DiPECS 结题研究报告

——Android 端侧 AIOS 的预测质量、资源动作与治理边界评估

结题研究报告

项目名称：DiPECS

作者单位：DiPECS Team

报告类型：结题研究报告

完成时间：2026 年 7 月 5 日

目录

1 引言 **3**

2 相关工作 **3**

3 系统与方法 **4**

3.1 系统架构 4

3.2 评估维度 4

3.3 测量环境 5

4 实验结果 **5**

4.1 决策延迟 5

4.2 Next-App 预测质量 6

4.3 资源开销与运行稳定性 7

4.3.1 管线吞吐 7

4.3.2 设备内开销 8

4.3.3 稳定性 9

4.4 隐私与治理边界 9

4.4.1 隐私空气隙 9

4.4.2 策略引擎 10

4.5 动作级收益 10

4.6 动作执行闭环 11

4.7 信号到动作的映射覆盖 12

5 讨论与未来工作 **12**

6 结论 **13**

A 复现说明 **13**

DiPECS 结题研究报告

——Android 端侧 AIOS 的预测质量、资源动作与治理边界评估

摘要：目的：评估 DiPECS Android 端侧 AIOS 原型能否将本地上下文预测转化为可测量的设备侧性能与资源收益，并在该过程中维持必要的隐私保护与动作治理边界。**方法：**本研究从决策延迟、next-app 预测质量、资源开销、运行稳定性、隐私边界、治理边界、动作收益和动作执行闭环八个维度展开评估，综合使用确定性测试、Android 模拟器测量、Pixel 6a 真机实验、LSApp 真实移动应用使用轨迹以及真实云端 API 采样。**结果：**LSApp 标准分割上 DiPECS ensemble 达到 hit@1 56.509%、hit@5 84.588%、MRR@5 0.671，高于 StrongPredictiveActionBaseline 的 hit@1 53.784%；Pixel 6a 上 PreWarm、PrefetchFile 与升级后的 ReleaseMemory `cache:volatile` 均通过 n≥20 动作收益验收，其中 PrefetchFile 命中读均值 79.993 ms、miss 下载 + 读取均值 1860.332 ms，ReleaseMemory 在真压力下带来 55 MB 级可用内存增益；核心管线在 128 ms 内处理 1631 条事件，峰值内存约 11 MB；隐私空气隙将敏感文本泄漏从 22 条降至 0 条、模型输入从 63 KB 降至 645 B。**结论：**DiPECS 的主要贡献是在本地治理边界内，将 Android 安全约束下的 PreWarm、PrefetchFile 与应用自有易失内存释放转化为动作级正收益。实验结果表明，该原型已经形成从上下文感知、策略决策、授权执行到收益度量的完整闭环，并为后续系统态与内核态资源管理扩展提供了可复用的评估框架。

关键词：结题评估；Android AIOS；next-app 预测；动作收益；资源管理；Pixel 6a 真机评测

1 引言

端侧人工智能操作系统（AIOS）需要在本地上下文采集、模型辅助决策与系统动作执行之间建立可验证的边界。DiPECS 围绕这一需求构建 Android 平台原型系统：系统通过分层管线将 Android 本地事件转换为结构化上下文，在本地策略审查后执行授权动作，并通过 Pixel 6a 真机实验验证部分动作能够带来可测量的延迟或内存收益。

结题阶段需要回答的核心问题是：DiPECS 的分层管线能否在端侧同时满足隐私保护、动作治理、低开销运行和可测量动作收益四类要求。为回答该问题，本报告从延迟、next-app 预测质量、资源开销、运行稳定性、隐私边界、治理边界、动作收益和动作执行闭环八个维度进行测量；不同证据按确定性测试、LSApp 真实轨迹、模拟器实测、Pixel 6a 真机实验和云端直采分别标注，以保证结论口径清晰。

2 相关工作

AIOS 与 LLM OS 相关工作将大语言模型代理从单个应用提升到系统级平台，为端侧智能系统提供了重要背景[1;2]。OS-Copilot、FRIDAY、UFO 等桌面代理证明模型可以

