Yujia Qin、Yining Ye、Junjie Fang、Haoming Wang、Shihao Liang、Shizuo Tian、Junda Zhang、Jiahao Li、Yunxin Li、Shijue Huang 等。UI-TARS：借助原生智能体开创自动化 GUI 交互。arXiv 预印本 arXiv:2501 12326，2025。6、8。

[24] Alec Radford、Jong Wook Kim、Chris Hallacy、Aditya Ramesh、Gabriel Goh、Sandhini Agarwal、Girish Sastry、Amanda Askell、Pamela Mishkin、Jack Clark 等。通过自然语言监督学习可迁移视觉模型。载于《国际机器学习会议》，第8748–8763页，2021年。7、8[25] Jeff Rasley、Samyam Rajbhandari、Olatunji Ruwase 和 Yuxiong He。DeepSpeed：系统优化支持训练参数量超过1000亿的深度学习模型。载于第26届 ACM SIGKDD知识发现与数据挖掘国际会议论文集，2020年。6、9[26] Jianlin Su、Murtadha Ahmed、Yu Lu、Shengfeng Pan、Wen Bo 和 Yunfeng Liu。RoFormer：采用旋转位置嵌入的增强型 Transformer。Neurocomputing，568：127063，2024年。3。[27] Fei Tang、Yongliang Shen、Hang Zhang、Siqi Chen、Guiyang Hou、Wenqi Zhang、Wenqiao Zhang、Kaitao Song、Weiming Lu 和 Yueting Zhuang。三思而后点：通过快慢系统增强 GUI 定位。arXiv 预印本 arXiv:2503.06470，2025。8[28] Peng Wang、Shuai Bai、Sinan Tan、Shijie Wang、Zhihao Fan、Jinze Bai、Keqin Chen、Xuejing Liu、Jialin Wang、Wenbin Ge 等。Qwen2-VL：提升视觉语言模型在任意分辨率下对世界的感知能力。arXiv 预印本 arXiv:2409.12191，2024。3、8[29] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、R´emi Louf、Morgan Funtowicz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander M. Rush。Hugging Face 的 Transformers：自然语言处理领域的先进技术，2020年。7[30] Qianhui Wu、Kanzhi Cheng、Rui Yang、Chaoyun Zhang、Jianwei Yang、Huiqiang Jiang、Jian Mu、Baolin Peng、Bo Qiao、Reuben Tan 等。GUI-Actor：面向 GUI 智能体的无坐标视觉定位。arXiv 预印本 arXiv:2506.03143，2025。5、6、8、9[31] Zhiyong Wu、Zhenyu Wu、Fangzhi Xu、Yian Wang、Qiushi Sun、Chengyou Jia、Kanzhi Cheng、Zichen Ding、Liheng Chen、Paul Pu Liang 等。OS-ATLAS：面向通用 GUI 智能体的基础动作模型。载于《第十三届国际学习表征会议》，2024年。6、7、8、9[32] Tianbao Xie、Danyang Zhang、Jixuan Chen、Xiaochuan Li、Siheng Zhao、Ruisheng Cao、Toh J Hua、Zhoujun Cheng、Dongchan Shin、Fangyu Lei 等。OSWorld：在真实计算机环境中对多模态智能体执行开放式任务的基准测试。《神经信息处理系统进展》，37：52040–52094，2024年。10[33]Tianbao Xie、Jiaqi Deng、Xiaochuan Li、Junlin Yang、Haoyuan Wu、Jixuan Chen、Wenjing Hu、Xinyuan Wang、Yuhui Xu、Zekun Wang、Yiheng Xu、Junli Wang、Doyen Sahoo、Tao Yu 和 Caiming Xiong。通过用户界面分解与合成扩展计算机使用定位，2025。6、7、8、9、10

[34] Long Xing、Qidong Huang、Xiaoyi Dong、Jiajie Lu、Pan Zhang、Yuhang Zang、Yuhang Cao、Conghui He、Jiaqi Wang、Feng Wu 和 Dahua Lin。PyramidDrop：通过金字塔视觉冗余削减加速大型视觉语言模型，2025年。8[35] Yiheng Xu、Zekun Wang、Junli Wang、Dunjie Lu、Tianbao Xie、Amrita Saha、Doyen Sahoo、Tao Yu 和 Caiming Xiong。Aguvis：用于自主图形用户界面交互的统一纯视觉智能体。arXiv 预印本 arXiv:2412.04454，2024年。6, 8[36] Senqiao Yang、Yukang Chen、Zhuotao Tian、Chengyao Wang、Jingyao Li、Bei Yu 和 Jiaya Jia。Visionzip：在视觉语言模型中，更长固然更好，但并非必需。载于计算机视觉与模式识别会议论文集，第19792–19802页，2025。7, 8[37] Yuhao Yang、Yue Wang、Dongxu Li、Ziyang Luo、Bei Chen、Chao Huang 和 Junnan Li。Aria-UI：图形用户界面指令的视觉定位。载于计算语言学协会年会论文集：ACL 2025 Findings，第22418–22433页，2025。8[38] Bofei Zhang、Zirui Shang、Zhi Gao、Wang Zhang、Rui Xie、Xiaojian Ma、Tao Yuan、Xinxiao Wu、Song-Chun Zhu 和 Qing Li。TongUI：通过从多模态网络教程中学习来构建通用图形用户界面智能体。arXiv 预印本 arXiv:2504.12679，2025。6, 8[39] Ce Zhang、Kaixin Ma、Tianqing Fang、Wenhao Yu、Hongming Zhang、Zhisong Zhang、Yaqi Xie、Katia Sycara、Haitao Mi 和 Dong Yu。VScan：重新思考高效大型视觉语言模型的视觉令牌缩减，2025。8[40] Qizhe Zhang、Aosong Cheng、Ming Lu、Zhiyong Zhuo、MinQi Wang、Jiajun Cao、Shaobo Guo、Qi She 和 Shang- hang Zhang。 [cls] 注意力就是你所需的一切：用于免训练视觉标记剪枝，加速视觉语言模型推理。arXiv 预印本 arXiv:2412.01818，2024年。8[41] Yuan Zhang、Chun-Kai Fan、Junpeng Ma、Wenzhao Zheng、Tao Huang、Kuan Cheng、Denis A. Gudovskiy、Tomoyuki Okuno、Yohei Nakata、Kurt Keutzer 和 Shanghang Zhang。SparseVLM：用于高效视觉语言模型推理的视觉标记稀疏化。载于国际机器学习会议，2025。8[42] Shuyan Zhou、Frank F Xu、Hao Zhu、Xuhui Zhou、Robert Lo、Abishek Sridhar、Xianyi Cheng、Tianyue Ou、Yonatan Bisk、Daniel Fried 等。Webarena：用于构建自主智能体的真实网页环境。arXiv 预印本 arXiv:2307.13854，2023。8