跨应用完成复杂任务，但其依赖的高权限 GUI 自动化接口难以直接迁移到 Android 普通应用环境[3–5]。

在移动行为预测方面，FALCON、AppUsage2Vec、DeepAPP 等工作说明，利用位置、时间、历史序列等上下文预测用户下一次启动应用并做预加载，是可研究的系统优化方向[6–8]。然而，这些工作主要关注预测精度，对动作授权、隐私隔离和可审计执行的系统化处理相对有限。

DiPECS 与上述工作的主要差异在于：系统并不以模型能力或隐私边界本身作为最终目标，而是以 Android 端侧资源动作收益为核心评价对象；LLM 或统计模型仅生成结构化建议，最终执行权和资源预算控制均保留在本地策略引擎中。

3 系统与方法

3.1 系统架构

DiPECS 的架构遵循 “机制与策略分离” 原则。Android 侧 Kotlin 采集器负责权限申请、通知监听、前台服务和设备状态采集；Rust 核心负责事件 schema、隐私隔离、窗口聚合、决策路由、策略审查、动作生命周期和审计记录。采集层与核心层之间通过 JSONL 交换数据，使同一套 Rust 逻辑既能处理真实 Android trace，也能处理离线评估夹具。

3.2 评估维度

本报告从以下八个维度评估系统：

(1) **决策延迟**：本地后端与云端 LLM 的决策耗时；

(2) **next-app 预测质量**：LSApp 标准分割与冷启动分割下的 hit@k 与 MRR；

(3) **资源开销**：核心管线 replay 与设备内运行的 CPU、内存占用；

(4) **运行稳定性**：短窗口持续运行中的内存增长与 CPU 占用；

(5) **隐私边界**：原始敏感文本是否越过 PrivacyAirGap；

(6) **治理边界**：PolicyEngine 对越权动作的拒绝能力；

(7) **动作收益**：PreWarm、PrefetchFile 与 ReleaseMemory 的真机 hit/miss、压力收益和强基线对照；

(8) **动作执行闭环**：AndroidAdapter 经 HMAC 校验后在设备端执行并落审计。

3.3 测量环境

测量环境分为四类，报告中均明确标注：

- **离线测试基准**：工作区内 `cargo test` 驱动，确定性、可在 CI 复现；用于决策延迟、replay 资源开销、隐私边界、治理边界和映射覆盖。
- **LSApp 轨迹评估**：使用 LSApp 顺序移动应用使用数据，按标准时间分割与冷启动用户分割评估 next-app 预测排序器。
- **模拟器实测**：使用 Android Studio 模拟器 AVD `dipecs_e2e`（system image 为 `android-35;google_apis;x86_64`），经 adb 采样；用于设备内资源开销、稳定性、启动/帧率采样和动作执行闭环[9;10]。
- **Pixel 6a 真机实测**：使用 Pixel 6a（bluejay，序列号 `2B071JEGR05551`）经 adb 和 Android action bridge 采集；用于 PreWarm、PrefetchFile、ReleaseMemory 的 n≥20 动作收益验收、动作回执延迟扫描和短窗口资源开销。
- **云端直采**：调用真实 DeepSeek API；用于云端延迟与场景决策质量。本报告按实验配置记录模型名为 `deepseek-v4-flash`，具体版本以服务商文档为准[11]。

口径声明：报告按证据来源分层呈现结果。离线测试用于确定性回归，LSApp 用于真实移动应用序列上的预测质量评估，模拟器用于闭环与资源量级验证，Pixel 6a 真机用于动作收益验收，云端直采用于延迟与语义能力采样；不同层级的结果不混用，以保证结论口径清晰。

4 实验结果

4.1 决策延迟

本地后端（规则/轻量评估器）在亚毫秒级完成决策，云端 LLM 往返需数秒，相差 4~5 个数量级。这是“本地优先路由、云端仅处理复杂语义”策略的核心依据。表 1 给

出了离线基准测试结果。

表 1: 决策延迟：本地后端 vs 云端 LLM（离线基准）

后端	均值	p50	p95
RuleBased	<0.01 ms	<0.01 ms	0.02 ms
LocalEvaluator	0.01 ms	0.01 ms	0.05 ms
CloudLLM (DeepSeek deepseek-v4-flash)	6958.16 ms	7339.61 ms	10050.08 ms

云端直采 API 实测（morning-routine 场景，5 轮）进一步佐证了量级：p50 = 11331 ms、p95 = 12963 ms、成功率 100%、错误 0。四个真实场景（circuit-breaker、low-battery、morning-routine、rich-workflow）均端到端产出结构化 intent（如 `SwitchToApp`、`CheckNotification` + `HandleFile`）、0 错误，延迟 6∼14 秒。对应数据文件见 `data/evaluation/` 目录下的 `cloud-latency-*.json` 与 `cloud-scenarios-*.json`（真实采集日期为 2026-07-01）。

4.2 Next-App 预测质量

预测质量决定 PreWarm 与 PrefetchFile 等前瞻性动作的命中率上限。为避免仅与简单基线比较，本报告采用 LSApp 真实移动应用使用轨迹，并报告强预测基线与 DiPECS ensemble 在同一任务上的结果。LSApp artifact 共包含 1,361,978 个样本；标准分割使用 1,089,459 个训练样本和 272,519 个测试样本，冷启动分割使用 977,842 个训练样本和 384,136 个测试样本。

表 2: LSApp next-app 预测结果（Top-k 与 MRR@5）

分割	排序器	hit@1	hit@3	hit@5	MRR@5
Standard	`global_popularity`	13.585%	32.538%	45.892%	0.248
Standard	`markov`	39.624%	66.318%	77.154%	0.538
Standard	`strong_predictive`	53.784%	72.563%	80.428%	0.638
Standard	`adaptive_predictive`	52.378%	69.487%	78.299%	0.619
Standard	**`ensemble`**	**56.509%**	**76.059%**	**84.588%**	**0.671**
ColdStart	`global_popularity`	11.778%	23.328%	46.848%	0.224
ColdStart	`markov`	17.901%	35.492%	45.172%	0.278
ColdStart	`strong_predictive`	48.050%	58.263%	63.724%	0.537
ColdStart	`adaptive_predictive`	48.813%	61.489%	**67.871%**	0.558
ColdStart	**`ensemble`**	**50.446%**	61.384%	66.335%	**0.562**

标准分割上，ensemble 相比 `strong_predictive` 的 hit@1 提升 2.725 pp，hit@5 提升 4.160 pp，MRR@5 提升 0.033。冷启动分割上，ensemble 仍保持最高 hit@1（50.446%）和 MRR@5（0.562），说明系统在用户无重叠场景下仍能利用跨用户应用序列规律形成有效排序。该预测结果不仅是独立模型指标，也直接进入后续 PreWarm 与 PrefetchFile 的同预算净收益计算。

4.3 资源开销与运行稳定性

4.3.1 管线吞吐

对 2400 行合成大 trace（`android_synthetic_large.redacted.jsonl`，1631 条有效事件）做确定性 replay，结果见表 3。

表 3: 核心管线 replay 资源开销

指标	数值
Wall time	128.0 ms
Peak RSS	10.77 MB
Events ingested	1631
Windows closed	58
Actions authorized	206
Throughput	12742.2 events/s

核心管线开销较低：系统在百毫秒级处理 1600+ 条有效事件，峰值内存约 11 MB，满足端侧常驻管线的资源预算要求。

4.3.2 设备内开销

表 4 给出了模拟器上 30 样本/模式的资源开销测量结果。

表 4: 设备内资源开销（模拟器，30 样本/模式）

模式	Avg CPU	Avg RSS	Avg PSS	Avg jank
baseline_idle	0.493%	118.30 MB	36.02 MB	0.0%
dipecs_observe_only	0.387%	125.87 MB	39.63 MB	0.0%
dipecs_action_loop	0.0%	132.80 MB	41.62 MB	0.0%

结果解释：模拟器环境下 CPU 读数存在正常方差（两次运行分别约为 1.15% 与 0.4%，后者低于基线），因此 RSS/PSS 增量是更稳定的资源开销口径。可稳定报告的是：RSS 每叠加一层约 +7 ∼ 8 MB（采集层 +7.6 MB，动作回路再 +6.9 MB），jank 均为 0。

Pixel 6a 短窗口资源采样也显示控制面开销量级较低：observe-only PSS 均值 38.946

MB，action-loop PSS 均值 40.726 MB，平均 jank 均为 0.0%。该结果作为模拟器资源开销结论的真机交叉验证。

4.3.3 稳定性

表 5 给出了 4 分钟、8 样本、30 s 间隔的稳定性采样结果。

表 5: 稳定性采样（4 分钟、8 样本、30 s 间隔）

指标	数值	阈值	结论
RSS 变化	−5.41 MB	—	未增长
PSS 增长/小时	3.91 MB/h	<20	通过
RSS 增长/小时	6.08 MB/h	<50	通过
平均 CPU	0.95%	<10	通过

短窗口稳定性采样中 RSS 未增长，表明系统具备作为设备常驻服务的基础。

4.4 隐私与治理边界

4.4.1 隐私空气隙

在 circuit-breaker 场景下对比 “无 DiPECS（naive 云端 prompt）” 与 “有 DiPECS”，结果见表 6。

表 6: 隐私边界：有/无 DiPECS 对比

指标	无 DiPECS	有 DiPECS
原始通知文本片段数	300	300
泄漏到模型输入/审计的数量	22	**0**
Prompt/模型输入大小	63178 bytes	645 bytes

`PrivacyAirGap` 使 `raw_title/raw_text` 等敏感原文不流入模型输入或审计流，同

时将输入从 63 KB 压缩到 645 B，从而同时降低隐私风险与模型侧输入成本。辅证包括：`replay_audit_leak_test` 2/2 通过（审计流与 NDJSON 均不含 raw PII）；`golden_trace_integration_test` 6/6 通过（确定性审计 hash 捕获任何管道状态变化）。

4.4.2 策略引擎

`policy_engine_test` 20/20 通过，覆盖：高风险默认拒绝、中风险按 capability 决定、低置信度拒绝、target 不在 context 拒绝、每 batch 动作上限、deferred urgency 过滤、FallbackNoOp 拦截 PreWarm。即使后端建议了动作，`PolicyEngine` 仍按风险/capability/target-in-context 多重规则复审，防止越权执行。

4.5 动作级收益

动作的价值不仅在于完成派发，还在于能否在同一预算下改善设备侧指标。上一节的 LSApp hit@1 结果为动作命中率提供了真实轨迹来源，本节进一步把命中率、设备侧 hit/miss 成本和控制面开销合成为净收益。表 7 汇总了当前通过真机动作收益验收的实验结果。

表 7: Pixel 6a 真机动作收益验收

动作	真机测量结果	适用范围与推广路径
`PreWarmProcess own:*`	n=20/mode；collector cold mean/p95 为 710.75/733 ms，prewarm hit mean/p95 为 201.55/213 ms；接入 LSApp standard hit@1 后 DiPECS `net_benefit_ms`=76,068,875.158，高于强预测基线 72,283,770.198。	当前聚焦 Android 安全约束下的自有资源预热；第三方应用级预热可在系统签名或系统服务部署后扩展。
`PrefetchFile`	n=20/mode；399,165-byte HTTPS 目标；prefetched read mean/p95 为 79.993/101.332 ms，miss fetch+read mean/p95 为 1860.332/2276.297 ms；投影 DiPECS 净收益 61,268,324.531 ms，高于强预测基线 34,859,928.678 ms。	设备直接测每次命中节省与 miss 成本；DiPECS-vs-strong 使用真实 LSApp hit@1 投影同预算收益。
`ReleaseMemory cache:volatile`	n=20/mode；512 MB 非 root 匿名内存压力 + 64 MB 应用自有易失缓存；可用内存增益 +55,158.6 KB，PSS 降幅增益 +64,621.3 KB，Welch p=0.00026891，jank 变化 0.0 pp。	当前聚焦应用自有易失内存释放；磁盘 prefetch cache 清理由存储缓存管理语义承接。

结果解释：PreWarm 与 PrefetchFile 的强基线对照均使用同一 LSApp standard split 的 hit@1 输入；设备侧直接测量 hit/miss 成本和控制面开销。ReleaseMemory 的收益结论来自 `cache:volatile`，即同一进程内可丢弃内存缓存的受控释放。三类动作分别覆盖启动、文件等待和内存压力三个资源管理维度。

4.6 动作执行闭环

四类可转发动作（`NoOp` 永走本地 stub 不转发）在模拟器端逐一执行并见终态审计事件；Pixel 6a 真机动作延迟扫描则给出了同类设备回执延迟，结果见表 8。

表 8: 四类动作设备端执行覆盖（Pixel 6a 动作延迟扫描）

动作类型	处理器终态审计事件	设备确认延迟	状态
KeepAlive	`keep_alive_job_executed`	973 μs	EXECUTED
ReleaseMemory	`release_memory_completed`	971 μs	EXECUTED
PreWarmProcess	`own_resources_prewarmed`	841 μs	EXECUTED
PrefetchFile	`prefetch_succeeded`	1964 μs	EXECUTED

链路为：`AndroidAdapter` → execute 信封（canonical HMAC-SHA256 + freshness window）→ 设备校验 → dispatch → 各处理器执行落审计。拒绝与失败类事件均为 0，且脱敏边界保持有效。模拟器 action-loop 记录 `authorized_action_socket_execute_ok = 4`；Pixel 6a 的 PrefetchFile 收益实验进一步证明异步下载链路能在真机 HTTPS 目标上落到 `prefetch_succeeded`。

设备内直连回路：把交叉编译的 `dipecsd` 推进模拟器内运行，直连 `127.0.0.1` 的 app 动作 socket、无 adb forward，判定 LOOP-CLOSED。这结构上根除了“开发机经 adb forward”通道特有的代理层数据/FIN 竞态，是最接近生产部署（`/system/bin/dipecsd`，与 app 同机走内核 loopback）的设备内回路。离线侧另有 `android_bridge_e2e_test.rs` 使用模拟 socket 独立重算 canonical HMAC，作为协议一致性的交叉验证。

4.7 信号到动作的映射覆盖

`noop_matrix` 覆盖 13 个信号模式，其中 9 个（69.2%）产出真实动作，其余低置信度或低收益场景保守回退。典型映射包括：`low_battery_release_memory` → ReleaseMemory、`file_access_prefetch` → PrefetchFile、`app_foreground_keepalive` → PreWarmProcess + KeepAlive。系统能把常见设备语义映射为维护动作，并在低置信度场景中保持动作预算克制。

5 讨论与未来工作

本报告的测量数据来自确定性测试、Android 模拟器、Pixel 6a 真机和真实云端 API；结果口径已在正文中分项说明。当前验证覆盖了隐私边界、治理边界、核心管线性能、

动作执行闭环和动作级收益等核心命题。后续工作可从以下方向继续深化：

(1) **长期观测与真实用户研究**：将短窗口动作验收扩展为真实用户场景下的匿名化真实标签、长期功耗和稳定性观测。

(2) **系统态/内核态资源管理路径**：沿 issue #115 推进 platform-signed `/system/bin/dipecsd` 或 system image 集成，把 KeepAlive 从 Android app 处理器迁移到系统部署形态；后续可结合 `oom_score_adj`、lmkd 接口与 sepolicy，把用户态动作治理层连接到更接近内核资源管理的执行边界。

(3) **动作撤销与预算**：在 PreWarm 与 PrefetchFile 已有同预算强基线投影的基础上，加入资源预算、错预热撤销和长期 cache 压力控制。

(4) **信号覆盖扩展**：继续扩展低置信度场景的信号融合策略，在保证动作预算克制的前提下提升有效动作覆盖率。

6 结论

DiPECS 在结题阶段以测量数据支撑了核心命题：Android 本地上下文可以转化为可评测的资源动作收益。LSApp next-app 预测结果表明，DiPECS ensemble 在标准分割和冷启动分割上均超过强预测基线的 hit@1；Pixel 6a 上三类 Android 安全约束下的动作级证据也已经补齐：PreWarm 与 PrefetchFile 的同预算净收益均高于 StrongPredictiveActionBaseline，ReleaseMemory `cache:volatile` 在真压力下显著改善 PSS 和可用内存。分层管线同时保持了较低资源占用（replay 128 ms/11 MB、设备内每层 $+7 \sim 8$ MB、短窗口运行未观察到内存泄漏），并控制了隐私泄漏（22 $\rightarrow$ 0 泄漏、63 KB $\rightarrow$ 645 B）与治理边界（策略 20/20 复审）。

综上，DiPECS 并非仅提供动作派发或隐私保护能力，而是在受控边界内将 Android 资源动作转化为可量化收益。后续沿 system image、platform-signed daemon 与更底层资源管理接口推进后，该控制面可以进一步从应用层原型演进为更接近操作系统资源管理器的 AIOS 执行层。

A 复现说明

本报告的测量数据可基于以下信息复现：

- **代码版本**：`23ecaa34d35181a0f3070d258b9e5fb5833bd516`（main 分支，#97 合并后）。
- **离线测试**：在仓库根目录执行 `cargo test --workspace`，可复现隐私、治理、聚合和 replay 相关测试。
- **模拟器实测**：使用 Android Studio 创建 AVD `dipecs_e2e`，system image 为 `android-35;google_apis;x86_64`[9;10]。
- **Pixel 6a 真机动作收益**：见 `data/evaluation/next-app/prewarm-net-benefit-real-device-2`、`data/evaluation/action-net-benefit/prefetch-file-benefit-20260705-201053.json` 与 `data/evaluation/action-net-benefit/release-memory-pressure-benefit-20260705-18`
- **LSApp 预测质量**：见 `data/evaluation/next-app/lsapp-standard.report.json` 与 `data/evaluation/next-app/lsapp-coldstart.report.json`。
- **云端延迟数据**：见 `data/evaluation/` 目录下的 `cloud-*.json` 文件；若文件为 Git LFS 指针，请先执行 `git lfs pull`。
- **设备内闭环**：按部署文档 `docs/src/team/deployment.md` 交叉编译 `dipecsd` 并推送至模拟器运行。

参考文献

[1] Kai Mei, Shiyao Zhang, Jiawei Liu, et al. Aios: LLM agent operating system. arXiv:2403.16971, 2024.

[2] Charles Packer, Vivian Fang, Shishir Patil, et al. MemGPT: Towards LLMs as operating systems. arXiv:2310.08560, 2023.

[3] Zhiyong Wu, Chengcheng Han, Zichen Ding, et al. OS-Copilot: Towards generalist computer agents with self-improvement. arXiv:2402.07456, 2024.

[4] OS-Copilot project page. Project Page, 2024.

[5] Chaoyun Zhang, Lichao Sun, Yingjie Li, et al. UFO: A UI-focused agent for Windows OS interaction. arXiv:2402.07939, 2024.

[6] Tingxin Yan, David Chu, Deepak Ganesan, et al. FALCON: Fast app launching via predictive user context. Microsoft Research, 2012.

[7] Sha Zhao, Julian Ramos, Ziwen Jiang, et al. AppUsage2Vec: Modeling smartphone app usage for prediction. IEEE Xplore, 2019.

[8] Zhihao Shen, Zhefu Yan, Zhongmin Chen, et al. DeepAPP: A deep reinforcement learning framework for mobile application usage prediction. ACM Digital Library, 2019.

[9] Android Developers. Run apps on the Android Emulator. Android Developers, 2024.

[10] Android Developers. Android 15 (API 35). Android Developers, 2024.

[11] DeepSeek. Deepseek API documentation. DeepSeek API Docs, 2025.